Causal Surrogate Models

Adding "What If" to Cyber-Physical System Testing

Richard J. Somers

Supervisors: Neil Walkinshaw and Robert Hierons

School of Computer Science
University of Sheffield
United Kingdom
June, 2025

A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

Abstract

Cyber-physical systems (CPS) are becoming more prevalent, especially in human-interacting environments. Identifying incorrect behaviour through software testing is, therefore, paramount to their use. Surrogate-assisted testing approaches aim to effectively test such systems by searching for scenarios that may result in system violations by using surrogate models to search for potential violations and evaluating those scenarios on a high-fidelity simulator. However, the development of surrogate models requires curated datasets to accurately represent system behaviour. Such datasets are typically unavailable for CPSs due to the limitations and expense of physical execution, especially for human-interacting systems. Pre-existing datasets may be used instead, which may contain spurious associations or may not cover all behaviours (a lack of controllability). Unmeasurable environmental factors may also affect system behaviour, making system outputs appear inconsistent for a given input (a lack of observability). In this thesis, we use a motivating example of an artificial pancreas system (APS) to investigate the limitations of testing such a system.

To account for the lack of observability and controllability of CPSs, we define a causal surrogate model to enable more effective testing of their behaviour. This surrogate model integrates with existing surrogate-assisted testing techniques. Our surrogate model uses causal inference, which can account for bias in pre-existing datasets and assess the expected causal relationships between variables. As a result, we enable the testing of systems for which curated data may not be available or that exhibit behaviours affected by external factors.

We perform two evaluations, first replicating an existing study of surrogate-assisted CPS testing on an automated driving system (ADS). In this evaluation, our results demonstrate how our approach found system violations with less computational expense than the state-of-the-art. We then test a more complex, safety-critical APS. However, to test the APS, we first develop and validate a digital twin of a person using an APS to act as a high-fidelity simulator. The APS can, therefore, be disconnected from the human-in-the-loop, allowing for the testing of potentially dangerous scenarios without clinical trials. Our causal surrogate model demonstrates the ability to uncover over double the number of violations using real-world clinical APS data, compared to the state-of-the-art surrogate-assisted testing approach. We show how causal surrogate models can alleviate the requirement of curated data for systems testing and present a novel way of navigating and finding system violations for inconsistent system behaviour.

Acknowledgements

I would first like to thank my supervisors for their help throughout the past years. Neil, you have always pushed me to learn and ensure my work is of great quality. I appreciate the effort you have put into my supervision and I am grateful for your guidance. Rob, your knowledge and sheer understanding of the academic process has been invaluable. Thank you for your understanding and patience as I learned over the past few years.

I would also like to thank my other colleagues: fellow PhD students, RAs, lecturers in the department and RSEs. The testing lab has made for a social and welcoming environment in which to conduct my research. I appreciate the friendly atmosphere and, especially, how open people were to being convinced to join me for running races in Sheffield. I would like to especially thank Michael for his constant willingness to help over the past few years and his patience while I grappled with causal inference.

I would like to thank my family and friends for their support during this process. To mum and dad for their ongoing interest in my work and patience as I ranted over failed experiments. David and others, you were a great distraction when necessary and a constant reminder to ensure a proper work life balance.

Contents

List of Figures

List of Tables

Acronyms

ADS Automated Driving System.

APS Artificial Pancreas System.

ATE Average Treatment Effect.

BG Blood Glucose.

CGM Continuous Glucose Monitor.

COB Carbohydrates On Board.

CPS Cyber-Physical System.

CS-RQ Case Study Research Question.

D_{KL} Kullback-Leibler Divergence.

DAG Directed Acyclic Graph.

DIY Do-It-Yourself.

IOB Insulin On Board.

RMSE Root Mean Squared Error.

RQ Research Question.

SUT System Under Test.

T1DM Type 1 Diabetes Mellitus.

TIR Time In Range.

Chapter 1. Introduction

Cyber-physical systems (CPS) present a challenge for software testers, combining the complexities of both physical and software components [140, 167]. Physical testing may require human interaction, resulting in users being put in potentially dangerous scenarios [19, 81, 150]. This difficulty is further exacerbated by the complex architectures of such systems and the long, expensive setups and runtimes of physical testing [49, 143].

CPSs are prevalent across many different human-interacting domains. These systems interact directly with human users or exist in human-collaborative environments [12,69]. To ensure such dynamic and human-interfacing systems act correctly, their behaviour must be tested. For example, an automated driving system (ADS) [70] must behave correctly around pedestrians and drivers, and portable medical devices [69], such as artificial pancreas systems (APS) [168], must ensure the correct medical intervention is applied to the user.

Simulation-based testing approaches, such as in-the-loop testing [107], aim to reduce the physical challenges of testing. However, the high computational cost and large search space can make it difficult to find behaviour that results in system violations [70,120]. Due to the recent increase in adoption of human-interacting CPSs, such as that seen in ADSs, APSs [100, 180] and other human-interacting domains [69, 140, 167], it is particularly important that we have techniques that find incorrect system behaviour.

Surrogate-assisted testing approaches have been used to reduce the time required to find CPS safety violations and alleviate the physical risks associated with testing [16, 70, 116]. These approaches use surrogate models (e.g., regression models, neural networks) [16,64] to provide a less computationally expensive approximation of system behaviour. For example, a surrogate model may represent the relationships between test inputs and system violations [70]. Therefore, the complexities of the software, hardware and physical environment of the CPS are abstracted away. Surrogate models are used in conjunction with meta-heuristic search to identify test cases that are likely to cause a violation. Therefore, only scenarios that are most likely to cause system violations are executed in high-fidelity simulations.

For surrogate models to predict scenarios that result in real system violations, those predictions must be an accurate representation of system behaviour. A surrogate model is typically reliant on machine learning techniques that are trained from examples of system behaviour (training data) [16,120]. These techniques exploit the associations in the training data to generate a representation of the relationships between system inputs and system violations.

In this thesis, we identify two challenges that may impact the ability to train a suitable surrogate model and, therefore, test CPSs: (1) system behaviour being affected by unmeasurable factors that are not explicitly accounted for in the training data, and (2) the difficulty

of generating datasets that cover the system's entire specification. We will see that these challenges can reduce the effectiveness of surrogate-assisted testing for CPSs.

System behaviour may appear to be inconsistent when affected by *unmeasurable factors*. Therefore, a set of inputs may not always lead to the same behaviour. For CPSs, external and unmeasurable factors [49] make it more difficult to identify inputs that cause violations. The apparent inconsistency represents a lack of system observability [54].

For example, a person's reaction to insulin depends on the unmeasurable dynamics of the human body. An APS's specific insulin prescription may, therefore, be correct for one person but not another, depending on the unknown body dynamics of the user [142]. Such unmeasurable factors make it challenging to define "correct" CPS behaviours, increasing the difficulty of identifying behaviours that result in system violations.

Obtaining datasets that *cover* a CPS's behaviours can also be challenging. Physical data collection for CPSs can require expensive setups and potentially dangerous behaviour in human-interacting environments [49]. This difficulty in obtaining data is known as a lack of system controllability [54]. For example, for an APS, data collection could require observing the user entering potentially dangerous hyperglycaemia [81] to find scenarios where the control algorithm does not supply enough insulin.

In order to avoid physical data curation, pre-existing datasets may be used. However, pre-existing datasets may not cover all CPS behaviours and may, therefore, give rise to spurious associations. Such associations may appear in the data as a result of the data collection process and are not representative of system behaviours [58]. Surrogate models rely on machine learning approaches to estimate system behaviour based on the associations in the data [27, 136]. As a result, the testing procedure may maximise for behaviours based on spurious associations. Such behaviour may not translate to real world violations, reducing the effectiveness of surrogate-assisted testing.

Causal inference provides a suite of statistical methods that use domain knowledge to account for spurious associations and inconsistent behaviour [84]. Prior works have applied causal inference in software testing to account for such testing difficulties [27,91,141]. These techniques have used causal graphs to represent domain knowledge, providing additional contextual information to software testing.

Causal inference allows pre-existing data to be used in software testing where data generation may not be feasible. Causal graphs can be used to identify spurious associations through causal identification [135]. Estimations about system behaviour can, therefore, account for spurious associations where they may be interpreted in pre-existing datasets.

Causal inference can also be used to represent behaviour that may appear inconsistent. Causal graphs can represent the expected causal relationships between exposed system variables [27, 159]. Causal questions can then be answered about the system, such as "What if I increase a variable? Will this cause the expected increase in another variable?" Relationships that do not hold can be identified by evaluating surrogate models, highlighting contradictions in expected behaviour without testing specific values.

Causal inference-based testing, however, can suffer from the oracle problem. The tech-

niques described rely on causal graphs, which are typically manually derived, requiring the resulting graph to be accurate. A potential lack of knowledge surrounding CPS behaviours due to their uncontrollability [49] may affect the viability of our approach. We further discuss this threat to our approach in Sections 4.2.1 and 8.5.

1.1. Contributions

In this thesis, we propose the "causal surrogate model" to increase the applicability of surrogate-assisted CPS testing. Our technique aims to integrate with current surrogate-assisted testing techniques, building upon their reduction of physical system executions and ability to efficiency find system violations. Our technique also uses causal inference techniques to account for behaviours that appear inconsistent and the potential for the interpretation of spurious association.

By integrating domain knowledge, our surrogate model can make predictions that adjust for potentially spurious associations and check causal relationships of the system. A causal graph allows for variables that may introduce bias into surrogate model predictions to be identified and included in the surrogate model. The resulting predictions, therefore, adjust for this association, producing more accurate surrogate model estimations.

We also show how one can evaluate system behaviour affected by external factors by answering "what if" questions about the relationships between system variables. Causal relationships that do not hold can be identified by finding changes in variables that do not cause an expected change in other system variable. By searching for contradictions to the expected causal relationships, our surrogate model drives the testing procedure towards potentially incorrect system behaviour.

We evaluated the use of our causal surrogate model using two case studies. In these case studies, we measured the extent to which our surrogate model found system violations with less computational expense than the state-of-the-art associative surrogate modelling technique [64]. We also measured whether our surrogate model could increase the number of system violations found.

Our initial evaluation re-implemented a published study [70] of an autonomous driving system (ADS), Pylot [65]. Pylot presented an off-the-shelf CPS that has been used in prior evaluations and presents a CPS that is difficult to test due to its computational expense. Incorrect ADS behaviour can be dangerous to drivers on the road and pedestrians in the environment. Therefore, finding such incorrect behaviour is essential to their use.

We then applied our technique to a safety-critical artificial pancreas system (APS) control algorithm, oref0 [98]. APSs are a safety-critical medical device that prescribes insulin to a user with type 1 diabetes mellitus (T1DM) [43]. The insulin prescription is dependent on the user's constantly changing blood glucose-insulin dynamics [4, 95]. Correct behaviour is critical for APS use since an incorrect prescription of insulin could result in dangerous blood glucose levels for the user [81, 150].

In the first (ADS) evaluation, we found that our causal surrogate model increased the

efficiency of testing Pylot. Our approach found violations with fewer uses of the high-fidelity simulator, reducing the computational expense of testing. From this evaluation, we presented the applicability of our causal surrogate model in an existing surrogate-assisted testing domain.

In the second (APS) evaluation, our technique found over double the number of violations than the state-of-the-art technique. We also explored how our surrogate model explored system behaviours differently to the state-of-the-art to better understand this improvement. Our results suggest that the proposed approach has the potential to greatly assist the testing of systems that are affected by external factors, such as a user's blood glucose dynamics, and where data that covers the entire system specification may not be available.

To perform surrogate-assisted testing, a high-fidelity simulator is required. For our second evaluation, no simulation environment was available for testing of oref0. Therefore, we performed an additional evaluative case study on developing a digital twin to act as the simulation environment for oref0. Digital twins have been proposed to test medical devices using the following steps: obtaining clinical data, generating a digital twin and fitting it to represent this data, validating the digital twin's model, and, finally, using the digital twin to evaluate human responses to clinical interventions [37].

In this case study, we evaluated the applicability of applying these steps to observe potentially dangerous APS configurations without the human-in-the-loop. We adapted an existing blood-glucose insulin dynamics model to represent a person with T1DM. We validated that our model could represent the dynamics of a clinical dataset, make predictions about blood glucose-insulin levels and enable a tester to interface with the oref0 control algorithm.

This thesis makes the following main contributions:

- A novel causal surrogate modelling technique that enables a tester to account for inconsistent system behaviours by checking causal relationships and enables causal identification to adjust model predictions for spurious associations.

- We reproduced an existing evaluation to demonstrate the effectiveness of using our causal surrogate model compared to the state-of-the-art in surrogate-assisted CPS testing.

- We developed and validated a digital twin of a person using an APS to act as a simulation environment in surrogate-assisted testing. We modified an existing blood glucose-insulin dynamics model, interfaced it with the APS oref0, and validated its behaviour against the largest open-source T1DM dataset, which has 156 users.

- We performed a second, more in-depth evaluation of using our causal surrogate model to test a more complex, safety-critical APS. A human body in the loop provided unmeasurable external factors and real clinical data was used as an uncontrollable data

source. We used the developed digital twin as a simulation environment to validate incorrect behaviour.

The significance of our work comes from allowing developers to use pre-existing datasets instead of curating them for CPS testing. For CPSs, it is not always feasible to produce curated datasets. The proposed technique also allows testers to find incorrect system behaviour that is affected by external unmeasurable factors. In the evaluations, the proposed approach improved the effectiveness of surrogate-assisted CPS testing with pre-existing datasets by reducing the computational budget required and increasing the number of violations found.

1.1.1. Papers

Throughout this thesis, the following papers have been peer reviewed and published or are currently under peer review:

P1: **Somers, R.J**., Douthwaite, J.A., Wagg, D.J., Walkinshaw, N. and Hierons, R.M., 2023. Digital-twin-based testing for cyber–physical systems: A systematic literature review. Information and Software Technology, 156, p.107145. `https://doi.org/10.1016/j.infsof.2022.107145`

P2: **Somers, R.J**., Clark, A.G., Walkinshaw, N. and Hierons, R.M., 2022, October. Reliable counterparts: efficiently testing causal relationships in digital twins. In Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings (pp. 468-472). `https://doi.org/10.1145/3550356.3561589`

P3: **Somers, R.**, Walkinshaw, N., Mark Hierons, R., Elliott, J., Iqbal, A. and Walkinshaw, E. (2025), Configuration Testing of an Artificial Pancreas System Using a Digital Twin: An Evaluative Case Study. Softw Test Verif Reliab, 35: e70000. `https://doi.org/10.1002/stvr.70000`

P4: **Somers, R.J**., Walkinshaw, N., Hierons, R.M., Shin, D. and Foster, M. Causal Surrogates: Adding "What If" to Cyber-Physical System Testing - ACM Transactions on Software Engineering and Methodology Journal (major revisions, under review)

In Section 1.3, we describe how we have adapted the content of these papers for use in this thesis.

1.1.2. Software and Replication Packages

Throughout this thesis, the following pieces of software were developed:

S1: Causal Testing Framework - Github link: `https://github.com/CITCOM-project/CausalTestingFramework`

S2: APS Digital Twin (Replication Package for P3) - Github link: `https://github.com/CITCOM-project/APSDigitalTwin`

S3: Surrogate Assisted Testing (Replication Package for P4) - Github link: `https://github.com/CITCOM-project/SurrogateAssistedTesting`

To increase the reproducibility of our evaluations in Chapters 5 and 7, we provide a comprehensive replication package (S3). This package includes: scripts for data generation and application of our methods for both the Pylot and oref0 evaluations, and the figures generated for this thesis.

To also ensure the reproducibility of our digital twin case study in Chapter 6, we have created a reproducibility package (S2), including the code of the digital twin model of a person with T1DM, a script for applying our data cleaning procedure to the OpenAPS Data Commons, and scripts for each research question.

For both replication packages, we cannot redistribute OpenAPS data due to our data management agreement with OpenAPS Data Commons [97]. Our work was completed using version 'n=183' of the OpenAPS Data Commons.

Through this software, we made the following contributions:

1. A replication package containing the software required to implement our causal surrogate model and the reproducible methodology of our evaluation for both motivating examples.

2. A second replication package containing the software for the digital twin and accompanying scripts required to perform its validation.

1.2. Ethical Concerns

In the motivation of this thesis and our latter evaluation, we explore the testing of an open-source, unregulated APS. This may raise concerns as to whether testing such a system was encouraged its use, even though it has not been medically approved.

We justify our research into the topic by comparing it to research conducted to support those who use recreational drugs [87]. Such research aims to protect and ensure the safety of those who are going to use an unregulated substance, as opposed to encouraging it's use. We argue that people will use open-source APS implementations due to lack of technological innovations, explored in Section 3.3, so research into ensuring their safety is important.

We also use a large open-source dataset of people using this APS in our latter evaluation. To ensure such data was handled correctly, we sought approval from the University of Sheffield's Ethics Committee, resulting in a data management plan that was used to ensure the data was used and stored appropriately.

1.3. Thesis Outline

The remainder of this thesis is laid out as follows:

Chapter 2 motivates the thesis by outlining the testing difficulties associated with testing cyber-physical systems. We then examine how these testing challenges relate to the moti-

vating example of the thesis, the artificial pancreas system. This chapter is derived from the text in both P2 and P4 from Section 1.1.1.

Chapter 3 presents the background required for the remainder of the thesis. We expand upon the notions of software testing, cyber-physical systems and present the open-source artificial pancreas, oref0. We describe surrogate-assisted testing as the state-of-the-art technique for testing such systems and describe how this technique may struggle with the testing challenges identified in Chapter 2. We explain how digital twins have been used for simulating human-machine interactions and their proposed use in medical device testing to alleviate testing directly with the user. We also introduce causal inference-based testing and explore its potential in solving the testing challenges that surrogate-assisted testing may struggle with. This chapter is derived from the background sections of all papers in Section 1.1.1.

Chapter 4 presents the *causal surrogate model* to be used with existing surrogate-assisted testing techniques. We describe how causal inference techniques can be used to generate a causal model that can complement existing state-of-the-art surrogate models. We then explain how our causal surrogate model examines the expected causal relationships between variables and adjusts its estimations based on domain knowledge. This chapter is derived from P4 in Section 1.1.1.

Chapter 5 presents a replication of an existing evaluation. In this chapter, we evaluate the efficacy of our causal surrogate model in the context of a prior evaluation of an automated driving system. This chapter is derived from P4 in Section 1.1.1.

Chapter 6 presents an evaluative case study of developing a digital twin to enable the configuration testing of an artificial pancreas. We use this digital twin to act as a simulation environment for surrogate-assisted testing in Chapter 7. We describe the development of the digital twin and evaluate its accuracy with respect to the original data, prediction of blood-glucose dynamics, and its ability to observe configurations that may result in dangerous behaviour. This chapter is derived from P3 in Section 1.1.1.

Chapter 7 presents a second, more in-depth evaluation. In this chapter, we aim to test our subject system, the artificial pancreas system. Similar to Chapter 5, we evaluate the efficiency of our approach in a clinical context. We also investigate how combining data from our causal surrogate model andthe state-of-the-art surrogate model affects the diversity of the dataset used to train the surrogate models. This chapter is derived from P4 in Section 1.1.1.

Chapter 8 discusses the results of the evaluations in Chapters 5 and 7, derived from P4 in Section 1.1.1. Chapter 9 outlines the related work, derived from all papers in Section 1.1.1. Finally, Chapter 10 concludes the thesis.

Chapter 2. Motivation

In this chapter, we provide the motivation for the thesis. We present cyber-physical systems and the factors that may affect their testability: human interaction, complex input spaces, a lack of curated data, and unmeasurable factors. We then introduce the motivating example for the thesis, the artificial pancreas system and investigate how each of our testability factors may affect this system.

2.1. Cyber-Physical Systems

Cyber-physical systems (CPS) consist of both software and physical components [162]. These are software driven systems which "interact with the physical world, and must operate dependably, safely, securely, and efficiently and in real-time" [143]. They include different levels of autonomous ability, further described in Section 3.2.1. CPSs are well established as a concept and have been widely explored by other systematic reviews [140, 167] and our own published systematic review (P1) [160].

In recent years, CPSs have been adopted into human-interacting environments and domains through Industry 4.0 [72]. Industry 4.0 has seen such systems implemented in manufacturing, interacting directly with, or in proximity of people [46]. The adoption of CPSs has also spurred the implementation of human-interfacing medical devices [5, 42, 55, 69], which rely on a human user in order to operate. As a result, it is important that the systems behave correctly for both the user and others in the working environment.

2.1.1. Testability of Cyber-Physical Systems

CPSs present a challenge for testers. Physical environments and extensive test setups [49] make testing the system expensive and time consuming. However, the physical and human-interacting environments in which CPSs exist highlight the importance of ensuring system behaviour *conforms* to its specification as to not endanger their users [49,55,160]. We further explore CPS *conformance testing* in Section 3.2.2.

In order to reduce manual system execution, systems such as CPSs can be tested through the use of surrogate models [120]. Such approaches forego physical testing for testing the behaviour of models of the system instead. System models can be inferred from pre-existing execution data of the system in order to accurately represent system behaviour. The behaviour of this model can then be tested, reducing the need for expensive physical testing.

Modern testing approaches, such as surrogate-assisted testing [107] (described in Section 3.4), rely on the *testability* [54] of the system. We identify four factors which may affect the testability of CPSs: human interaction, complex input spaces, a lack of curated data,

and unmeasurable factors. In Section 2.2.1, we describe how these difficulties affect our motivating example.

Human Interaction

The introduction of CPSs into human-interacting domains [46,55,72] presents a testing challenge in itself. Systems may work in environments that are dependent on human interaction or may interact with humans directly. Test cases in which the system exhibits incorrect behaviour may put users in dangerous scenarios.

Jones [79] presents an example of this in which performing physical tests where an arc welder acts incorrectly could cause conjunctivitis, burns or inhalation of noxious fumes. As a result, performing physical tests may not be feasible.

Complex Input Spaces

The interaction between physical and software components can make CPS behaviour complex [49]. Such behaviours tend to be a result of continuous inputs from sensors and time dependencies. As a result, it can be difficult to identify a combination of specific system inputs that result in incorrect behaviour.

Nejati et al. [116, 120] present an example of this in which a satellite system has time dependency in its behaviour across four continuous sensors. Executing the system has a large computational and temporal cost. Although there are only four inputs, since they are continuous, executing different combinations of these variables is very computationally expensive. As a result, finding a specific combination that results in incorrect system behaviour through manual execution would require prohibitively long execution times.

Lack of Curated Data

Modern approaches to testing, such as surrogate-assisted testing [107] (described in Section 3.4), learn system behaviours from historical system data. However, a CPS's interaction with a complex, potentially human-interacting environment can reduce the data availability for systems testing [46, 49]. From our prior example, the potential physical hazards when using an arc welder, described by Jones [79], may reduce the availability of data that covers system behaviours. Henceforth, this is referred to as a lack of curated data.

Definition 1 (Curated Data) *Curated data is a dataset that has been generated specifically for system testing, typically covering all system behaviours to represent the system's operation.*

Due to the difficulty of curating data of a CPS, testers may rely on pre-existing data. Such datasets were not curated for system testing and, therefore, likely do not cover the system's specification. We define pre-existing datasets as follows:

Definition 2 (Pre-existing Data) *An existing dataset for a system that was not generated for a specific purpose. This data may be from prior executions of the system and has no guarantee that system behaviours or inputs will have coverage.*

We also define the inability to curate datasets for testing as a lack of system *controllability*.

Definition 3 (Controllability) *A system is* controllable *if a tester is able to generate inputs and outputs of a system such that they are able to cover the system's entire specification [54].*

Pre-existing datasets that were not curated specifically for testing likely follow the operational profile (the typical patterns of use) of the system [118]. As a result, there is no guarantee that all system behaviours are covered by such datasets. It is also possible that the datasets result in spurious associations (described in Section 3.4.1).

Unmeasurable Factors

The adaptive behaviours of CPSs may rely on unmeasurable environmental factors [49, 55]. Since a tester cannot measure or control these factors, system behaviours may appear inconsistent as they adapt to their environment. As a result, defining what behaviour *conforms* to an oracle becomes difficult. We define system behaviours which rely on unmeasurable factors as behaviours that lack *observability*.

Definition 4 (Observability) *A system is* observable *if distinct outputs are generated from distinct inputs and that the output is a function of the input values only [54].*

Defining whether behaviour that lacks *observability* is "correct" can be challenging. Shetty et al. [156] present an example of how external factors may affect a self-driving car, but may not be observable to the system. External factors, such as other driver behaviour, may result in seemingly inconsistent self-driving car behaviours for the same system inputs.

2.2. The Artificial Pancreas System

In this section, we introduce the artificial pancreas system (APS) as a motivating example for the thesis. We explore how this system is affected the the testing challenges described in Section 2.1.1 and then use it as the basis for the evaluation of our upcoming approach in Chapter 7.

APSs present a modern solution to managing the effects of type 1 diabetes mellitus (T1DM). APSs account for a person's inability to produce insulin by mimicking the behaviour of a healthy pancreas [168]. These systems typically have a control algorithm working in conjunction with a continuous glucose monitor (CGM) and an insulin pump, creating a closed feedback loop. These systems present a combination of physical components, which must communicate while accounting for the inconsistent nature of the human-in-the-loop.

Figure 1 shows the data exchange between devices in the APS. Both a CGM and insulin pump are attached to the user. The CGM records the user's blood glucose levels. CGM and insulin pump data, along with prior insulin pump activity are passed to the APS control algorithm in order to calculate the user's insulin requirements. This requirement is fed back to the insulin pump for insulin to be prescribed to the user. The data is then interpreted, stored and used to inform future insulin prescription.

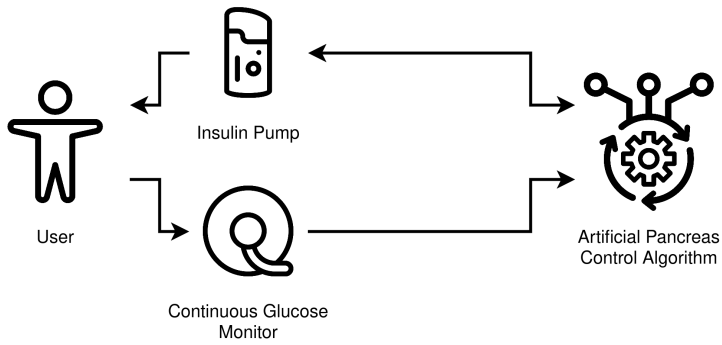


Figure 1.: The different systems and sensors used in an APS. This diagram shows how data flow between the user, a continuous glucose monitor, an insulin pump and the control algorithm. This creates a feedback loop mimicking a healthy pancreas.

The APS continuously adjusts insulin requirements based on the feedback loop between the user and the APS. The amount of insulin can, therefore, be configured based on its understanding of the user's blood glucose-insulin dynamics. As a result, the system is able to generate complex behaviours that adapt to the system's environment (the user).

The use of an APS allows for better glycaemic control for a person with T1DM [176]. For example, sleep tends to be a challenge for people with T1DM, either waking up continuously or sleeping through alarms [36]. APSs have been used to mimic background insulin values when sleeping, aiming to alleviate these issues. As a result, this has improved the quality of life for users through less disturbed sleep and reduced anxiety [100].

Ensuring such APS behaviours are correct is paramount to a user's safety. Incorrect configuration or system behaviour can lead to too much or too little insulin being administered. Incorrect behaviour, in this case, can lead to hyperglycaemia or hypoglycaemia, which, if severe, can have life-changing consequences such as strokes or heart disease, and potentially be life-threatening [81, 150].

2.2.1. Testing Difficulties

APSs are CPSs and, therefore, exhibit the same testing challenges. In this section, we revisit each of the testing challenges outlined in Section 2.1.1 and describe how they apply to an APS. We use these testing difficulties to motivate the remainder of this thesis.

Human Interaction

We have seen that an APS creates a feedback loop with the human body, allowing for the complex behaviours of an APS to develop and adapt to the user over time. As a result, the user of the APS must act as a human-in-the-loop in order to observe APS behaviour.

However, interacting with a human-in-the-loop to find scenarios that may cause system violations of an APS can be potentially dangerous to the user of the APS and time consuming. Scenarios that result in safety violations may endanger the user [81, 150] with blood glucose levels outside the safe glycaemic range [17]. Therefore, physical testing approaches

may not be feasible, especially when trying to find incorrect system behaviours.

Complex Input Spaces

The inputs to an APS are complex. APSs rely on blood glucose data, as well as insulin prescription and carbohydrate intake histories. Such values are continuous values or complex data structures with theoretically infinite combinations. Also, APS behaviours evolve over hours of execution, requiring long runtimes to fully observe their behaviour.

Finding the specific inputs that result in incorrect APS behaviour is, therefore, challenging. Physically trialling individual combinations of inputs requires physical clinical trials, which are time consuming and expensive [37, 166]. As a result, finding incorrect behaviour of an APS in this way is not viable.

Lack of Curated Data

Having a human-in-the-loop greatly reduces the availability of curated datasets for APSs. Datasets that span all APS behaviour would include dangerous scenarios and, therefore, may be unethical to generate [81, 150]. As a result, curating datasets that cover all APS behaviours may not be viable. From this, we see that APSs lack *controllability* (see Definition 3).

Pre-existing datasets of APS behaviour may be used in testing. For such datasets, there is no guarantee that all APS behaviours are covered and they may potentially give rise to spurious associations [27,118]. In Section 3.4 we discuss how the use of pre-existing datasets may affect the reliability of modern CPS testing techniques.

A lack of curated data is even more of an issue for a medical device such as an APS. Pre-existing medical data intrinsically may be missing values, have inconsistent formatting, or suffer from sensor error [52,94]. In such cases, performing software testing using this data becomes very challenging. In this thesis, we will see the difficulties of trying to perform software testing using a real, volunteered dataset from people with T1DM.

Unmeasurable Factors

APS behaviour is heavily dependent on the feedback loop between itself and the human body. Nuanced human body dynamics, such as sleep and exercise [142], affect blood glucose-insulin dynamics. A tester may be able to measure factors such as the user's blood glucose or carbohydrate consumption, but may struggle to measure more complex body dynamics. As a result, observed APS behaviour may appear inconsistent due to its dependence on such external, unmeasurable factors. An APS thus lacks *observability* (see Definition 4).

This lack of observability can make it difficult to determine whether behaviour is "correct". For example, a prescribed amount of insulin may be correct for one user but not another [88], or be correct for during the night but not during the day [142], due to unmeasurable external factors. As a result, defining whether system inputs will result in correct behaviour is non-trivial.

Defining correct behaviour for such complex system behaviours can result in an incorrect expectation of system behaviour. In Chapter 4, we attempt to alleviate the challenge of unmeasurable factors by using domain knowledge to evaluate system behaviours. Defining a domain knowledge, however, requires human comprehension of the domain. The extent to which the system tester understands the system and the domain in which it is used will be paramount for capturing this behaviour. Incorrect assumptions about system behaviours may be included in the test oracle, giving rise to the oracle problem [15], further discussed in Section 3.1.

2.3. Chapter Summary

In this motivating chapter, we discussed the testing difficulties associated with CPSs. We then presented a motivating example, an APS, and describe how these testing difficulties apply to this system.

In the following chapter, we expand on our motivating example, describing different levels of CPS autonomous behaviour and how an APS implements them. We then introduce an APS implementation to be used in our future evaluations. We also explore current CPS testing techniques, describe how they attempt to alleviate the testing challenges, and suggest how causal inference could help with the remaining challenges.

Chapter 3. Background

In our motivating chapter, we highlighted the testing difficulties associated with testing CPSs. We also introduced our motivating example: the APS. In this chapter, we expand on the behaviour of CPSs and APSs, as well as the software testing challenges that make such systems "untestable". We also introduce the subject system that will be used in the future evaluations of this thesis: oref0, an open-source APS.

In this chapter, we also cover the remainder of knowledge required for this thesis. We describe the state-of-the-art surrogate-assisted testing technique and discuss its potential challenges when testing CPSs. We discuss high-fidelity simulation through digital twins and explore how they have been used in testing and healthcare applications. Finally, we present causal inference-based testing and how it can use domain knowledge to alleviate the CPS testing challenges.

3.1. Software Testing

Software-controlled systems are inherently fallible [80]. Software testing provides a framework for validating software behaviour in order to ensure it works as expected. We use this section to outline what software testing is and some examples of its implementation.

Staats et al. [161] present a framework for defining the key concepts in software testing. We illustrate this framework in Figure 2. This framework consists of: the program, its specification, a set of tests, and the oracle. The specification represents the expected system behaviour in an "abstract, perfect notion of correctness" [161]. The specification is used to inform the development of the program itself, as the program should embody the implementation of the specification. The tests are also derived from the specification and are intended to highlight executions of the program in which the program behaviours deviate from the specification. The oracle, again informed by the specification, identifies such deviations and, in turn, determines whether the behaviours of the system are correct or not. This framework also describes the relationships between the program, oracle and test set, which we discuss through the remainder of this section.

Definition 5 (Test case) *An expression of a particular piece of program behaviour with a set of inputs and expected outcomes [80].*

System behaviour can be observed by executing the system with a test case and comparing the resulting behaviour to the oracle. The resulting behaviour can be validated as to whether the inputs result in the correct expected behaviour. Note how the test case is expressed in terms of program inputs and expected outcomes, *not* specifically outputs. Many testing approaches validate behaviour based on program outputs, such as unit testing [39] and

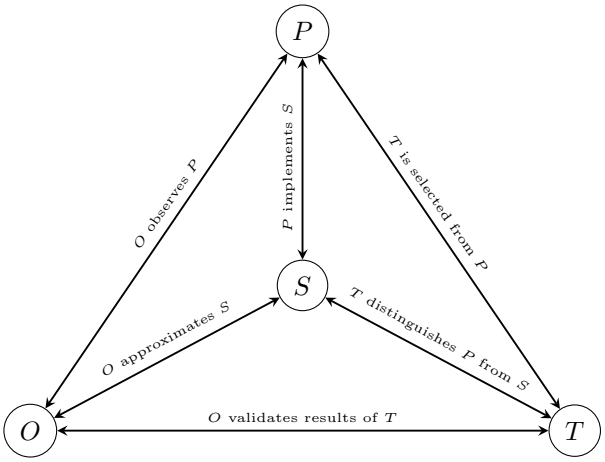


Figure 2.: The software testing framework outlining the relationships between the specification (S), program (P), oracle (O) and test set (T). Figure inspired by Staats et al. [161].

configuration testing [178], which validate whether program inputs or configurations produce the correct corresponding outputs. However, other approaches look beyond just the outputs. For example, metamorphic testing [151], further described in Section 3.1.1, validates program behaviour by verifying the expected relationship between two variables when one is changed.

Definition 6 (Test oracle) *A process of defining whether a test case being executed provides behaviour that is correct or not [15].*

Defining whether a test case results in correct behaviour is performed by the test oracle. However, defining such correctness (the oracle) from the specification can be non-trivial. Barr et al. [15] refer to this as the "oracle problem". We define this as follows:

Definition 7 (Oracle problem) *Defining an oracle for a program is not always feasible where formal specifications are not available, those specifications may be "imprecise", or the program's behaviour may not have "concrete" interpretation [15].*

To give an example of the oracle problem, we can look to our motivating example from Section 2.2. For an APS, specifications are available. However, due to the behaviour of the APS being heavily dependent on the user (lack of observability, Definition 4), such specifications are "imprecise", allowing for a large variety of behaviours to be correct for different users. For example, a specific prescription of insulin may be correct for one user but not another [88], even if they have the same blood glucose level and have consumed the same meals. As a result, defining a "precise" oracle of specific expected outputs for given inputs infeasible.

A system's behaviour may also not be "concrete". In Section 3.2.1, we introduce CPSs that can adapt their configuration to their environment. Such systems will have adaptive behaviours [11], resulting in behaviour that may change overtime. Therefore, a test oracle for this system would also need to be adaptive, changing what behaviours are defined as correct over time.

Definition 8 (Test case selection) *Approaches aimed at choosing a subset of test cases within a specific domain according to a criterion of interest [124].*

Another potential testing challenge is that of test set selection. This is represented in Figure 2 as the relationship between the program and the tests. It is not always trivial to select a test set that covers the program behaviours. Combinatorial testing techniques [33,92] aim to generate test cases in such a way to cover combinations of system inputs. For example, for an APS, this may require testing all combinations of different blood glucose levels with different carbohydrate consumptions.

However, for CPSs testing such a large number of combinations may not be feasible. CPS program behaviours may be complex and have continuous input spaces that are impractical to cover [49]. The test set cannot be too large or testing becomes computationally impractical [114], requiring the tester to search for tests that are most likely to reveal program faults [120].

Selecting test cases for an APS is non-trivial due to their large input spaces that result in complex behaviours (Section 2.2.1). Covering the potentially infinite combinations of carbohydrate consumptions, insulin prescription histories and environmental factors is infeasible. To test such behaviours, we require a different approach to find test cases that are most useful for uncovering incorrect system behaviour. In the following subsections, we describe potential solutions to test selection, search-based testing and configuration testing, in Sections 3.1.2 and 3.1.3. We also describe a potential solution to the oracle problem: metamorphic testing (Section 3.1.1).

3.1.1. Metamorphic Testing

Metamorphic testing aims to alleviate the oracle problem by testing metamorphic relationships instead of specific inputs and outputs [151]. A metamorphic relationship is the expected change in outputs when an input parameter is changed. A metamorphic test case can then be generated from a starting input value, its resulting program output value, a change to the input value, and the expected change to the output value [22].

Using our motivating example of an APS from Section 2.2, we expect different insulin prescriptions for different people [88,142]. However, regardless of the user we expect that when a user consumes carbohydrates, the insulin prescription from an APS should compensate for that and, in turn, increase the insulin prescription. The increase may be different for different users, but there should be an increase nevertheless. An expected increase presents a metamorphic relationship between the amount of carbohydrates consumed and the insulin prescription and can, therefore, be tested independently from precise inputs and output values. Metamorphic testing can, therefore, alleviate the oracle problem for systems where a "precise" oracle is not feasible.

3.1.2. Search-Based Testing

Search-based testing aims to find test cases that maximise a problem-specific fitness function within a practical time limit [115]. Such approaches use meta-heuristic search in order to

navigate potentially infinite input spaces to find relevant test cases. The resulting test set is, therefore, more likely to provide better system coverage or uncover more incorrect system behaviour than a manually generated one. In prior works, search-based testing has been used to enable different testing techniques, such as test prioritisation [104] and mutation testing [78].

Search-based testing could be used to alleviate the complex input space of an APS. Since meta-heuristic search approaches are able to navigate continuous input spaces [115], such a technique may be able to identify test cases that are more likely to uncover incorrect system behaviours. As a result, meta-heuristic approaches would reduce the computational expense of testing by finding combinations of carbohydrate consumptions, insulin prescription histories and environmental factors that are likely to result in incorrect APS behaviour.

However, meta-heuristic search requires a large number of executions. Executing an APS would require a considerable computational expense to account for their time evolutionary behaviour. As a result, meta-heuristic search may become infeasibly expensive as it attempts to search the behaviours of an APS. We, therefore, require a way of reducing the computational expense of executing the system when searching for incorrect system behaviours.

3.1.3. Configuration Testing

Configuration testing is another technique for addressing test case selection. Configuration testing [178] is not a test generation technique but a procedure for ensuring that new configuration values lead to correct system behaviour [23]. When a system configuration is changed, unit and integration tests should be executed in an isolated development environment with the new configuration before the new configuration is applied to the production system. Configurations that enable erroneous system behaviour can be identified before they are deployed, reducing the potential risks associated with system misconfiguration.

Configuration testing does not attempt to 'cover' the configuration space, as is already done by combinatorial testing techniques [33]. Instead, it tests system executions with selected configurations expected for system deployment [178]. In the context of an APS, we identify in Section 3.3.1 that the blood glucose target is a configuration that can result in dangerous scenarios if misconfigured. An APS can, therefore, be executed with expected configurations of the blood glucose target in order to evaluate the resulting behaviour.

However, applying configuration testing to a medical device, such as an APS, can be challenging. The human-in-the-loop is necessary for the device to function correctly. Unfortunately, this makes testing configurations that may lead to incorrect behaviour potentially dangerous to the user. We require a way of decoupling the human-in-the-loop from the system to safely test configurations of these systems and assess their behaviour.

3.2. Defining Cyber-Physical Systems

We use this section to outline the CPS concepts used in the remainder of this thesis. We first cover a taxonomy for different autonomous behaviour capabilities. We use this taxonomy

in the justification of our selected motivating example. We then introduce CPS testing objectives.

3.2.1. Autonomous Ability

To help our understanding of CPS, we use the 5C taxonomy proposed by Bagheri et al. [11] for defining and designing a CPS at different levels of autonomous ability. In particular, we highlight the **configuration** layer and use this to later motivate our subject system. We provide a brief summary of the 5C taxonomy with examples of how an APS implements each layer:

- **Connection** - CPSs allow for a transfer of data between the physical environment and virtual software elements. Sensors allow for the virtual aspects of the system to obtain physical data and use it as part of the control process. This level is seen in the majority of CPS.

  An APS transfers the physical blood glucose readings of a CGM from the physical domain to the software domain. This data can then be used by the remaining CPS layers.

- **Conversion** - The data gathered at the connection level can be converted into more useful, machine-understandable information for the system. Whether this is temperature data for health management or motor speed for later visualisation, interpretation of the physical data into something useful, such as an overall system health value, allows for a more informed awareness of the physical system.

  An APS converts CGM and other user inputs, such as carbohydrate intake and insulin histories. By interpreting the data, raw data, such as CGM readings, can be interpolated into JSON files containing the necessary information for later layers.

- **Cyber** - A centralised hub for all information gathered by the CPS is known as the cyber layer. This non-physical layer allows for calculation and identification of erroneous behaviour in the system as well as comparison to past iterations and states of the system.

  The APS acts as a hub for storing all interpreted data from the CGM, insulin pump and prior APS executions. This allows for past behaviour to be analysed by a user through logs, and enables that past behaviour to be used in the calculation of new behaviour.

- **Cognition** - For data about a system to be used in an intelligent and informed way, it must be understandable and provide support to users of the system, whether they be humans or other systems. This level restructures the data from the *Cyber* layer to enable other systems or users to be better informed of system behaviours. An example of this would be generating visualisations to better inform users of a system's behaviour.

The APS can predict and suggest insulin prescriptions for the user. This is achieved by making informed decisions based on the data stored in the *Cyber* layer. The effects of prescriptions can be displayed to the user, allowing them to make informed decisions about their blood glucose levels.

- **Configuration** - This level of the CPS allows for self-adaptation using the data acquired and synthesised in the previous levels. This ability to change behaviour allows for resilience and optimisation to the current physical environment.

  An APS is able to adapt its behaviour based on the predictions made at the *Cognition* layer. The effects of prior insulin prescription and carbohydrate consumption can be extrapolated, allowing the APS to configure future insulin prescription based on the predicted reaction of the user.

Systems start with the simplest layer, connection, and adopt further levels the more intelligent they are. Only very few CPSs achieve configuration. The configuration layer allows for adaptive behaviour, resulting in more complex behaviours dependent on the physical environment in which the system exists. Ensuring such complex behaviour is correct is paramount to the use of these systems, especially in human-interacting environments [49].

We observe how an APS implements all five CPS layers, exhibiting behaviours that adapt to its environment. In the case of the APS, this environment is the constantly changing blood glucose-insulin dynamics of the user. In Section 2.2, we described the importance of ensuring that such complex behaviours do not lead to dangerous scenarios for the user [81, 150].

3.2.2. Testing Objectives

In order to test the complex behaviours of a CPS, such as an APS, it is important to define which aspect of the behaviour is being tested. Zhou et al. [184] present four different testing objectives for cyber-physical systems: conformance, robustness, security and fragility. We define these objectives with examples of how they may be applied to an APS:

- **Conformance Testing** - Testing that a CPS conforms to the behaviour of some oracle. Conformance testing can be implemented using a model representing the specification as the test oracle and creating tests to ensure that the physical system conforms to the expected behaviour [9]. Conformance testing of an APS could involve ensuring its behaviour results in the blood glucose level being kept safe.

- **Robustness Testing** - Investigating whether a CPS reacting to its stochastic environment will be classified as erroneous behaviour [2]. This is particularly important in CPSs as their physical environment will change over time, and this should not cause test failures. For an APS, such an environmental factor could be the user exercising. Robustness testing would test whether the user exercising affects if the APS is able to keep the user safe.

- **Security Testing** - Ensuring that a system cannot be affected by cyber-attacks that could either change the behaviour of the system or allow for system information to be

intercepted. Due to the connection level of CPSs requiring networks, security testing is important to ensure they are safe and secure [26], especially when conducting safety-critical tasks. An APS's behaviour is paramount to its user's health and, therefore, security testing would ensure the system is safe from cyber-attacks.

- **Fragility Testing** - Ensuring that minor changes to inputs or the environment do not cause the system to completely stop working. Environmental uncertainties can cause minor changes in sensor readings due to being physical systems and this should not drastically affect system behaviour [184]. For an APS, fragility testing could test that small changes to environmental factors, such as consuming slightly more carbohydrates, does not drastically change the system's behaviour.

In Section 2.1.1, we highlighted the difficulties of testing whether a CPS's behaviour *conforms* to an expected behaviour. Prior work (P1) [160] has found that the conformance testing objective is widely used in CPS testing. We, therefore, we focus the conformance objective when testing an APS in this thesis.

The other testing objectives are important for ensuring correct CPS behaviour, but we designate these as out of scope of this work. However, future work should be conducted to cover these objectives. For example, an APS's behaviour is paramount to its user's health and, therefore, ensuring the system is *secure* makes it good candidate for security testing. It should also be noted that our upcoming approach in Chapter 4 can also be applied to robustness and fragility testing.

3.3. OpenAPS

In Section 2.2, we described the difficulties of testing an APS. In this section, we build on this by exploring a widely used APS [41], OpenAPS.

Typically, an APS is validated through clinical trials [166] in order to ensure that system behaviour cannot harm the user. Such trials, therefore, are used to test for erroneous behaviour and reduce potential risks to the user. Clinical trials, however, can be expensive and, in the case of APSs, have been seen to delay the deployment of technological advancements in the field [100].

Due to the delay in technological advancements, open-source "do-it-yourself" (DIY) APS alternatives have been developed [21].[1] Open-source APSs require the user to download, compile and install the control algorithm themselves, typically without the oversight of a medical professional. DIY approaches, therefore, present a risk as the systems have not undergone clinical trials and could potentially have unsafe behaviours.

oref0 [98] is the open-source reference APS reference control algorithm used by APS implementations [100]. This algorithm has been implemented across multiple different APS applications [4,95]. In this thesis, we do not elaborate on alternative algorithms as oref0 is the

[1] Anecdotally, during my research I found that many people I spoke with who had T1DM trusted their own intuition from their experience dealing with T1DM over medically approved APSs. There could be a potential link between this and people using open source APSs and I believe this would make for an interesting investigation.

only open-source control algorithm. oref0 is typically implemented as a mobile application that can interact with most existing CGMs and insulin pumps. A user must compile and install the application themselves from source code and configure it to interact with their existing T1DM hardware.

With an increase in the adoption of oref0 and similar systems [69, 100], it is vital to have techniques that identify scenarios in which systems allow a person to become unsafe. Testing the behaviour of oref0 would reduce the risk to the user by allowing for the examination of incorrect behaviour. However, as identified in Section 2.2.1, testing such a safety-critical, human-interfacing system is non-trivial.

3.3.1. Configuration

oref0 presents a highly configurable environment, where users require personalised configurations. The oref0 documentation presents 13 commonly changed parameters, from 47 parameters in total. The documentation also presents how a single person requires multiple different configuration profiles for different times of day and activities [131]. For example, exercise requires a specific configuration profile that has 6 additional exercise specific parameters.

The *blood glucose target* of oref0 is an example of such a configuration that defines the value that the APS attempts to keep the user's blood glucose at. The target may vary with different activities, such as sleep and exercise, as well as different people. Testing different configurations of the *blood glucose target* may produce undesirable and potentially dangerous behaviours if misconfigured. Performing this testing with a human-in-the-loop would enact these behaviours on the user.

The documentation for the iOS implementation of OpenAPS, Loop [127], contains the following quote: "You can count on your fingers the number of doctors in the US who are capable of properly adjusting settings for Loop. You can probably count on your fingers and toes the number worldwide who can successfully help you with Loop settings." This is an indicator of how much prior knowledge is required to correctly configure these systems. As stated in Section 2.2, if an APS is misconfigured, it can have severe consequences [81, 150].

OpenAPS has a thriving community across Github issues [126], Gitter [129], Facebook groups [125] and a Google group [128]. At the time of viewing, one of the main forums for APS configuration with over 30,000 users [125] contained multiple daily posts where users were asking for advice with configuration, struggling with code compilation and having difficulties integrating with other applications and hardware. From both the documentation and online communities, we identified that the configurability of oref0 is a challenge.

These configuration difficulties are not only faced by DIY implementations of APSs but are more widely faced by medical devices [106]. Closed-source APS devices have recently posed a challenge for system configuration. A recent example presented a clinically approved device parsing configuration inputs incorrectly, ignoring decimal points [109]. In some cases, this caused a potentially dangerous amounts of insulin to be administered.

3.3.2. Subject System

Due to the large number of users, configurability and open-source nature of oref0, it makes a suitable subject system for evaluations throughout the remainder of this thesis. In Chapter 6, we describe the development of a simulation environment for oref0 to enable surrogate-assisted testing (Section 3.4). In Chapter 7, we describe the evaluation of our upcoming testing technique against oref0. We investigate whether our upcoming technique (Chapter 4) is able to alleviate the lack of *controllability* and *observability* of an APS, as described in Section 2.2.1.

3.4. Simulation and Surrogate-Assisted Testing

Finding scenarios in which CPSs do not *conform* to their expected behaviour, as described in Section 3.1, allows for incorrect behaviour to be identified. Physical constraints reduce the ability to execute physical tests and the complexity of system behaviours makes finding specific inputs that lead to incorrect behaviour very challenging [49]. We use this section to introduce simulation-based testing, how surrogate models are used to improve the efficiency of searching for incorrect behaviour and the potential weaknesses of machine learning-based approaches when using pre-existing datasets.

"In-the-loop testing" [107] aims to alleviate the physical constraints of CPS testing. This testing approach is used to test a specific component of a system, while simulating the remainder of the system and its environment. Modelling the rest of the system allows for more specialised testing of a specific component without the physical constraints of testing the entire system [49].

This is especially useful for CPSs that include the configuration layer (Section 3.2.1), such as an APS (Section 2.2), since the environment can be modified for each test. Environmental factors, which cannot be controlled during physical testing, can, therefore, be accounted for without physical constraints.

For example, an APS should adapt the insulin it prescribes based on the blood glucose level of the user. In-the-loop testing would allow for the APS control algorithm to be tested while the interacting environment (the human user) is simulated. As a result, blood glucose levels that are unsafe for physical testing can be trialled without risk to the user.

However, due to the high fidelity nature of CPS simulation, in-the-loop testing can be computationally expensive. For example, Menghi et al. [116] tested using simulations of a satellite that each took $\sim$1.5 hours to execute. As a result, for such systems using simulation alone may not be feasible when searching for inputs that result in incorrect system behaviour.

In the remainder of this section, we outline a state-of-the-art technique that builds upon in-the-loop testing: surrogate-assisted CPS testing [120]. Figure 3 illustrates the approach. This testing technique aims to reduce the computational expense of in-the-loop testing by searching for system violations using a computationally efficient surrogate model as the fitness function of search-based testing techniques (Section 3.1.2). We describe each stage of the procedure and provide a running example describing how a tester may apply these

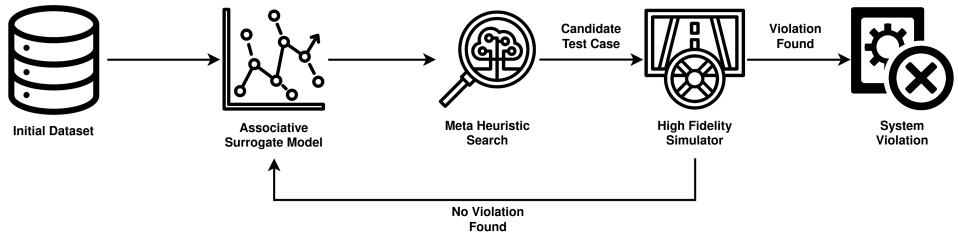


Figure 3.: Surrogate-assisted CPS testing technique. An initial dataset is used to generate a surrogate model. The surrogate model is used in meta-heuristic search to find scenarios that likely cause system violations. Such scenarios are validated on a high-fidelity simulator.

steps while testing an APS control algorithm.

3.4.1. Generate Surrogate Models

Surrogate-assisted CPS testing requires a surrogate model that is a representation of the CPS's behaviour that is less computationally expensive than a high-fidelity simulator. Surrogate models normally provide a simplification of the CPS behaviour, for example, by representing only the relationships between potential system violations and the system's inputs [16].

To achieve this, these approaches typically use machine learning models, such as polynomial regressors, gaussian process regressors and radial bias function networks [38, 64, 163]. These models are trained on historical data of the system in order to represent system execution. As a result, the system can be represented in a way that is more computationally efficient than running the system, whilst approximating its behaviour.

Following our running example, the behaviour of the APS control algorithm can be implemented as a surrogate model. Following prior works [70], this model could have the inputs of the APS and return the likelihood that those inputs would result in a system violation. In this case, such a violation could be severe hyperglycaemia or severe hypoglycaemia. These likelihoods could then be evaluated by the testing procedure as a less computationally expensive interpretation of an APS's behaviour.

The Ensemble Model

In prior work, a single machine learning model has not been seen as sufficient represent the behaviours of systems [70]. Goel et al. [64] describe the use of an *Ensemble* surrogate model, which trains multiple individual machine learning models and combines their outputs based on a learned weighting. The model presented by Goel et al. [64] consists of a polynomial regressor, a gaussian process regressor and a radial bias function network [38, 163]. We use this Ensemble as the state-of-the-art baseline in our upcoming evaluations in Chapters 5 and 7.

By using multiple different models, models that may struggle to interpret data or represent behaviour can be supplemented by other models. For example, if the radial bias function network was inaccurate in representing APS behaviour, and a polynomial regressor more

appropriate, the polynomial regressor could be given more weight when determining the ensemble's output.

To determine how the individual models should be weighted, each model is evaluated to determine its accuracy with regards to the training data. The dataset is split into a training set and a testing set [165]. The individual models are then trained on the training set and scored based on their ability to accurately replicate the testing set. The scores derived from this process outline a weighting for each model with respect to the training data.

Haq et al. [70] performed surrogate-assisted testing on a automated driving system (ADS) using this ensemble surrogate model. For their evaluation, they used a fitness function that maximised for ADS violations based on scenario inputs. We use this same fitness function for the baseline of our evaluation in Chapter 5.

Spurious Associations

It should be noted that the machine learning models typically used in this step are trained from associations in the training data [64]. However, due to the difficulty of obtaining curated datasets for CPSs (lack of controllability, Definition 3), the training data may result in *spurious* associations. Spurious associations are associations that are not causal but exist in the data as a product of the data collection process, such as not covering all system behaviours [58]. By training on such associations, the resulting machine learning model may learn these spurious associations as causal, potentially representing the system behaviour incorrectly [27].

As with our APS example, pre-existing data could include scenarios in which the user only enters severe hypoglycaemia when eating a specific meal. A surrogate model may learn this association between spurious associations as causal [58, 136], resulting in that specific meal being associated with severe hypoglycaemia. The resulting surrogate model would, therefore, not be an accurate representation of actual APS behaviour. We explore how this relationship may not be causal in Section 3.6.3.

3.4.2. Search for System Violations

Given a simplified representation of a CPS (the surrogate model), meta-heuristic approaches are used to search for potential violations. This step implements search-based testing, as described in Section 3.1.2.

A fitness function that maximises the likelihood of a violation drives the search process towards potential incorrect system behaviours. Examples of such fitness functions include estimating and maximising the input-violation relationships for scenarios [16] or searching for the input-output timesteps of a system that result in a system violation [116]. The test case identified as that most likely to result in a system violation is denoted as the *candidate test case* (as seen in Figure 3).

For example, Haq et al. [70] investigated 6 potential violations: being too far from the centre of the road, not avoiding other vehicles, not avoiding pedestrians, not avoiding other obstacles, not abiding by traffic laws, and not reaching the scenario destination. In this

case, the surrogate model estimated the likelihood of these violations from Pylot inputs. A meta-heuristic search could then find scenario inputs that maximised for violations.

3.4.3. Simulation-based Evaluation

To evaluate whether a violation found by the meta-heuristic search translates to a real system violation, it is validated under high-fidelity simulation. The real system is executed in a simulated environment for the suggested inputs, evaluating the resulting behaviour against a test oracle to determine whether a violation is observed [107]. This process mirrors that of configuration testing (Section 3.1.3) and is further discussed in Section 3.4.5. This technique provides a way of finding incorrect behaviour of CPSs without the physical risks associated with them [49].

For an APS control algorithm, the remainder of the human-in-the-loop could be simulated. The simulation would require a high-fidelity in order to capture nuanced behaviour of blood glucose-insulin dynamics. Such a simulation would be able to complete the feedback loop between the APS control algorithm and the human body, allowing APS behaviours to be observed. The realism of the simulation, however, would make it computationally expensive, hence the requirement to only execute scenarios with a high likelihood of resulting in a violation.

3.4.4. Data Feedback

Given a scenario in which the candidate test case, identified in Section 3.4.2, does not result in a system violation, the surrogate model is re-trained based on this additional information. Recent works [70, 116] have achieved this by simply adding the new data into the original dataset and re-training the surrogate models with the new expanded dataset. The original authors viewed this approach as a way for the surrogate to learn about the environment it represents as it explores incrementally.

3.4.5. Parallels to Configuration Testing

The simulation-based evaluation step (Section 3.4.3) of surrogate-assisted CPS testing executes specific system configurations in order to evaluate the resulting system behaviour. This technique parallels configuration testing, described in Section 3.1.3. Both approaches do not aim to cover the system's input spaces, but instead execute specific system configurations. Incorrect system behaviour can be identified from this evaluation, highlighting system configurations that result in behaviour violations. We use this parallel to configuration testing when developing a simulation environment for our motivating example in Chapter 6.

3.4.6. Summary

Surrogate-assisted CPS testing uses surrogate models to find test cases that are likely to result is system violations. Those test cases can be evaluated in a high-fidelity simulator to

observe if the test case results in an actual violation. As a result, CPSs can be tested in a computationally effective manner.

This approach aims to solve the testing difficulties of human-interactive environments and complex input spaces (presented in Section 2.1.1). Simulation-based evaluation reduces the set up expense of physical testing by simulating the environment in which tests occur. This is particularly useful for human-interactive environments where the user may be put in dangerous scenarios. Complex input spaces can also be more efficiently explored using meta-heuristic search. By using a surrogate model that is less computationally expensive than the high-fidelity simulator, meta-heuristic search can be used to efficiently identify scenarios that may result in system violations.

However, surrogate-assisted CPS testing does not account for a lack of *controllability*. Such a technique is reliant on curated data to generate surrogate models that do not learn spurious associations [58, 147]. Spurious associations would result in less accurate representations of the system and, therefore, reduce the reliability of the scenarios found from them [27].

Surrogate-assisted CPS testing also relies on the ability to define whether system inputs will result in violations. However, some systems implement the *configuration* CPS layer (Section 3.2.1), resulting in behaviours that adapt to their environment through potentially unmeasurable factors (lack of *observability*) [49]. As a result, system behaviour may appear inconsistent, making defining which behaviours are "correct" difficult.

3.5. Digital Twins

In order to perform surrogate-assisted testing, we require a high-fidelity simulation environment. A modern approach to defining such a simulation environment, specifically for CPSs, is that of a *digital twin*.

A digital twin is a simulation that runs in parallel to a physical entity. Digital twins change and adapt their behaviour to match that of their physical counterpart [101]. As a result, they present a virtual replica of the system on which the behaviour from interventions can be predicted. Insights can be gained from the simulation to enhance the behaviour of the physical twin in the real world [46, 51]. Examples of this enhancement include real-time visualisation [185] and physical degradation prediction [139].

It should be noted that digital twins have been defined to adapt system behaviour in a "time-evolutionary" [47] or "live" [51] manner. This can, but does not always, equate to real-time. A digital twin should adapt alongside its physical counterpart, with that time-evolutionary adaptation being dependent on the system's domain.

For example, a digital twin used in cyber-attack detection [181] would require real-time adaptation to quickly identify system anomalies. However, a digital twin used to identify the degradation of a wind turbine [6] may update in a less timely manner due to the time-scale of such degradation.

Ensuring confidence in digital twin model predictions is paramount to their trustworthiness. By using an explainable model, it allows for a more informed confidence in digital twin predictions regarding the physical system. Domain experts are able to fully understand the

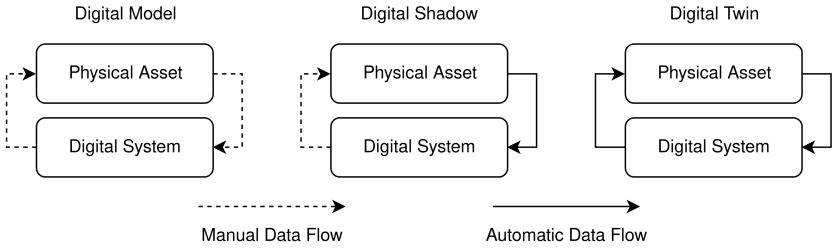


Figure 4.: Digital model, digital shadow and digital twin

causes of potentially faulty behaviour by tracing back through the model's execution. Digital twins have achieved this through physics-driven and explainable AI approaches [51,149]. Prior work, however, has found that these predictive capabilities are not widely used, especially when validating system behaviour [160].

3.5.1. Classifying Digital Twins

Classifying different types of digital twins is important for understanding their capabilities and the context in which they should be used. Due to digital twin definitions being so diffuse [56, 160], there is currently no consensus on how to classify digital twins. This has led to multiple different ways of doing so, with overlap in these taxonomies. To provide an overview of the different taxonomies used to classify digital twins, we explore two different classification methods in this section.

Eyre et al. [51] propose three classifications for digital twins based on their most complex functional output. We revisit this classification when assessing the requirements for a digital twin in Chapter 6. The classifications are outlined below:

- **Supervisory** - Supervisory digital twins accept data from the physical counterpart to provide information to a human observer, allowing them to act on this information. The functional output for this classification of digital twin is simply visual information.
- **Interactive** - Interactive digital twins close the feedback loop by using data gathered from the physical counterpart to make changes to the system in a time-evolutionary manner based on its current and historical states. Interactive digital twins have a functional output that can affect the physical counterpart.
- **Predictive** - Predictive digital twins predict the future states of systems by using information gathered in real-time. They can also infer additional information for unseen contexts through physics based or data driven inference. The digital twin can use these predictions to change the physical counterpart's behaviour or issue preemptive warnings to human operators.

Douthwaite et al. [46] propose a digital twinning framework with three different classifications for interacting with its physical counterpart. These are similar to other definitions found elsewhere in the literature [56, 90]. As with the prior taxonomy, we also revisit this

taxonomy in Section 6. Figure 4 presents the data flow between the physical and digital systems for each of them.

- **Digital model** - Modelling provides a system with no automatic data exchange between the simulation and the physical counterpart so the simulation can run independently. This classification is outside our definition of a digital twin but importantly highlights how model-based approaches are used for CPSs.
- **Digital shadow** - Digital shadowing allows a simulation to mimic a physical counterpart where data is passed only from the physical counterpart to the digital twin in real-time. This can provide a "visual representation" [51] of the system. This classification provides behaviour equivalent to the supervisory classification above.
- **Digital twin** - The final classification is that of the digital twin itself where both the physical counterpart and digital twin exchange data to allow for real-time analysis, interaction and adaptive behaviour. This classification closes the feedback loop allowing for interactive digital twins with the ability to provide predictive capabilities where necessary.

Both classification systems outlined above have overlap but present different perspectives on the digital twin. The taxonomy set out by Eyre et al. [51] focuses primarily on the functional output of the digital twin, whereas Douthwaite et al. [46] explores the way in which information is shared within the system.

3.5.2. Digital Twin Oracles

For surrogate-assisted CPS testing (see Section 3.4) to be used, a simulation environment is required. Digital twins allow for an entire system to be modelled "live" [51] as a coupled entity to allow for system verification throughout the system's lifetime. By adapting to the system over its lifetime, a digital twin can provide a more rigorous and ongoing testing environment. The digital twin can be used to test the consequences of such adaptations throughout the system's life-cycle. The user can also achieve testing of the complete system as well as connections between CPSs.

Due to the constant adaption, a digital twin's behaviour should be indistinguishable from its physical counterpart. Grieves [67] suggests the "Grieves Performance Test" where a human is asked to decipher the difference between the output of a physical counterpart and that of a simulation. A similar test is presented by Worden et al. [175] where a Turing Mirror is used to provide test cases to either a physical counterpart or a simulation. Both suggested tests are iterations on the Turing Test [171], in which a human is asked to distinguish between another human and a computer.

These tests are intended to provide insight into the success of simulating physical counterparts based on their accuracy. Grieves goes as far to say that simulations can be more useful in testing due to the physical limitations of testing environments. These tests show that comparison to a representation of a system is not a new concept, but has only recently been introduced to CPSs [72].

3.5.3. Healthcare Applications

In order to support our motivating example from Section 2.2, we use this section to describe how digital twins have been used to enable to testing of healthcare systems. Digital twins have been proposed to provide personalised medicine by enabling added safety through simulation and improved explainability of treatments [20, 35]. Computational modelling [73, 122] and digital twins [37, 85] are an emerging technology in healthcare as a part of Healthcare 4.0 [5, 170].

Digital twins present an ability to adapt and trace clinical interventions [82] as well as decoupling the human-in-the-loop through simulation [50]. The system itself can be evaluated across different configurations without putting the user in potentially dangerous scenarios. This is particularly important for such a configurable system as an APS (see Section 3.3.1).

Corral-Acero et al. [37] propose an approach for the optimisation of clinical devices through the use of a personalised explainable model. This model aims to capitalise on the predictive power of digital twins, allowing for responses to clinical intervention to be predicted without putting the user in potentially dangerous scenarios. This approach present a theoretical framework for optimising clinical devices based off digital twin predictions. We summarise the approach outlined by Corral-Acero et al. as follows:

1. Obtain clinical data, medical images or other context specific information.
2. Generate a mechanistic or statistical model to replicate the mechanics of the body dynamics being modelled.
3. Calibrate and optimise this model based on the users data.
4. Validate the model's accuracy.
5. Present human responses to clinical interventions based on model predictions

Such steps enables the configuration testing [178] of medical devices. As described in Section 3.1.3, configuration testing executes specific configurations of a system to validate its behaviour. This execution parallels step 5 of the approach proposed by Corral-Acero et al. [37]. The medical device can be executed as part of a digital twin in order to observe its behaviour for a specific environment or person (configuration). We use this notion in Chapter 6 when developing a simulation environment for oref0 (Section 3.3) to enable surrogate-assisted testing (Section 3.4).

3.6. Causal Inference-based Testing

We recall from Section 3.4.6 that surrogate-assisted CPS testing struggles with systems that have a lack of *controllability* and *observability*. In this section, we explore causal inference-based testing and how it has been used to alleviate such challenges in similar domains.

Causal inference is a family of statistical methods that combines domain knowledge with statistical techniques, traditionally used in epidemiology [71, 84]. These techniques are used

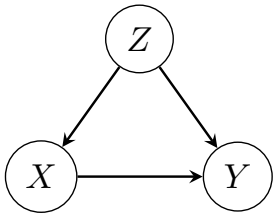


Figure 5.: An example of a causal DAG

in the analysis of clinical trials where pre-existing data is used and unmeasured external factors are expected [135]. Causal inference uses domain knowledge to account for these difficulties and improve the accuracy of statistical estimations.

In this section, we highlight a link between the lack of *observability* and *controllability* in a CPS [54], outlined in Section 2.1.1, and how causal inference alleviates similar issues in epidemiology. We first discussed how previous work has used domain knowledge in software testing in the form of a causal *directed acyclic graph* (DAG) [132]. We then explore how the encoding of domain knowledge in a causal DAG can help mitigate the challenges associated with CPS testing.

3.6.1. Incorporating Domain Knowledge

In prior works [27, 159], causal DAGs have been used to represent the domain knowledge of a software system. We define a causal DAG (V, E) as follows:

Definition 9 (Causal DAG) *A set of vertices (V), which represent the visible variables of a system, and a set of edges (E), which represent the expected causal relationships between variables, as defined by domain expertise.*

Figure 5 illustrates an example of a causal DAG of 3 variables (X, Y and Z). Here, we expect X to causally affect Y, and both to be causally affected by Z. A domain expert may provide domain knowledge about the causal relationships, defining how such variables should affect one another. An example of such a causal relationships could be "by increasing X, this should cause in an increase in Y." By incorporating an expectation of program behaviour (a specification) into the causal DAG, we present a parallel between the DAG and the test oracle (Section 3.1).

In this work, we only consider situations where all of the variables in the DAG are visible to the tester. We refer to *visible* variables as those in pre-existing datasets. This is different to *observability* (see Definition 4), which described whether system behaviour appears inconsistent due to unmeasurable external factors. For example, *Carbohydrates* and *Insulin Prescription* may exist in a pre-existing dataset for an APS and are, therefore, visible. Body dynamics may make the interaction between these variables appear inconsistent and, therefore, the relationship lacks *observability*. Causal inference does provide additional tools for representing variables that are not visible in the DAG (instrumental variables [112], robustness calculations [25]), but we limit this thesis to only visible variables.

In the following sections, we explore how a causal DAG of a system presents a potential solution for representing complex system behaviours and the identification and adjustment of biases in pre-existing data.

3.6.2. Evaluating Behaviours which lack Observability

By encoding the causal domain knowledge about a software system, a causal DAG allows for domain-specific properties to be validated. A tester can evaluate the existence of expected causal relationships between variables, ensuring these relationships hold within actual system executions. Since these properties may lack observability [54], such as an APS's interaction with the human-in-the-loop (see Section 2.2.1), two executions of the same *visible* inputs may not result in the same output. Therefore, traditional techniques described in Section 3.4, which rely on specific inputs to find system violations, may struggle.

A tester can instead use causal inference to answer causal questions about system behaviours. For example, we can assume that an APS should adapt the amount of insulin prescribed based on the amount of carbohydrates consumed by the user, increasing insulin as the carbohydrates increase. The exact amount of insulin prescribed by the APS, however, may differ for a given amount of carbohydrates due to the body's unmeasurable dynamics. The relationship between carbohydrates and insulin prescription, therefore, lacks *observability* (see Definition 4) since values of carbohydrate consumption cannot be mapped to specific insulin prescription values. However, we can assume that the trend of increasing insulin prescription as carbohydrate consumption increases should still hold.

We can encode the above behaviour in a causal DAG as follows: the fact that a change in the variable *Carbohydrates* should cause a change in the variable *Insulin Prescription* can be encoded as *Carbohydrates* $\rightarrow$ *Insulin Prescription*. A domain expert can then express the expected behaviours of such relationships, such as changing *Carbohydrates* should cause a change, in this case an increase, in *Insulin Prescription*. The causal DAG allows a tester to identify relationships for which they can answer causal questions about a system, such as "What if a user consumes more carbohydrates? Would that cause the expected increase insulin prescription?"

To inspect the relationships between variables, a tester can generate a *causal model* based on the causal relationships identified from the DAG. This model contains only the variables that relate to the causal relationship being modelled. This model approximates the behaviour of how changing one variable causes another to change in the form of a causal relationship.

Prior work has used regression models trained on pre-existing datasets to accomplish this [27]. As a result, a tester can use this model to answer "what if" questions by estimating how changing one variable may affect another. Causal model estimations can, therefore, test whether expected relationships hold for prior executions of the system. This is similar to metamorphic testing, with the causal model removing the requirement for multiple system executions. We further discuss this similarity to metamorphic testing in Section 3.6.4.

A causal model can be used to calculate an average treatment effect (ATE) to quantify

causal relationships in a system [27,159]. An ATE, as presented in Equation (3.1), estimates the effect of a treatment (t) by estimating population outcomes with and without a given treatment. From our previous example, a treatment could be observing the effect of changing a user's *Carbohydrates* intake from their typical consumption ($t = 0$) to a specific diet ($t = 1$). Assuming a diet which increases *Carbohydrates* (t) should cause an increase in *Insulin Prescription* (y), we would expect a positive ATE for this causal relationship.

ATEs that contradict the expected causal relationships between variables suggest that a system may be behaving incorrectly [27]. As a result, the tester can evaluate a system's, or a surrogate model's [159], input-output relationships compared to the expected relationships encoded in the DAG. In this thesis, we use expectation notation ($\mathbb{E}[Y|X]$) based on that defined in Neal's Introduction to Causal Inference [119]. We use this notation to express how a statistical model, such as a causal model or regression model, estimates its output (Y) from a set of inputs (X). In the case of CPSs, this relates to an estimation of system outputs (Y) with respect to system inputs (X).

$$ATE = \mathbb{E}[y|t = 1] - \mathbb{E}[y|t = 0] \tag{3.1}$$

Causal questions present a potential way of alleviating the testing challenge of the lack of *observability* of APS behaviour (see Section 2.2.1). A tester can answer questions about the relationships between variables in order to verify whether expected behaviours hold. This can account for inputs that may not result in a consistent output, due to unmeasurable factors, as the relationship between the variables is tested instead of specific input values.

For example, an APS user may consumes more carbohydrates on a specific diet ($t = 1$) than their typical consumption ($t = 0$). The specific values of an APS control algorithm's insulin prescription ($\mathbb{E}[y|t = 1]$ and $\mathbb{E}[y|t = 0]$) may not always be consistent due to unmeasurable external factors. However, we expect the APS control algorithm to increase its insulin prescription, resulting in the ATE always being positive.

Observing the effect of changing a system input is different to the state-of-the-art surrogate model, which maximises for violations based on specific inputs, as outlined in Section 3.4.1. In Section 4.2, we use this concept to aid in defining the correct and incorrect behaviour of an APS.

3.6.3. Reducing Bias in Datasets that lack Controllability

A causal DAG can be used to identify variables that may give rise to spurious associations [135], as described in Section 3.4.1, through a technique called *causal identification* [134]. This allows a tester to first identify and then adjust for bias-inducing variables in a surrogate model through adjusted estimation. Performing causal identification on the causal DAG in Figure 5 would uncover that the variable Z may introduce spurious associations when estimating $X \rightarrow Y$, due to Z causally affecting both X and Y [14].

As explored in Section 3.4.1, pre-existing data may introduce spurious associations when training surrogate models. This is due to CPSs having a lack of *controllability* (see Definition 3). As a result, the surrogate model may learn these incorrect associations, leading

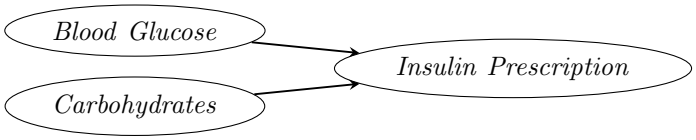


Figure 6.: An simplified example of a causal DAG for an APS.

to an incorrect representation of system behaviours. Inputs that are less likely to cause system violations may, therefore, be suggested, resulting in more use of the computationally expensive simulator.

For example, an APS may decrease *Insulin Prescription* to ensure higher *Blood Glucose Levels* during exercise, independent of the consumption of *Carbohydrates* [142]. We illustrate this example in Figure 6. The data collection only including exercise could result in a spurious association between *Carbohydrates* and a decrease in *Insulin Prescription* being inferred. As a result, a surrogate model may be trained with an incorrect relationship of *Carbohydrates* $\rightarrow$ *Insulin Prescription* even though the actual cause of the *Insulin Prescription* was the higher *Blood Glucose Levels* required for exercise.

Causal identification enables estimations to be adjusted to reduce the impact of spurious associations [134]. Bias inducing variables can be identified from the causal DAG. These variables can then be accounted for through adjusted estimation in the causal model [27]. Equation (3.2) presents adjusted estimation using a surrogate model to estimate an output of a system (y) while including identified potentially bias-inducing variables (B) at constant values (X) as model covariates.

$$\textit{Adjusted Estimation} = \mathbb{E}[y|B = X] \tag{3.2}$$

In our previous example, causal identification could be used to highlight *Blood Glucose Levels* as a potentially bias inducing variable (B). This could then be held constant in the causal model in order to estimate the adjusted causal relationship between *Carbohydrates* and *Insulin Prescription*. As a result, the causal model can estimate a more accurate representation of this causal relationship.

In Section 4.2, we use adjusted estimation to alleviate the lack of data *controllability* for APSs (see Section 2.2.1). By using causal identification, pre-existing datasets can be used while reducing the impact of potential spurious associations.

3.6.4. Parallels to Metamorphic Testing

In prior work [27], asking causal questions in terms of calculating ATEs has been equated to metamorphic testing of systems [151]. Metamorphic testing aims to alleviate the "oracle problem" [15], which arises when defining correct behaviour for specific inputs is difficult (Section 2.1.1). Instead of evaluating the correctness of individual system executions, metamorphic testing checks certain expected relationships (metamorphic relationships) across system executions [22]. For example: "a user expects the insulin prescription of an APS to increase if they have consumed more carbohydrates." We observe, therefore, a similar-

ity between causal relationships, as encoded by domain knowledge in the causal DAG, and metamorphic relationships.

Traditional metamorphic testing, as described in Section 3.1.1, executes the system across multiple inputs and checking that the metamorphic relations hold [22]. However, metamorphic testing would require multiple executions of the system, which, as described in Section 2.1.1, is not always feasible.

By using causal inference, a causal model can be used to estimate system behaviours while accounting for potential spurious associations [147], as described in Section 3.6.3. This is ideal for an APS, as interacting with the system directly is potentially dangerous for the user, due to the human-in-the-loop, and computationally expensive in simulation.

3.7. Chapter Summary

In this Section, we expanded on the context required to tackle the testing challenges outlined in Section 2.1.1. We described how the difficulties of human interaction and complex input spaces affect CPSs and our motivating example, the APS. Surrogate-assisted testing presents a potential solution to these testing challenges, allowing for simulation environments, such as digital twins, to decouple human interaction and surrogate-based search to reduce the complexity of finding system violations. However, these approaches struggle to overcome the lack of controllability and observability found in CPSs.

We outline how causal inference-based testing has been used to alleviate a lack of controllability and observability in prior works. Causal testing, therefore, highlights the potential solution of incorporating domain knowledge to alleviate the remaining challenges for surrogate-assisted testing. In the following chapter (Chapter 4), we use causal inference-based testing to propose a *causal surrogate model* to be used with existing surrogate-assisted testing techniques.

In this chapter, we also described the APS oref0. Due to it being open-source and having data availability, we use oref0 as the subject system of our evaluation in Chapter 7. From this evaluation, we aim to evaluate the efficacy of our upcoming surrogate model using a real, widely used APS.

Chapter 4.
Causally-Assisted CPS Testing

In Section 3.4, we described how surrogate-assisted CPS testing approaches aim to find scenarios that cause system violations. However, we also highlighted how pre-existing data and unmeasurable factors may affect the accuracy of surrogate model estimations. These factors may result in a decrease in the effectiveness of surrogate-assisted testing, requiring a way of mitigating these challenges.

Physical data collection for testing a system such as an APS may not always be feasible, as described in Section 2.2.1. Therefore, pre-existing datasets (Definition 2) may be used in testing, meaning there is no guarantee that inferred associations may not be spurious. If we use the state-of-the-art technique (described in Section 3.4.1), the surrogate model may struggle to account for incorrect associations [58], learning an inaccurate representation of system behaviour. The testing procedure may, therefore, suggest test cases that are not representative of system violations.

The state-of-the-art surrogate modelling techniques may also struggle to represent APS behaviour that may appear inconsistent. For example, the insulin required for a specific consumption of carbohydrates may differ due to the intricate nature of blood glucose-insulin dynamics [142]. These behaviours lack a precise test oracle [15]. Therefore, defining which behaviours are correct and incorrect is non-trivial. As a result, the testing procedure may struggle to determine whether the resulting behaviour is incorrect.

Causal inference provides mechanisms to deal with these limitations in the context of testing. In prior works [27, 159] (including P2), causal inference has been used to account for pre-existing data and complex, seemingly inconsistent behaviours in software testing. In this chapter, we incorporate the opportunities provided by causal inference to alleviate the difficulties of applying surrogate-assisted CPS testing.

In the remainder of this chapter, we propose the use of a *causal surrogate model* and integrate it with the existing surrogate-assisted testing technique. The causal surrogate model enables a tester to more effectively identify scenarios that result in system violations for systems where behaviours lack observability (Definition 4) and datasets lack controllability (Definition 3). Our surrogate model implements causal inference techniques, described in Section 3.6, utilising domain knowledge to build more accurate surrogate models.

4.1. Integration with Surrogate-Assisted CPS Testing

Figure 7 presents how our surrogate model integrates with the existing approach. In Section 3.4, we described how the state-of-the-art surrogate-assisted testing approach implements an associative surrogate model. This surrogate model represents associations between system inputs and system violations from training data.

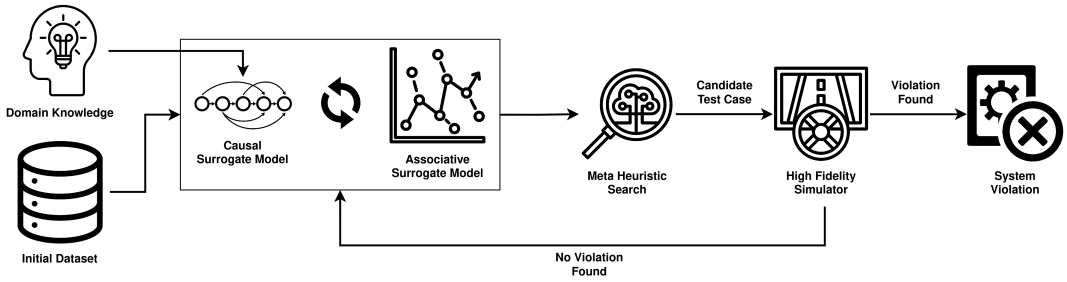


Figure 7.: Integration of the causal surrogate model into surrogate-assisted CPS testing. Domain knowledge is used to generate the causal surrogate model. Both surrogate models are used in turn to build a more diverse dataset of behaviours to rebuild the surrogate models for later search iterations.

We use our surrogate model to complement, not replace, the associative surrogate model to provide a combination of different perspectives of system behaviours. As a result, the non-spurious associations (Section 3.4.1) can be verified by the associative surrogate model. Our causal surrogate model can then use domain expertise to enable CPS testing where the associative approach may struggle.

Causal inference techniques enable the tester to answer causal questions about system behaviour and account for potentially inferring spurious associations from pre-existing data [27]. These factors provide an opportunity to alleviate the testing difficulties associated with testing CPSs with the current state-of-the-art surrogate-assisted testing techniques (Section 3.4.6).

4.1.1. Complementary Surrogate Models

We generate both an associative surrogate model, which correlates inputs against outputs (explained in Section 3.4.1), and a causal surrogate model (further explained in Section 4.2). We alternate between each model (associative and causal) with each complete search procedure iteration. Algorithm 1 presents how the surrogate model is selected.

Algorithm 1: Select and Train Surrogate Model

Input : Initial Dataset (D),
Search Iteration (i)
Output: Surrogate Models (M)

```
Set of Surrogate Models M ← {}
if i is odd then
    M ← TrainAssociativeSurrogateModel(D)
else
    M ← TrainCausalSurrogateModel(D)
return M
```

For each odd search iteration, one surrogate model is selected and trained, and for each even iteration, the other is selected and trained. As a result, the first meta-heuristic search iteration will find a candidate test case based on the associative model, and then, providing

no violation is found, perform the next meta-heuristic search using the causal surrogate model. As with the technique described in Section 3.4, we use both models to provide a simplified representation of the system's behaviour that can be used to find system violations.

A meta-heuristic algorithm searches for a candidate test case based on the fitness function of the surrogate model being used. For the associative model, the fitness function maximises the likelihood of system violations based on learnt association with model inputs. For the causal surrogate model, the fitness function maximises for causal relationships between system variables that do not hold.

By using both models, we search system behaviours by both exploiting associative properties in the data (associative surrogate model), as well as evaluating domain knowledge of the relationships between variables (causal surrogate model). Test cases that do not result in a violation are fed back into the initial dataset to retrain both surrogate models.

Using our running APS example from Section 3.6, the associative surrogate model enables a tester to find consumptions of *Carbohydrates* that result in dangerous *Blood Glucose Levels* by exploiting existing associations in the data. These can be found where dangerous *Blood Glucose Levels* are associated with consumptions of *Carbohydrates*. However, due to a lack of controllability (see Definition 3) requiring the use of pre-existing datasets, such associations may not always be causal [58].

Where associations are not available, in a subsequent search iteration the causal surrogate model enables a tester to find instances where this causal relationship does not hold. The causal surrogate model answers causal questions, such as "What if we increase carbohydrate consumption? Will this cause the expected increase in insulin prescription?" These found instances would likely result in incorrect system behaviours and, potentially, lead to dangerous *Blood Glucose Levels*.

By alternating between different surrogate models in each search iteration, we can combine the resulting data into a combined training dataset. This dataset aims to provide a wider variety of system behaviours from which to train surrogate models. Data from the associative approach is used to build the causal surrogate models and vice versa. As a result, we retrain the models with data outside their typical search behaviours. This builds upon the data feedback technique outlined in Section 3.4.4 as a way of iteratively *learning* about the system's environment.

Having two different surrogate models working together does pose the question as to the efficacy of the causal model alone. After evaluating the applicability of our complimentary approach, we also evaluate the causal surrogate model alone in Section 7.4.

4.1.2. Incorporating Domain Knowledge in CPS Testing

By including a causal surrogate model, we enable a tester to account for a lack of *observability* and *controllability* in CPS testing.

A causal surrogate model accounts for *observability* by answering "what if" questions about system behaviours by evaluating ATEs of the expected relationships between variables (Section 3.6.2). System behaviours that lack a precise specification can, therefore, be tested.

For example, an APS should increase its *Insulin Prescription* if more *Carbohydrates* are consumed. This increase will be different for different people or at different times though, due to the varied nature of blood glucose-insulin dynamics [142]. Tests that rely on specific values of *Carbohydrates* resulting in specific values of *Insulin Prescription* may, therefore, not be reliable.

Nevertheless, the *Insulin Prescription* should increase when more *Carbohydrates* are consumed. This expected increase would allow a tester to answer the causal question "What if I increase the amount of carbohydrates consumed? Does this cause the expected increase in insulin prescription?" System executions for which this causal relationship does not hold suggest that the system may be behaving incorrectly.

Including a causal surrogate model also allows a tester to account for a lack of *controllability*. The surrogate can identify the interpretation of spurious associations from pre-existing datasets through causal identification (Section 3.6.3). Pre-existing data can then be used when training surrogate models where physical data collection for testing may not be feasible.

Spurious associations are more prevalent in systems where exhaustive physical data collection is not feasible, such as an APS [27, 81, 150]. By using causal inference to adjust for the potential of spurious associations, we remove the requirement for curated datasets. As a result, we allow for pre-existing datasets to be used in system testing.

For our approach, we decided to use meta-heuristic search, following the approach defined in Section 3.4. Other techniques, such as combinatorial coverage [92], would have covered the solution space, but may have increased the computational complexity of testing. Others, such a configuration testing [178], would have only executed a few key system inputs, but may have missed inputs that result in system violations. By using meta-heuristic search in our later evaluations, we ensure the only difference between the baseline and our approach is the causal surrogate model.

4.2. Causal Surrogate Model

In order to account for biases in the training dataset and external unmeasurable factors of the system-under-test (SUT), we generate a *causal surrogate model*. As with associative models, this model approximates SUT behaviour in a manner that is less expensive to execute than a high-fidelity simulator. However, unlike associative ones, this model evaluates the expected relationships between variables and uses domain knowledge to adjust for the interpretation of spurious associations from the training data.

The causal surrogate model allows domain knowledge about the SUT to be encoded through a causal DAG. The resulting surrogate model enables the tester to answer causal questions (e.g., "what if ...") about system behaviours that may be affected by external factors and, therefore, challenging to define for associative approaches (lack of *observability*). This is achieved while using the causal DAG to identify potential data that may give rise to spurious associations (lack of *controllability*). Potential spurious associations can then be accounted for in surrogate model predictions.

4.2.1. Constructing the Causal DAG

Before generating our causal surrogate model, a tester must construct a causal DAG of the SUT. Constructing a causal DAG is a manual process requiring domain knowledge of the SUT to demonstrate the expected behaviour between system variables [132]. However, a causal DAG enables a tester to more effectively test a CPS where system behaviours are affected by external factors and physical data collection is not feasible. Answering causal questions about system behaviours can account for a lack of observability, and alleviating the interpretation of spurious associations from pre-existing datasets can account for a lack of controllability [27, 54].

To generate a causal DAG, a tester must identify the nodes and edges of the graph. The nodes represent the system variables, and the edges represent the expected causal relationships between said variables. Identifying the nodes is fairly simple, as they are the variables that are available in pre-existing datasets. By extracting these variables from pre-existing datasets, the tester can then start establishing the expected causal relationships of the system.

However, identifying the causal relationships between system variables requires domain knowledge of the SUT. These relationships cannot, typically, be extracted from pre-existing datasets (discussed further in Section 8.5) and, therefore, must be generated manually. A tester expresses their expectations of how changing one variable will cause a change in another. For example, by increasing the amount of *Carbohydrates* consumed, a tester would expect an increase in the *Insulin Prescription* from an APS.

Each expected relationship is derived from the testers understanding of the system. The resulting causal DAG, therefore, represents the expected correct behaviour of the system. We note a parallel between the causal DAG and the test oracle, as seen in Section 3.1. We further discuss the difficulty of generating a causal DAG and its susceptibility to the *oracle problem* (Definition 7) as a potential threat to our approach in Section 8.6.1. However, by allowing the tester to account for a lack of both observability and controllability, a causal DAG may allow a tester to more effectively test CPSs.

It should also be noted that the lack of an edge between two variables is as important as the presence of an edge [27, 29, 132]. In some cases, changing a variable should have no direct effect on another variable. This is a property that can then be tested, ensuring independence when that is the expected behaviour. For example, using the ADS example from Section 3.4, it is expected that an APS should not crash more often given an increase in the number of pedestrians.

4.2.2. Causal Model Generation

In order to generate a causal surrogate model, the tester must supply two pieces of information: an initial dataset and the generated causal DAG. The initial dataset is used to train the model and learn the functional nature of the causal relationships of the system. With respect to Figure 7, this section describes how the causal surrogate model is generated from the initial dataset or as more data is fed back into the model.

Algorithm 2: Causal Surrogate Model Generation

Input : Causal DAG (V, E),
Initial Dataset (D)
Output: Set of Surrogate Models (M)

1 Set of Surrogate Models $M \leftarrow \{\}$
2 **for** *Edge e in E* **do**
3 Set of Bias-inducing Variables $B \leftarrow$ *CausalIdentification*(e, V, E)
4 Surrogate Model Structure $m_s \leftarrow$ *GenerateModelStructure*(e, B)
5 Trained Surrogate Model $m \leftarrow$ *TrainModel*(m_s, D)
6 $M \leftarrow M \cup \{m\}$
7 **return** M

Algorithm 2 presents the approach used to generate the causal surrogate models for a system. The aim of this process is to generate a causal surrogate model to represent each relationship of the causal DAG. The process takes a causal DAG (V, E) (see Definition 9) and an initial, potentially pre-existing dataset (D) as inputs. For each edge (e), we perform causal identification [134] (Section 3.6.3) to uncover other variables (B) that may introduce bias.

We then generate a *causal model*, as described in Section 3.6.2, to represent system behaviours in a computationally efficient manner. We generate the causal surrogate model's structure (m_s) for each edge while accounting for bias-inducing variables. Our models can then be trained against the initial dataset in order to represent the causal relationships between variables of the system.

In *GenerateModelStructure* (line 4), we adjust model estimation (Equation (3.2)) in order to account for bias-inducing variables (lack of *controllability*). By adjusting model estimations, bias-inducing variables (uncovered from causal identification [134]) can be included in the surrogate model and held constant. This means that the causal surrogate model estimates the relationship between only the variables of the edge it is representing, independent of other, potentially bias-inducing, SUT variables [147].

For example when testing an APS, the edge *Carbohydrates* $\rightarrow$ *Insulin Prescription* in a causal DAG could be represented by a causal surrogate model. The DAG itself is used in causal identification to identify any other potentially bias-inducing variables to be held constant when generating the model. In Section 3.6.3, we described an example of how *Blood Glucose Levels* could induce bias and would, therefore, need to be adjusted for in the causal model. The model could then be trained on historical *Carbohydrates*, *Insulin Prescription* and *Blood Glucose Levels* in order to estimate the relationship between *Carbohydrates* and *Insulin Prescription*.

In contrast to existing approaches, outlined in Section 3.4, we generate a separate causal surrogate model for each causal relationship of interest in the DAG. Given data for each relationship, we use a cubic spline regression to fit the data, following prior works [27,28]. A cubic spline regression presents a computationally efficient simplification of the relationship between two variables. Furthermore, it can represent non-linear behaviour across inputs and include bias-inducing variables (B) as constants.

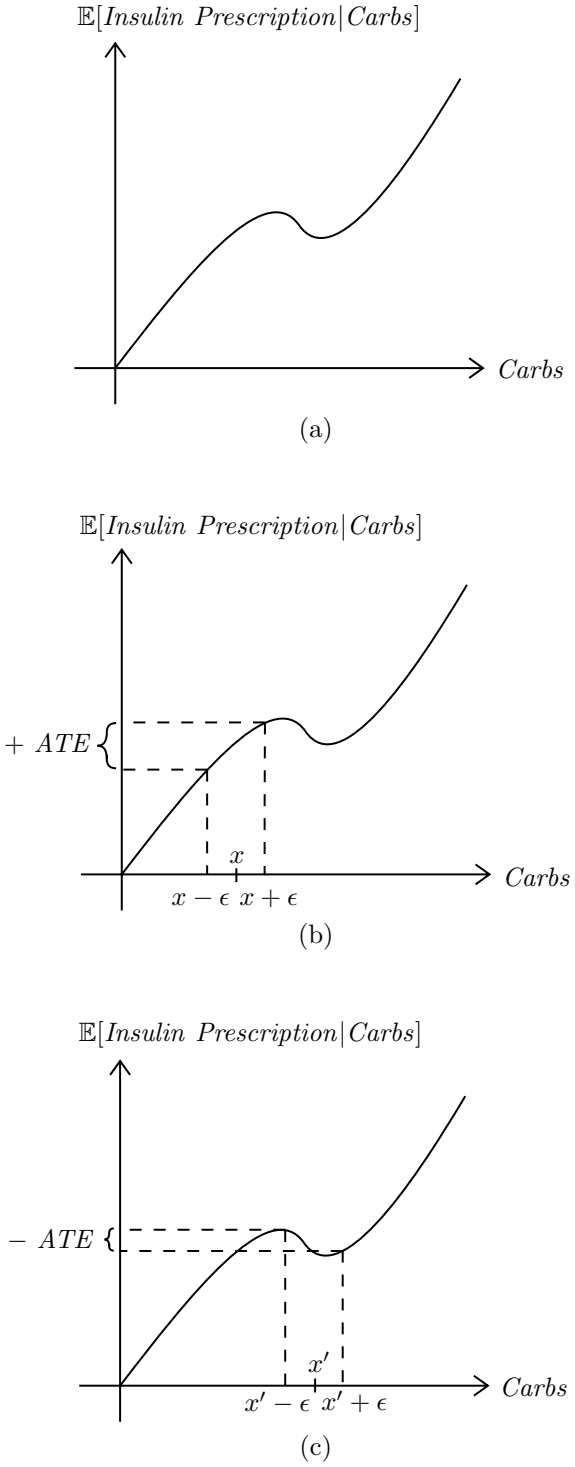


Figure 8.: The solution space of a cubic spline regression model representing the relationship between variables *Carbohydrates* (*Carbs*) and *Insulin Prescription*. (a) presents the surrogate models estimated relationship. (b) presents an expected positive relationship at a point x in the input space. (c) presents an unexpected negative relationship at a point x' in the input space.

Figure 8 (a) presents how a cubic spline regression can be used to represent the relationship between two variables, which we use as an example throughout this section for illustrative purposes. This example follows our APS motivating example from Section 2.2 and the causal relationship presented in Section 3.6.3.

The model in Figure 8 (a) represents the relationship between the variables *Carbohydrates* and *Insulin Prescription* where *Carbohydrates* $\rightarrow$ *Insulin Prescription* appears in the causal DAG (*Carbohydrates* has been abbreviated to *Carbs* in the figure). The cubic spline regression would be trained from pre-existing data of the SUT. We use this model to estimate values of *Insulin Prescription* given values of *Carbohydrates*, based on their relationship in the training data while accounting for any bias-inducing variables.

4.2.3. Causal Model Evaluation

Given the causal surrogate models generated in the previous section, we are able to find incorrect behaviour by answering causal questions. By finding contradictions between the expected relationships between variables and those of the causal model trained from system behaviour, we can drive the search procedure towards behaviours where causal relationships do not hold.

Taking the example from Section 3.6 and Figure 8, our causal surrogate model enables us to to answer the causal question, "What if we increase carbohydrate consumption? Will this cause the expected increase in insulin prescription?" We can then search for scenarios where increasing the amount of *Carbohydrates* consumed by a user does not increase *Insulin Prescription*. Such scenarios are assigned a high fitness value used to search for contradictions in expected behaviour in meta-heuristic search.

Notice that our technique does not directly identify scenarios that cause violations; instead, it evaluates scenarios in which causal relationships do not hold. For example, our technique does not identify behaviour that causes hyperglycaemia; instead, it identifies incorrect behaviour that may result in it. This approach is in contract to the surrogate models described in Section 3.4.1, which evaluate inputs based on their direct likelihood of causing a violation by learning associations. The remainder of this section describes the process of evaluating the causal surrogate model during meta-heuristic search, as presented in Figure 7.

In order to drive the meta-heuristic search to find contradictions in the expected causal relationships between variables, we define a fitness function. This fitness function maximises for inputs where a causal relationship does not hold. As a result, a meta-heuristic search can identify inputs for which causal relationships that are not representative of their expected behaviour.

ATEs (Section 3.6.2; Equation (3.1)) allow us to answer causal questions about the surrogate model's outputs, such as "What if a user consumes more carbohydrates? Would that cause the expected increase in insulin prescription?" By doing this, we can define the fitness function for this model in terms of the gradient of the model outputs [44].

To calculate the ATE, we evaluate the model at two distinct inputs and calculate the

difference in the output. As a result, we can observe how changing one variable from one value to another would affect the value of another variable. This approach is akin to checking a metamorphic relationship (Section 3.1.1). Equation (4.1) represents this by implementing Equation (3.1) through a mutation of model inputs (x) by a deviation (ϵ). This deviation is used to calculate system behaviour using the causal model with inputs just greater and less than the input of interest. As a result, an ATE can be calculated at this point, representing the gradient at the point of interest (Figure 8).

$$ATE = \mathbb{E}[Y|\ t = x + \epsilon] - \mathbb{E}[Y|\ t = x - \epsilon] \tag{4.1}$$

Following the example from the previous section, Figures 8 (b) and (c) present how the ATE for the relationship *Carbohydrates* $\rightarrow$ *Insulin Prescription* can be calculated. The causal surrogate model predicts *Insulin Prescription* ($\mathbb{E}[$*Insulin Prescription*$|$*Carbohydrates*$]$) for an input of *Carbohydrates* (x). *Insulin Prescription* is estimated at input of interest (x) plus or minus a small deviation (ϵ).

Figure 8 (b) presents a value of *Carbohydrates* for which the expected causal relationship holds. Assuming the expected causal relationship of *Carbohydrates* $\rightarrow$ *Insulin Prescription* is positive, this value of *Carbohydrates* would produce a positive ATE (Equation (4.1)) and therefore be assigned a low fitness score. This is because the ATE does not contradict the expected causal relationship between *Carbohydrates* and *Insulin Prescription*.

Figure 8 (c), however, presents a value for *Carbohydrates* for which the expected causal relationship does not hold. By assuming the same positive expected relationship of *Carbohydrates* $\rightarrow$ *Insulin Prescription*, the value x' would result in a negative ATE being calculated. This does not represent the expected relationship between *Carbohydrates* and *Insulin Prescription* and could, therefore, be an input that produces an SUT violation. We assign such inputs a high fitness score.

For our upcoming evaluations, we define 3 fitness functions for different expected system behaviours. Equation (4.2) presents the fitness functions for expected positive, negative and independent relationships when increasing the treatment variable. Since our approach aims to identify contradictions of expected relationships, we define the fitness value as the opposite of the expected ATE. For example, in our example of increasing *Carbohydrates* expecting an increase in *Insulin Prescription*, this is a positive relationship. Therefore, the fitness negates the ATE, resulting in a high fitness being assigned for a negative ATE.

$$Fitness = \begin{cases} -ATE, & \text{if } Expected = Positive \\ ATE, & \text{if } Expected = Negative \\ |ATE|, & \text{if } Expected = Independent \end{cases} \tag{4.2}$$

4.2.4. Surrogate Search Procedure

After defining a fitness function, we can use meta-heuristic search to find causal relationships that do not hold. Algorithm 3 presents the procedure of searching for system violations from the surrogate models.

Having generated the surrogate models (M), we can then search each model for contradictions in their expected behaviour. We achieve this by taking each surrogate model (m) and using a meta-heuristic search algorithm (A_{MH}) to maximise the fitness values defined in Section 4.2.3. Each surrogate model finds a set of inputs (i) that maximise the fitness value (f) for that single expected relationship.

Algorithm 3: Surrogate Model Violation Search

Input : Surrogate Models (M),
Meta-heuristic Search Algorithm (A_{MH})
Output: Surrogate Model (m),
Set of Inputs (i)

1 Set of Model Fitnesses $F \leftarrow \{\}$
2 **for** *Surrogate Model* m *in* M **do**
3 Inputs and Fitness $(i, f) \leftarrow CalculateFitness(m, A_{MH})$
4 $F \leftarrow F \cup \{(m, i, f)\}$
5 $(m, i) \leftarrow MaxFitness(F)$
6 **return** (m, i)

MaxFitness on line 5 of Algorithm 3 identifies the causal relationship and corresponding inputs with the highest fitness (m, i). As a result, we highlight the causal relationship and corresponding inputs that cause the largest contradiction in expected behaviour. These inputs are most likely to result in incorrect system behaviour. The high-fidelity simulator can be executed using the inputs corresponding to the highest fitness value, simulating only the scenario that most likely leads to incorrect behaviour.

Using our prior example, we can use meta-heuristic search to find increases in *Carbohydrates* that result in a decrease in *Insulin Prescription*. Since we assume an increase in *Carbohydrates* should lead to an increase in *Insulin Prescription*, we can identify values of *Carbohydrates* where this relationship does not hold (such as Figure 8 (c)). As a result, only values of *Carbohydrates* that result in a high fitness, and therefore cause potentially incorrect behaviour, are evaluated on the high fidelity simulator. As a result, we reduce the computational expense of simulating other values of *Carbohydrates*.

By using a causal DAG, we are able to deconstruct our search procedure to a set of independent objectives. As a result, we can evaluate each causal relationship separately. Decomposition presents an opportunity to use more computationally efficient single-objective search algorithms [99], as opposed to multi-objective search used in prior works [62, 70]. We assume independence between the causal relationships by including bias-inducing variables in the surrogate models [1, 133].

4.3. Chapter Summary

In Section 3.4.6, we described how the traditional surrogate models used in surrogate-assisted CPS testing may be affected by a lack of controllability and observability. Traditional approaches may, therefore, incorrectly interpret pre-existing datasets and struggle to evaluate inconsistent behaviours for systems such as an APS.

In this chapter, we introduced the *causal surrogate model* to enable these challenges to be addressed using domain knowledge. This surrogate model implements causal identification to account for potential biases in the training data, reducing the likelihood of learning spurious associations. It allows a tester to evaluate system behaviours by answering causal questions (e.g., "what if ..."). As a result, the meta-heuristic search in surrogate-assisted testing techniques can find expected causal relationships that do not hold.

In Chapters 5 and 7, we evaluate the efficacy of our new surrogate model compared to the state-of-the-art surrogate-assisted testing approach. From these evaluations, we aim to identify whether our approach improves the effectiveness of testing for systems that lack observability and controllability, therefore increasing the applicability of surrogate-assisted CPS testing.

Chapter 5.
Evaluation - Replication of an Existing Study

As a first step in the evaluation of causal surrogate models, we start with a replication of a prior study before moving on to a more in-depth evaluation. We compare the use of our causal surrogate model to the approach described in Haq et al. [70], as described in Section 3.4. We also perform this evaluation based on their subject system, the automated driving system (ADS) Pylot [65], which has also been used in other evaluations [62]. We use a single research question (RQ) to drive our evaluation:

RQ1: Can using causal surrogate models improve the effectiveness of surrogate-assisted ADS testing?

5.1. Evaluation Methodology

For this study, we evaluate the effectiveness of the surrogate models used in current surrogate-assisted CPS testing approaches compared to our causal surrogate model. We use a testing procedure that follows the description of surrogate-assisted CPS testing in Section 3.4 and applied the proposed approach to Pylot. The remainder of this section introduces the requirements for the baseline testing procedure [70] (simulator, violations, initial dataset) as well as the additional requirements for our causal surrogate models (causal DAG).

To perform this evaluation, we follow Figure 7:

1. We identify initial datasets from which to build our surrogate models, generate a causal DAG to specify the additional domain knowledge for the causal surrogate model, identify a set of system violations that the search procedure attempts to find, and obtain a simulation environment to validate those potential system violations.

2. We generate the surrogate model required for the current testing iteration. We start the testing procedure with the associative surrogate model, then build the causal surrogate model for the second iteration, repeating this alternation for each testing procedure iteration. The data used to generate the surrogate is the initial dataset and any outputs from prior search iterations.

3. We use the chosen surrogate model as the fitness function of a meta-heuristic search algorithm. This search finds the scenario that will most likely result in a system violation (candidate test case) based on associations with violations (associative surrogate) or causal relationships that do not hold (causal surrogate).

4. We then execute the found candidate test case on the high-fidelity simulator. As a result, we determine whether the given scenario results in a real system violation.

5. If no violation is found, we introduce this new data back into our initial dataset. The testing procedure is repeated, alternating between each surrogate model, until a system violation is found or a computational budget is exceeded.

6. We repeat this approach for each initial dataset, recording the search iteration at which the first system violation was found for each dataset.

In order to answer RQ1, we compare our approach with the state-of-the-art technique. We can then determine which approach can find violations more efficiently with datasets that are not curated. For each technique, we measure the number of search iterations required to find an initial violation for a set of pseudo-random initial datasets. As a result, we establish whether our approach reduces the computational expense of testing Pylot.

5.1.1. Subject System, Simulator and Violations

Pylot is an automated driving system (ADS) with the goal of controlling an ego car to successfully navigate driving scenarios [65]. It is comprised of multiple pre-trained deep neural network modules, each with goals such as object detection [102] and tracking [18,173]. These neural networks, however, reduce the internal understandability of the system, making the classification of "correct" behaviour challenging.

Pylot is an example of a system that may lack observability (Definition 4). ADSs are CPSs and, therefore, affected by the external world around them. In Section 2.1.1, we described how an ADS's behaviour may appear inconsistent for the same set of inputs due to external factors, such as another driver's behaviour [156]

Pylot may also lack controllability (Definition 3). Similar to an APS, for ADS behaviour to be observed, the system must interact in an environment that may give rise to dangerous behaviours [7]. The user, other drivers and surrounding pedestrians could by impacted by incorrect ADS behaviour, making it more difficult to cover all system behaviours. As a result, pre-existing datasets may be required for use in testing, which may not contain behavioural coverage.

For the simulator, we use CARLA [45], a driving simulator for trialling ADS algorithms, such as Pylot. It provides a high-fidelity environment for highly configurable driving scenarios to support ADS modules, such as computer vision and LiDAR. Pylot's interface with CARLA provides a modular approach to scenario generation, allowing a user to specify the inclusion or exclusion of certain factors in the driving scenario.

We measure the same Pylot violations defined in Haq et al. [70], which are presented in Section 3.4.2. We reiterate these 6 potential violations as follows: being too far from the centre of the road, not avoiding other vehicles, not avoiding pedestrians, not avoiding other obstacles, not abiding by traffic laws, and not reaching the scenario destination.

In this evaluation, we only ran the testing procedure until an initial violation was found. By only investigating the initial violation, we focus on evaluating only the difference between

Table 1.: The input variables and potential violation variables for Pylot. Each input variable is given a set of categorical inputs to be selected. Each potential violation is returned from Pylot as a number between 0 and 1 representing the likelihood of the violation.

Variable Name	Potential Values
road_type	{Straight, Left turn, Right Turn, Cross Road}
road_id	{0, 1, 2, 3}
scenario_length	{0}
time	{Noon, Sunset, Night}
weather	{clear, cloudy, wet, wetcloudy, SoftRain, MidRain, HardRain}
pedestrian_density	{True, False}
target_speed	{20, 30, 40}
trees	{True, False}
buildings	{True, False}
task	{follow_road, take 1st exit, take 2nd exit}
vehicle_front	{True, False}
vehicle_adjacent	{True, False}
vehicle_opposite	{True, False}
vehicle_front_two_wheeled	{True, False}
vehicle_adjacent_two_wheeled	{True, False}
vehicle_opposite_two_wheeled	{True, False}
follow_center	[0 - 1]
avoid_vehicles	[0 - 1]
avoid_pedestrians	[0 - 1]
avoid_static	[0 - 1]
abide_rules	[0 - 1]
reach_destination	[0 - 1]

the two surrogate models and no other part of the testing procedure. We used the fitness functions defined in Section 4.2.3 for the causal surrogate model in order to find scenarios in which the causal relationships do not hold.

5.1.2. Datasets

In Section 2.1.1, we describe how a lack of controllability may require pre-existing datasets to be used in CPS testing. These datasets are likely not curated and exist from prior executions of the system. It is possible that spurious associations can be inferred from these datasets due to the data collection process or a lack of behaviour coverage. We describe how this may affect surrogate-assisted testing in Section 3.4.1. For Pylot, such pre-existing datasets do not exist. For this evaluation, we instead generated pseudo-random datasets to represent pre-existing datasets.

Pylot requires 16 input variables and, after simulation, returns 6 potential violations as outputs. To mimic pre-existing data from which to build surrogate models, we produced 30 datasets that contained random distributions of system behaviours. To follow the evaluation set out in Haq et al. [70], our generated datasets consisted of 39 'scenarios', each consisting of a set of 16 inputs and the resulting 6 violations. A surrogate model was then trained

from a dataset consisting of these 39 scenarios to represent system behaviour. We repeated our experiment 30 times with different pseudo-random datasets to present a distribution of pre-existing data from which the surrogate models were trained. Each testing technique was evaluated using the same 30 datasets with the same initial random seed. Table 1 presents each input and violation with their respective potential values.

Our evaluation mimics the use of pre-existing datasets, contrasting the dataset used in the original study. Haq et al. [70] generated a single dataset with 4-way combinatorial coverage [92] of the input space to use as an initial dataset to build the surrogate model. In Section 2.1.1, we described how such controlled data may not be available and, therefore, is a testing challenge for CPSs.

5.1.3. Causal DAG

Recall that our causal surrogate approach requires a causal DAG to identify variables that may potentially introduce biases. In Section 4.2.1 we explained how generating this causal DAG can be non-trivial. A tester must first identify the variables available in pre-existing datasets, which is fairly straightforward. However, they must then use their domain expertise to designate the expected causal relationships between these variables, requiring an in-depth knowledge of the behaviours of the SUT.

Figure 9 presents the causal DAG representing a single Pylot scenario, presenting the expected relationships between 16 scenario inputs (middle 2 rows) and 6 potential violations for Pylot (bottom row). This DAG represents a black-box example where the internal variables of the system are inaccessible, so only the causal relationships between inputs and violations can be represented.

As well as the inputs and violations, we include an unmeasurable variable *search_bias*. We use this variable to represent the feedback of the search process as prior values of each input variable will affect the distribution of values in the resulting dataset. It is important to include this in our causal DAG to represent how the inputs may be indirectly influenced by one another.

We expect no causal relationships between inputs and potential violations. In the DAG, we represent this as a lack of an edge between the inputs and violations. This is based on the domain knowledge that a correctly working ADS should account for different scenarios without causing violations. For example, we assume that regardless of the density of pedestrians, Pylot should still not hit pedestrians.

We also add edges between the *search_bias* and the input nodes. These relationships are not part of the ADS so we do not test them. However, they are used in causal identification (Section 3.6.3) to identify potential biases in the dataset.

Figure 9.: A causal DAG of Pylot. We illustrate causal relationships of interest that should not exist between the scenario inputs (middle 2 rows) and the potential violations (bottom row) with a lack of an edge. We also include relationships from the search procedure (dashed line). Although it may look strange that there are no relationships between inputs and violations, the reader is reminded that this is a representation of expected behaviour based on specification, not actual system behaviour.

Table 2.: The configuration of the meta-heuristic search algorithm PyGAD when searching for system violations. This was derived from preliminary experimentation.

Parameter	Value
Generations	200
Solutions per Generation	10
Parent Selection	Tournament
Number of Mating Parents	4
Mutation Percentage	50%

5.1.4. Baseline

To evaluate the effectiveness of our approach with pre-existing datasets, we compare the number of simulations required to find an initial violation to that of the *Ensemble*, i.e., the ensemble surrogate model outlined in Haq et al. [70]. This ensemble surrogate model comprises a polynomial regressor, gaussian process regressor and radial bias function network, as described in Section 3.4.1. *Random Search* is also evaluated to show the efficacy of both techniques.

For this evaluation, we use the genetic algorithm PyGAD [60] as the meta-heuristic search algorithm in our surrogate-assisted testing approach. PyGAD provides an open-source implementation of a single-objective optimisation algorithm. Haq et al. [70] also used a genetic algorithm in their evaluation. The configuration for the genetic algorithm used in our evaluation is presented in Table 2 and also available in our replication package, described in Section 1.1.2. Fitness functions for both our and the state-of-the-art approaches are presented in Sections 3.4.1 (*Ensemble*) and 4.2.3 (*Causally-Assisted*).

5.2. RQ1 Results: Effectiveness

For our evaluation, we report our findings using the same technique as used in Haq et al. [70]. Figure 10 illustrates the total number of violations found at each search iterations across the 30 traces. When a trace results in a system violation, the corresponding search iteration is increased by 1. As a result, we can observe the number of traces that have resulted in a system violation throughout the search procedure for each testing technique.

Figure 10 presents the number of search iterations required to find a system violation cumulatively across each of the 30 datasets. Both the *Ensemble* and *Causally-Assisted* approaches were able to find violations in all 30 datasets, whereas *Random Search* found violations in fewer than 15 datasets.

Our causal approach is more effective, requiring 3 search iterations, whereas the baseline approach required 4. This shows that our approach is able to find violations using pre-existing datasets with fewer iterations than the associative method. Fewer iterations would require less use of the high-fidelity simulator to check whether a scenario results in a system violation. By reducing the simulators use, computational expense is also reduced, allowing the resources and time to be allocated elsewhere.

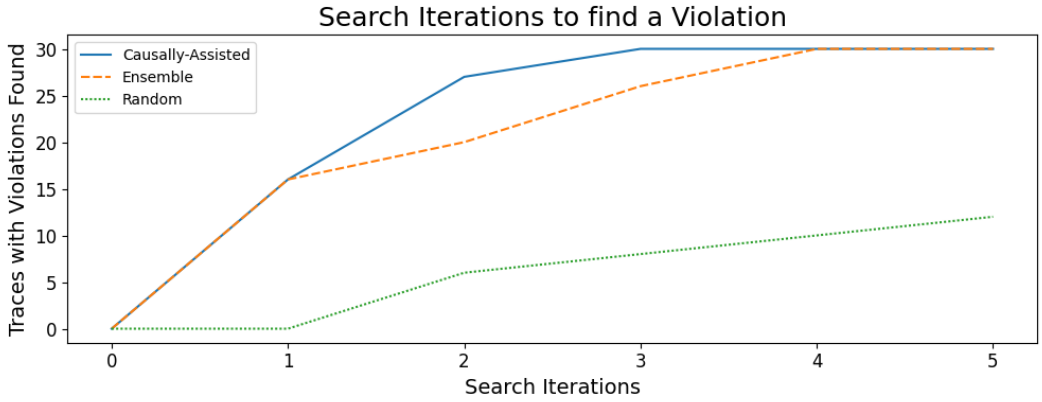


Figure 10.: Total number of traces where a Pylot violation was found and the search iteration at which it was executed on the simulator. The testing procedure was executed once for each of the 30 initial pseudo-random datasets.

The initial trajectory of both approaches is the same, showing how both approaches easily find violations where inputs are highly correlated with violations in the surrogate training data. However, the baseline approach then only finds 4 violations in the next iteration whereas our approach is able to find 11. In this iteration, our approach uses the causal surrogate to check whether causal relationships do not hold, more effectively exploring unknown behaviours than the associative approach. This better accounts for datasets where associations between inputs and violations may not exist, leading to fewer search iterations that require high-fidelity simulation. From this, we infer that our approach was able to find violations more successfully where the dataset is not curated.

Summary: The results of this experiment show that we were able to increase the effectiveness of surrogate-assisted ADS testing. We were able to find violations in all 30 pseudo-random initial datasets within 3 search iterations. The state-of-the-art technique required 4 search iterations. We, therefore, reduced the computational expense of finding violations for an ADS.

5.3. Statistical Significance

Table 3.: The number of Pylot violations found and the resulting Pearson's chi squared values calculated for them at different search iterations. Chi Squared values outlined in **bold** indicate an inability to demonstrate independence. Iterations 4 and 5 are omitted since all violations are found. Values are given to 5.d.p.

Search Iteration	Violations Found (Ensemble)	Violations Found (Causally-Assisted)	Pearson's Chi-Square Value
1	16	16	**0.00000**
2	20	27	4.81178
3	26	30	4.28571

To ensure our evaluation demonstrates a significant difference between the state-of-the-art approach and our own, we performed a statistical significance analysis [10]. To achieve this, we performed Pearson's Chi Squared test [24], henceforth referred to as the cs-test, on the results from RQ1.

The cs-test enabled us to reject the null hypothesis that the results of our approach are not independent from those of the state-of-the-art. A higher cs-test value determines a stronger rejection of this null hypothesis and, therefore, that our result are statistically independent from the state-of-the-art. The cs-test was chosen due to our data being categorical (violation either found or not found) at each timestep.

We performed the cs-test at each timestep of the evaluation from search iteration 1 to search iteration 3. Search iterations 4 and 5 were omitted due to there being no difference in the number of violations found between the techniques. The resulting value from the cs-test was verified against a p-value table to determine whether our technique was statistically significant to the state-of-the-art. In this case, a p-value of 0.05 was used and therefore a cs-test-value greater than 3.841 suggests statistical independence [145].

Table 3 presents the cs-test values across the the testing procedure. Search iterations 2 and 3 demonstrate statistical independence and, therefore, a difference in the results of our approach and the state-of-the-art. This pattern can be seen mirrored in Figure 10 as search iteration 2 is where the two approaches diverge as our approach finds additional violations.

5.4. Threats to Validity

In this section, we present some potential threats to validity of this evaluation. We focus specifically on the internal validity of this evaluation. In Section 8.6, we present a more in-depth threats to validity in which we discuss the internal and external threats of our approach.

A potential threat is that the datasets used in this evaluation were not representative of pre-existing datasets. We generated the datasets used in this evaluation to mimic the structure of those used in the replicated study [70]. However, our datasets differed from the data used in this study by being randomly sampled, instead of using combinatorial coverage, to mimic pre-existing, non-curated datasets. 30 different datasets were generated using this approach. This allowed us to ensure that the same 30 datasets were used in the evaluation of the causally-assisted, associative and random search approaches. Each testing technique was executed using the same initial random seed to ensure both reproducibility and that the only difference between each execution was the testing technique.

It could be argued that our single research question does not adequately evaluate our approach. We use this evaluation to simply test the efficiency of our approach in the context of an existing evaluation. By doing so, we were able to quickly determine whether our approach has merit before approaching a more complex, system without an existing evaluation methodology. In Chapter 7 we build upon this methodology to further evaluate our approach beyond just its effectiveness.

Similar to the last point, it could be argued that the subject system of this evaluation

is not a good candidate. We chose to replicate this study due to Pylot's use in multiple prior evaluations [62, 70]. Pylot also presents a large and complex input space and human interaction in the form of other drivers and pedestrians, making it very difficult to physically curate datasets for this system. Due to this, Pylot demonstrates similar testing challenges to our APS, as presented in Section 2.2.1.

A potential threat to validity is that we only provided a single statistical significance test [10]. Due to the limited number of results in this evaluation (30 outcomes of a violation found or not found), we found only Pearson's Chi Squared test appropriate for these results. We do, however, take this into account when performing our upcoming evaluation in Chapter 7. This evaluation contains a significantly larger dataset, allowing for the application of more statistical tests to further analyse the significance of our results.

5.5. Chapter Summary

In this chapter, we evaluated the use of our causal surrogate model in the replication of an existing study that performed surrogate-assisted testing of an APS. We found our approach was able to find violations in all initial pre-existing datasets in fewer search iterations than the state-of-the-art.

Although this evaluation shows that our approach is promising, both the *Ensemble* and *Causally-Assisted* approaches ultimately found faults in all initial datasets. This raises the question of how the approaches might fare on a system where it is more difficult to find faults. In Chapter 7, we describe a second evaluation using an APS, presented in Section 2.2. This is a particularly challenging example since it has a continuous input space, complex behaviour and a human-interacting environment.

Chapter 6.
Artificial Pancreas System Simulator

Before we can conduct our evaluation of an APS, we must first develop a simulation environment to enable surrogate-assisted testing. Unlike Pylot, our subject system oref0 does not have an existing simulator. In Section 3.5, we describe how a modern solution to such simulation environments is the use of a digital twin. In this Chapter, we develop a digital twin of a person interacting with oref0. The digital twin can then be used to observe the behaviour of oref0, providing the simulation-based evaluation step of surrogate-assisted testing (Section 3.4.3).

To ensure our digital twin is suitable, we validate this simulation environment through an evaluative case study. To achieve this, we implement the 5 steps for using a digital twin to test a medical device outlined by Corral-Acero et al. [37] described in Section 3.5.3. We note how this framework parallels configuration testing since specific configurations can be tested against the system to ensure correct behaviour. In Section 3.4.5, we also described how configuration testing can be applied to simulation-based evaluation.

6.1. A Digital Twin for Testing an APS

In Section 2.2.1, we discussed the challenge of human interaction when testing and configuring an APS. Digital twins have shown promise in using predictions to decouple testing from the human-in-the-loop in other domains [5, 37]. However, for blood glucose-insulin dynamics, only models are currently available [13, 34]. Such models do not implement T1DM blood glucose-insulin dynamics and do not interact with an APS through simulation.

To enable configuration testing in a safe environment, we require a digital twin that is capable of predicting user responses to clinical interventions. The key task is to provide an environment where new configurations can be trialled, without interacting directly with the user.

We use the 5 steps in the framework set out by Corral-Acero et al. [37], presented in Section 3.5.3, to guide this implementation. We first present an existing model of blood glucose-insulin dynamics and adapt it to represent people with T1DM (Step 2). We then devise a strategy to fit the large number of model constants required to model the compexities of blood glucose-insulin dynamics [34]. This allows the model to simulate real world blood glucose-insulin dynamics, personalised to the user (Steps 3,4). Using this model, we complete the digital twin by interfacing it with the oref0 APS algorithm (Section 3.3). As a result, we can observe how a person would be affected by different configurations of an APS without requiring clinical trials (Step 5). We explore Step 1 of the framework, gathering data, in Section 6.2.2.

Figure 11 illustrates our approach. The model of a person with T1DM represents the

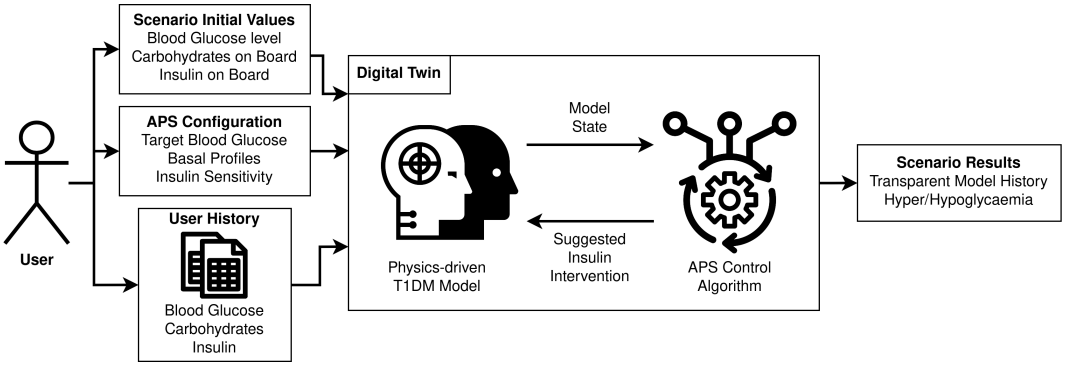


Figure 11.: The process of a digital twin of a person with T1DM using an APS simulating user provides scenarios for different APS configurations. A model is generated based on the user's historical data representing their blood glucose-insulin dynamics. This model communicates with an APS control algorithm, using the user supplied APS configurations, to simulate blood glucose dynamics for user generated scenarios.

blood glucose-insulin dynamics of a user by fitting its parameters based on their blood glucose, carbohydrate, and insulin history. The user then provides an initial blood glucose, carbohydrate and insulin value, henceforth referred to as a *scenario*, for which they would like to observe APS interaction over time with the model. The model periodically sends its current state to the APS and simulates any suggested insulin interventions. A user can explore different APS configurations, observing the impact of potentially dangerous scenarios within a simulated environment.

When classifying our proposed digital twin, we look back to the taxonomies described in Section 3.5.1. In order to infer the blood glucose-insulin responses of clinical intervention, our digital twin must be *predictive*. This allows for unseen states to be investigated, based on the physics-driven modelling. Our digital twin will also take the form of a *digital shadow* due to the limitations of data flow. To provide a simulation environment to enable surrogate-assisted testing of oref0, reintroducing data back to a physical device is not necessary. However, ensuring the blood-glucose-insulin dynamics are time-evolutionary (Section 3.5) is still required for accurate behavioural predictions and would, therefore, require up-to-date training data.

6.1.1. Modelling Blood Glucose-Insulin Dynamics

For our evaluative case study, we require an explainable model that models the blood glucose-insulin dynamics of a person. For this, we present work by Contreras et al. [34] in which they define a model for representing the blood glucose-insulin dynamics within a healthy body that is *not* affected by T1DM. This model uses differential equations to provide an explainable, physics-driven representation of blood glucose-insulin dynamics. The model is supported by an extensive sensitivity analysis [34] and has been used to inform methodologies in recent works [59]. This model is shown in Equation (6.1).

The differential equations of the model represent the process of carbohydrates being consumed and the subsequent insulin-glucose dynamics. The differential equations are used to

calculate the amount of carbohydrates in the stomach (S), Ilium (L) and Jejunum (J), as well as the blood glucose level (G) and insulin on board[1] (I). This allows for representation of carbohydrates as they move from the stomach and are absorbed across the Ilium and Jejunum in the small intestine. These equations show how carbohydrate absorption, insulin production and hepatic glucose production ($Gprod$) regulate blood glucose levels. The rate of these dynamics are represented by the remaining constants including kinetic constants (k_{js}, k_{xi}, etc), steady states (G_b, I_b), a time delay (τ), and insulin production scales (β, γ).

$$
\begin{aligned}
\frac{dS}{dt} &= -k_{js}S \\
\frac{dJ}{dt} &= k_{js}S - k_{gj}J - k_{jl}J \\
\frac{dL}{dt} &= k_{jl}\varphi(t) - k_{gl}L(t),\ \varphi(t) = \begin{cases} 0, & \text{if } t < \tau \\ J(t-\tau), & \text{if } t \geq \tau \end{cases} \\
\frac{dG}{dt} &= -(k_{xg} + k_{xgi}I)G + G_{prod} + \eta(k_{gj}J + k_{gl}L) \\
\frac{dI}{dt} &= k_{xi}I_b \left(\frac{\beta^\gamma + 1}{\beta^\gamma (\frac{G_b}{G}^\gamma + 1)} - \frac{I}{I_b} \right) \\
G_{prod} &= \frac{k_\lambda}{\frac{k_\lambda}{G_{prod0}} + (G - G_b)}
\end{aligned}
\tag{6.1}
$$

This model provides explainability by representing physiological behaviour with transparent mathematical equations. Figure 12(a) demonstrates a scenario in which this model is used to represent the glucose-insulin dynamics for a person with normal insulin sensitivity[2] consuming carbohydrates. Intuitively, a person with a lower insulin-sensitivity should have a higher blood-glucose level over time. We can simulate this by changing the insulin sensitivity constant in the model, the result of which is represented in Figure 12(b).

However, this model does not represent the blood glucose-insulin dynamics of a person with T1DM and is not a digital twin. In the following section, we take this model and adapt it to represent the dynamics of T1DM. We then interface it with oref0 to generate a digital twin which can be used to predict user responses to oref0 interventions.

6.1.2. Adapting the Contreras Model to Represent People with T1DM

To accurately represent the blood glucose-insulin of a person with T1DM, we first adapted the model proposed by Contreras et al. [34], presented in Section 6.1.1, to represent the physiology of T1DM. This can be seen as Step 2 of the Corral-Acero et al. [37] approach, generating a mechanistic model. We use this section to describe the required changes to the model so it can then be fit to real world data in the following section.

For our evaluative case study, we focus on the implementation of a mechanistic model to enable personalised predictions. Mechanistic models allow for prediction of unseen out-

[1] Insulin in the bloodstream.
[2] How much a person's blood glucose is reduced by an amount of insulin.

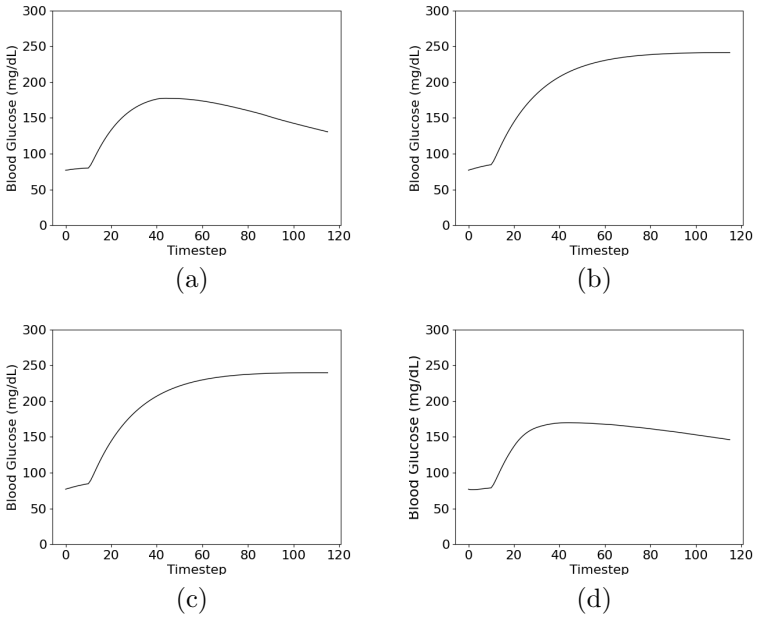


Figure 12.: Model outputs presenting blood glucose dynamics over time for an initial blood glucose, insulin and carbohydrate value (a scenario). (a) blood glucose dynamics as defined by Corral-Acero et al. [37] for a healthy pancreas. (b) presents the same scenario with suppressed insulin sensitivity. (c) presents the same scenario with 1% insulin secretion. (d) presents the same scenario with suppressed insulin sensitivity when interacting with oref0.

comes due to their encapsulation of human physiology [40]. Corral-Acero et al. [37] also propose the use of statistical models in digital twins. However, they are proposed to be used for the analysis of behaviours and for when the underlying behaviours are not well understood. Future work could be performed to evaluate the introduction of a statistical model to complement the mechanistic model but we find this out of scope for our simulation environment.

The adapted model makes the simplifying assumption that a person with T1DM does not produce any insulin. We illustrate this change in Equation (6.2). We set the insulin production term $\left(\frac{\beta^\gamma+1}{\beta^\gamma(\frac{G_b}{G}^\gamma+1)}\right)$ of this equation to zero. This allows the insulin steady state (I_b) to cancel out, resulting in the simplification of insulin dynamics to the rate of insulin degradation (k_{xi}).

$$\frac{dI}{dt} = k_{xi}\cancel{I_b}\left(\frac{\cancel{\beta^\gamma+1}}{\cancel{\beta^\gamma(\frac{G_b}{G}^\gamma+1)}} - \frac{I}{\cancel{I_b}}\right) = -k_{xi}I \tag{6.2}$$

In practice, some people with T1DM do produce a small amount of insulin and are known as 'micro-secretors'. Januszewski et al. [76] measured the concentration of insulin-producing beta cells in people with T1DM compared with a control group. They found that, on average, 55.3% of people with T1DM secrete insulin, but at a level less than 1% of the control. Figure 12(c) presents how there is no noticeable difference between 1% insulin secretion and no

insulin absorption, as in Figure 12(b). As such, secretion would not noticeably impact the adapted model's blood glucose-insulin dynamics.

As with the original model by Contreras et al. [34], we define a person's internal mechanics by the differential equations presented in Equation (6.3)[3]. As a person's internal mechanics are specific to them, we define the constants for these equations constants in Equation (6.4) as the set $K_x(T)$, where x is a specific person. This values of the constants in this set are bounded by the time period T since the internal mechanics change over time based on factors, such as the time of day and exercise [142]. The meaning of each constant can be found in Table 12 in Appendix B.1. By adapting the original model, we present a mechanistic approach (Step 2 of Corral-Acero et al. [37]) to calculate a person's blood glucose-insulin dynamics based on a set of personalised constants.

$$
\begin{aligned}
\frac{dS}{dt} &= -k_{js}S \\
\frac{dJ}{dt} &= k_{js}S - k_{gj}J - k_{jl}J \\
\frac{dL}{dt} &= k_{jl}\varphi(t) - k_{gl}L(t),\ \varphi(t) = \begin{cases} 0, & \text{if } t < \tau \\ J(t-\tau), & \text{if } t \geq \tau \end{cases} \\
\frac{dG}{dt} &= -(k_{xg} + k_{xgi}I)G + G_{prod} + \eta(k_{gj}J + k_{gl}L) \\
\frac{dI}{dt} &= -k_{xi}I \\
G_{prod} &= \frac{k_\lambda(G_b - G)}{k_\mu + (G_b - G)} + G_{prod0}
\end{aligned}
\tag{6.3}
$$

$$
\begin{aligned}
K_x(T) = \{&k_{js},\ k_{gj},\ k_{jl},\ k_{gl},\ k_{xg},\ k_{xgi},\ k_{xi}, \\
&\tau,\ \eta,\ k_\lambda,\ k_\mu,\ G_{prod0}\}
\end{aligned}
\tag{6.4}
$$

6.1.3. Fitting the Model to a Person with T1DM

Since blood glucose-insulin dynamics vary from person to person, the model will need to be tailored to an individual. To achieve this, a large number of constants ($K_x(T)$) must be set using historical blood glucose-insulin data. Contreras et al. [34] provides error equations but do not explicitly show how they can be used for parameter fitting. This equates to Steps 3 and 4 of the approach outlined by Corral-Acero et al. [37].

First, we take a single trace from our data source. The trace will be used as the training data for fitting. This trace represents the desired behaviour our model should represent after training. From this trace, we can extract the interventions applied to the human body. These include how much and when the user has eaten, if insulin was manually injected, and the insulin injected by oref0. Such interventions can then be applied at the relevant time steps to the model during training. For example, if the user injected insulin after consuming

[3] We also present a model correction to G_{prod} in Equation (B.1) in Appendix B.1

Table 4.: The configuration of PyGAD used when fitting our model. This was derived from preliminary experimentation.

Parameter	Value
Generations	1500
Solutions per Generation	30
Parent Selection	Elitism
Number of Mating Parents	4
Mutation Percentage	20%

a meal, the model would be trained using both of these extracted interventions since the blood glucose-insulin dynamics would represent the resulting behaviour.

We then use meta-heuristic search to find the values of $K_x(T)$ that best represent our desired behaviour. From this, we have fit our model, allowing for the blood glucose-insulin dynamics of the data source trace to be represented. The fitting process is available in our replication package (Section 1.1.2).

We use a genetic algorithm to minimise the error between the training data set and the model output for the parameters $K_x(T)$. We describe how this error is calculated later in this section. Genetic algorithms are meta-heuristic optimisation algorithms that use chromosomes to represent different solutions [83]. For this optimisation, we used the genetic algorithm PyGAD [60] as it provided an open-source implementation of a highly configurable optimisation algorithm. This is the same meta-heuristic search algorithm that we use in Chapters 5 and 7 to perform surrogate-assisted testing, as described in Section 5.1.4. The configuration of our genetic algorithm is presented in Table 4.

In our case, each chromosome is a configuration of $K_x(T)$ and the relative fitness of each chromosome is defined by the error of the resulting model compared to the training data. Low error chromosomes are assigned a high fitness and stochastically combined and mutated to search the solution space to optimise the configuration of $K_x(T)$. Other meta-heuristic algorithms were trialled for this evaluative case study and the results to those preliminary tests are available in Appendix B.2.

Our error function for model optimisation is the sum of the root mean squared error (RMSE) and a scaled oscillation error:

$$e_m(T) = RMSE + k_{osc}\ e_{osc} \tag{6.5}$$

To determine the accuracy of each chromosome, we calculate the root mean squared error (RMSE), presented in Equation 6.6. This provides an error value based on the difference between the model outputs ($z_m(t)$) and the observational training data ($z_o(t)$) across a given time period (T). RMSE provides an error that is always positive and does not increase with the number of model timesteps.

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=0}^{T}(z_o(t) - z_m(t))^2} \tag{6.6}$$

Differential equation models can introduce oscillations that are not possible in physiological systems [34]. To reduce unnatural oscillatory behaviour, we measure the oscillation error of the model using Equation (6.7). This value is scaled by an oscillation constant (k_{osc}) to reduce its impact on the overall error, allowing for physiologically correct oscillations when they existed in the data.

$$e_{osc} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(z_m(t) - z_m(t-1))^2} \tag{6.7}$$

Using the full error function $e_m(T)$ to drive a genetic algorithm, we generate configurations for $K_x(T)$ that best represent the observational training data. The training data set used in the fitting of our model was the OpenAPS Data Commons, presented in Section 6.2.2. We extracted blood glucose, insulin and carbohydrate values from this data to create a time series from which to train our model. This technique produces models that represent a person's historical blood glucose-insulin dynamics as illustrated in Figure 11.

6.1.4. Digital Twin of a Person Using an APS

Now we have generated a fitted model, we interface this with an APS control algorithm. As a result, we create a digital twin of a person with T1DM using an APS. We illustrate this as the interaction between the model and control algorithm in Figure 11. This interaction allows a user to simulate the exchange of data between their model and the APS to observe how APS interventions would affect their blood glucose levels over time. This equates to Step 5 of the approach outlined by Corral-Acero et al. [37] (Section 3.5.3).

We incorporate the oref0 control algorithm [130], as described in Section 3.3, as part of our digital twin to predict blood glucose-insulin dynamics when interacting with an APS. oref0 is a Javascript program made up of utility scripts for creating an APS pipeline invoked by shell scripts. Figure 13 presents a sequence diagram representing the pipeline used by our digital twin to invoke oref0 as well as the data required at each step. The constant exchange of information between the model and control algorithm creates a feedback loop that models the APS' effect on the person's blood glucose dynamics over time. This mimics the data flows from CGMs and insulin pumps to an APS control algorithm. The code for this exchange of data can be found within our replication package (Section 1.1.2). Following Figure 13, our digital twin interfaces with oref0 as follows:

1. The scenario runner fetches the current state of the blood glucose insulin model and then invokes an oref0 pipeline wrapper which manages the interaction with oref0 scripts. This passes in the model's current and past states into the wrapper, as reflected in Figure 11, and any oref0 configurations from the user.

2. The wrapper invokes the *oref0-calculate-iob* script passing in the data seen in Figure 13. This invokes oref0 to infer the insulin on board based on the state of the model for future calculations.

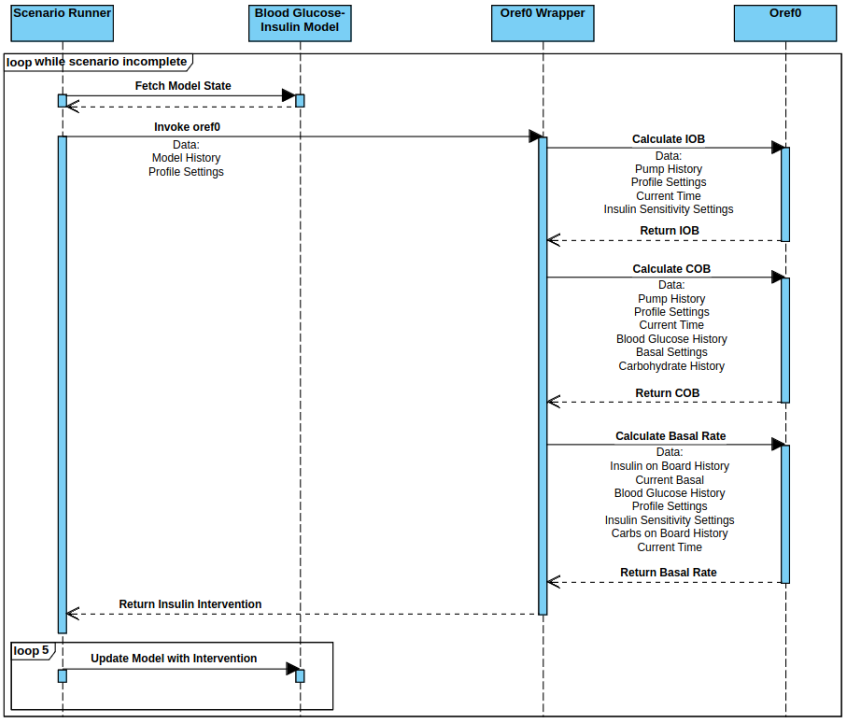


Figure 13.: A sequence diagram presenting how our model interacts with oref0 and the data required at each step. First, the insulin on board is calculated, then carbohydrates on board is calculated and finally the suggested insulin basal rate is calculated. This data is then fed back into the model to simulate an insulin pump supplying insulin.

3. The wrapper then invokes the *oref0-meal* script to generate a file containing all the carbohydrate on board[4] data required for future calculations. Figure 13 presents the data required for this step.

4. The wrapper calls a final script, *oref0-determine-basal*, passing in all the gathered and calculated data required. This script generates a suggestion for the insulin requirements of the user. The wrapper interprets this output and returns a value of insulin which should be "injected" into the model.

5. The scenario runner adds the suggested insulin to the model and then updates the model for the next 5 timesteps (5 minutes) to simulate an insulin pump injecting insulin over time. This process is repeated every 5 timesteps until the simulation concludes.

The transparency of the blood glucose-insulin model makes simulations explainable by allowing the internal state to be mathematically calculated for any timestep. Effects of an APS control algorithm intervention can be trialled in a controllable environment and traced back through the model without the need for physical trials, which may be expensive or hazardous to the user.

[4]Carbohydrates consumed.

An APS should aim to mimic a healthy pancreas as closely as possible. To test this, we refer back to Figure 12(a) of a person with a healthy pancreas. Figure 12(d) presents the same scenario replacing the original model with our digital twin. We observe a negligible difference between the healthy and APS-supported blood glucose curves, presenting oref0's proficiency in stabilising blood glucose levels.

6.2. Case Study Design

In this section, we present our case study to explore the suitability of using a digital twin for the simulation of APS behaviours. We also attempt to highlight any limitations and difficulties which may threaten future implementations. We follow the case study protocol outlined in Runeson et al. [148] as well as the ACM empirical case study standards [144]. First, we reiterate the motivations for the case study to justify our case selection. Then, we present the case study research questions that will drive our evaluation. From this, we are able to identify any data required. Finally, we outline any further measures required to answer the case study research questions.

As presented in Section 2.2.1, APSs present a challenging software environment within which to find correct software configurations, due to the human-in-the-loop. Incorrect configuration of the control algorithm can be potentially dangerous for the user. Section 3.5.3 highlighted how digital twins are a promising approach to reducing risk when configuring human-interfacing devices, but this technique has yet to be implemented.

We chose to evaluate our case study on the configuration testing of oref0. Section 3.3.1 presents how highly configurable the control algorithm is and the challenges and potential consequences associated with misconfiguration. oref0 also presents open-source data availability in terms of its source code and the OpenAPS Data Commons, outlined in Section 6.2.2. The remainder of this study presents a Healthcare 4.0 (described in Section 3.5.3) inspired case study to identify the benefits and challenges associated with configuration testing for an APS control algorithm using a digital twin.

6.2.1. Case Study Research Questions

In this section we present the case study research questions (CS-RQ). Note, CS-RQs are separate from the RQs defined in Chapters 5 and 7. Our aim is to evaluate the use of a digital twin in configuration testing of an APS control algorithm and its ability to successfully simulate the environment in which it is used.

CS-RQ 1 *To what extent can our digital twin provide a simulated environment for the configuration testing of an APS?*

For the validation of APS behaviour as a part of surrogate-assisted testing, the environment in which the APS is being simulated must be accurate. Here, we investigate whether the derived digital twin model is able to accurately represent the blood glucose-insulin dynamics of a person with T1DM. This will allow us to understand the effectiveness of adapting

the model described by Contreras et al. [34] for representing T1DM and any difficulties encountered with regards to configuration testing. We use this CS-RQ to evaluate Steps 3 and 4 of the approach outlined by Corral-Acero et al. [37], model calibration and model validation (Section 3.5.3).

CS-RQ 2 *To what extent can our digital twin make predictions to enable configuration testing for an APS?*

When observing expected configurations of an APS control algorithm, the predictions about its execution environment must also be accurate. From this research question, we investigate to what extent the adapted model's fixed parameters can accurately extrapolate unseen blood glucose-insulin behaviour of a person with T1DM as well as identify any challenges that occur when extrapolating blood glucose-insulin dynamics. We use this CS-RQ to further evaluate Step 4 of the approach outlined by Corral-Acero et al. [37], model validation.

CS-RQ 3 *Can digital twins be used to test the blood glucose target configuration of an APS system, without relying on humans-in-the-loop?*

After developing the simulated environment, we evaluate the behaviours and reliability of an APS control algorithm. We aim to observe the effectiveness of the APS control algorithm across multiple scenarios and configurations. We evaluate values of oref0's blood glucose target as, in Section 3.3.1, we present it as a setting which could produce potentially dangerous behaviour if misconfigured. We aim to investigate how the use of an explainable model as part of a digital twin allows for the tracing of medical interventions from different system configurations. This CS-RQ evaluates Step 5 of the approach outlined by Corral-Acero et al. [37], testing system configurations against predictions from the calibrated model.

6.2.2. Data Cleaning

In this section, we describe the data cleaning process used to find the candidate traces for this study. We first outline the data source for our case study, the OpenAPS Data Commons. We then present the steps required to process this data for use in our case study. Our data cleaning process equates to Step 1 of the approach outlined by Corral-Acero et al. [37], presented in Section 3.5.3. The data identified from this procedure is also used in the evaluation in Chapter 7.

OpenAPS Data

To enable the digital twin to make personalised predictions to the user during configuration testing, we use a historical clinical data set. The OpenAPS Data Commons [97] is the largest openly available APS data set. It contains approximately 10 million data points across 156 volunteers [152]. This dataset contains volunteered data from people using implementations of oref0 in the form of CGM, insulin pump, and control algorithm outputs.

However, we must first ensure the data retrieved from the OpenAPS Data Commons is compatible with our digital twin. Due to the intrinsic nature of medical data, the dataset may be missing values, have inconsistent formatting, or suffer from sensor error [52, 94]. This is especially important for data that is volunteered and not obtained through clinical trials as there may be less control in the data collection. The OpenAPS Data Commons, as with other medical datasets, suffer from these issues, requiring us to first pre-process the dataset.

Data Inclusion and Exclusion

To make the OpenAPS Data Commons data set compatible with the model fitting procedure, outlined in Section 6.1.3, the data must first be processed. The data must be transformed into blood glucose, insulin and carbohydrate time series with any traces with non-physiological behaviours or measurement errors excluded. We define our process for data cleaning as follows:

1. Exclude all traces that do not include blood glucose, insulin and carbohydrate data (which are required by the model). *This allowed us to exclude data sets that did not include all required historical data for training the digital twin.*

2. Exclude all traces that are not in a consistent format or contain null values. *This ensured our digital twin could correctly learn the blood glucose-insulin dynamics of a person with T1DM as a complete trace was required.*

3. Split up each trace into time periods of 2 hours with no carbohydrates on board before a single carbohydrates consumption. *The original data was in long continuous time series. The training procedure outlined in the original model by Contreras et al. [34] required 2 hours of training data after a single carbohydrate consumption. 2 hours of no carbohydrate consumption was required before each trace since prior consumption could result in blood glucose levels increasing from fac tors outside the training data.*

4. Exclude all traces that contain any large non-physiological increases or decreases in blood glucose. *This was used to remove traces which contain behaviours that should not be physiologically possible. This includes blood glucose changes over 50 mg/dL (2.8 mmol/L) in a 5 minute timestep or no absorption in carbohydrates over the trace. Such traces exhibit non-physiological behaviour and are therefore invalid. Such traces may exist in the data due to sensor error, human error in data collection or just invalid formatting as traces in the OpenAPS Data Commons were volunteered by users.*

Using these criteria, we processed appropriate traces from the OpenAPS Data Commons. We started with 156 original traces and ended with 930 time period traces after applying our procedure. These made up the candidate traces used throughout the case study. Our final traces were stored as 4 hour trace files with data points every 5 minutes.

The original data set contained 9423805 minutes (∼18 years) of data points. After applying our data cleaning process, we found that 223200 minutes (2.37%) of the data was suitable. This was due to model training requiring carbohydrate consumption, non-physiological

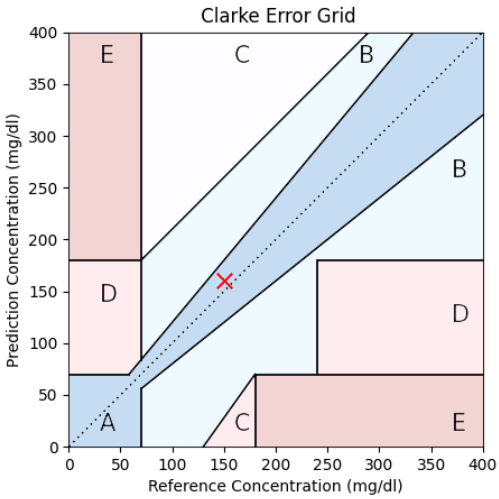


Figure 14.: An example of a Clarke Error Grid. A red X is used to denote a reading from a blood glucose monitoring device.

behaviour in the data or simply human error in capturing the data. The incredibly low acceptance of data highlights the difficulty of obtaining high-fidelity data, especially in a real time environment. Section 6.4.4 discusses how this is a potential threat to configuration testing of medical devices.

6.2.3. Measures

The effectiveness of an APS is typically evaluated by its user's time in range (TIR). Battelino et al. [17] define TIR as the "percentage of readings and time per day within target glucose range". This metric shows how effective an APS is at reducing hyperglycaemia (ensuring blood glucose stays below 180 mg/dL or 10.0 mmol/L) and not inducing hypoglycaemia (ensuring blood glucose stays above 70 mg/dL or 3.9 mmol/L).

Blood glucose monitoring devices also use Clarke Error Grids [30] to evaluate their accuracy and to determine the clinical significance of device outputs [111]. The results of the error grid are split into 5 zones, dictating the scale of error with respect to the correct blood glucose value: clinically accurate (A), clinically acceptable (B), over-correction (C), failure to detect (D) and erroneous (E) [89]. From this, a trained professional can assess the reliability of the blood glucose monitoring device.

Figure 14 presents an example Clarke Error Grid with a single reading. This reading shows a reference concentration (the true value) of 150mg/dL and the reading from the blood glucose monitoring device (equivalent to an output from our model) as 160mg/dL. The reading from the device is, therefore, slightly higher than the correct value. However, this reading is still classed as *clinically accurate* as the difference (distance from the dotted line) is small, meaning it falls within Zone A of the Clarke Error Grid.

6.3. Case Study Evaluation

In this Section, we present the evaluation of our case study. We describe the methodology used for each research question and then explore their findings. We further discuss these findings and their relation to the case study in Section 6.4.

6.3.1. CS-RQ1 - Model Validity

This CS-RQ evaluates the ability of a mechanistic model as part of a digital twin to represent mechanics of body dynamics. We equate this to an application and evaluation of Steps 3 and 4 of the testing procedure outlined in Section 3.5.3.

Methodology

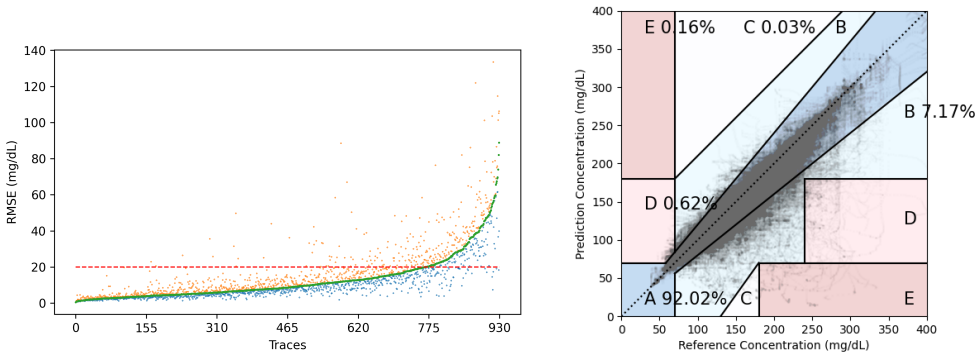
For this CS-RQ, we took the 930 clinical traces, generated from Section 6.2.2, to personalise the adapted model. These traces represent real blood glucose-insulin dynamics across 156 different people. Since the fitting process outlined in Section 6.1.3 is non-deterministic, we fitted each trace 10 times. We used the RMSE metric (Equation (6.6)) to quantify the difference between the observed data for each of the 930 traces and their corresponding fitted model outputs. We define an RMSE less than 20 as accurate as predicted values inside the safe blood glucose range would not result in severe hypoglycaemia or severe hyperglycaemia [17]. We use this metric to quantify the accuracy of a digital twin and, therefore, its ability to represent blood glucose-insulin behaviours.

We also measured the clinical accuracy on model executions using a Clarke Error Grid [30], described in Section 6.2.3. We use the 930 candidate traces as reference values to identify the zones generated by each of the personalised adapted models. From this, we were able to quantify the number of traces in the OpenAPS Data Commons for which a digital twin's simulation would lead to clinically accurate treatment.

We then identified traces with high and low RMSE values. Incorrect behaviours in the model were identified and the transparency of the internal variables used to find the causes of such behaviour.

To further evaluate the applicability of our adapted model, we also perform this methodology using other modelling techniques. More specifically, we compare our approach to a data driven explainable approach (a symbolic regressor) and a purely data driven non-explainable approach (a neural network). We replicated the methodology using the symbolic regressor and neural network to produce a distribution of RMSEs for each approach. The default settings were used for each modelling technique, except increasing the neural network's convergence iterations, which is explained in the findings.

A symbolic regressor is a model that represents the training data as a set of equations [8], similar to our model. However, unlike our approach, these equations are derived purely from the data, removing the need for an understanding of the underlying physiology. Symbolic regressors are also explainable since their resulting equations can be traced back, allowing a user to understand the causes of information flow within the model.



(a) (b)

Figure 15.: (a) RMSE values across the 10 runs of each trace. The median (green), first quartile (blue) and third quartile (orange) values are plotted. The red dashed line represents the maximum RMSE value for an accurate model. (b) A Clarke Error Grid showing model outputs across all 930 candidate traces. Trace blood glucose values are used as the reference and model outputs are used as the prediction.

A neural network is another data-driven approach that mimics the flow of information through neurons and synapses [3]. Similarly to the symbolic regressor, this approach learns its structure from the data so minimal knowledge of the underlying physiology is required. However, neural networks are not explainable. It is very difficult for a domain expert to understand the information flow through a neural network, making finding the causes of blood glucose behaviours difficult.

Findings

Figure 15(a) presents the distribution of RMSEs across each of the 930 traces. 82.5% of the traces produce a median RMSE less than 20. An RMSE of less than 20 would result in blood glucose levels within the safe glycaemic range (70 mg/dL or 3.9 mmol/L - 180 mg/dL or 10.0 mmol/L) not being misclassified as severe hyperglycaemia (over 200 mg/dL or 11.1 mmol/L) or severe hypoglycaemia (below 50 mg/dL or 2.8 mmol/L).

From these results, we show that the adapted model is able to accurately represent the blood glucose dynamics of a person with T1DM for most of our traces. Since there have not been any previous works on developing a digital twin to represent a person with T1DM interacting with an APS, we do not have a benchmark to compare this to.

We observe that the standard deviation of RMSE values for a single trace generally increases as the average RMSE increases. 640 (68.8%) of the traces had a low standard deviation of less than 10 (half of our accuracy value), which largely applied to those values with lower median RMSE.

Figure 15(b) presents a Clarke Error Grid for all 930 traces. 92.02% of the model outputs are within zone A of the error grid and are, therefore, clinically accurate. These data traces would lead to clinically correct treatment decisions [111]. As a result, we observed how the adapted model was able to clinically accurately represent 92.02% of the traces in the

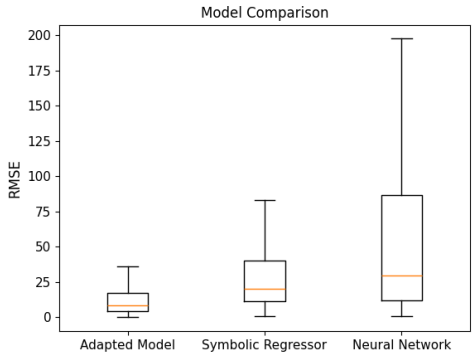


Figure 16.: Box plot presenting the distribution of RMSEs across all 930 traces for our adapted model, a symbolic regressor and a neutral network. Outliers have been removed.

OpenAPS Data Commons.

Although these results present the accuracy of our approach, we also assessed the applicability of our adapted model compared to other modelling approaches. Figure 16 presents the different RMSE distributions across the 930 traces for each different blood glucose-insulin modelling method. Our adapted model has the lowest RMSE, with the symbolic regressor being less accurate and the neural network being the most inaccurate.

The data driven techniques appear to struggle due to the nature of clinical data. These approaches lack the domain knowledge that makes up the 'template' for how blood glucose-insulin dynamics should behave. The symbolic regressor learned a simplification of the dynamics from the dataset, not taking into account factors such as the time delay of absorption across the Ilium and Jejunum, as described in Section 6.1.1. The neural network struggled to untangle the complex dataset, failing to converge for several traces even with 5 times the default number of convergence iterations.

Similar to our mechanistic approach, a symbolic regressor could be used to trace back through model traces. This explainability would allow a user to understand how different insulin interventions may have affected blood glucose levels. However, since there is no structure to the derived equations, a user may struggle to work out why a symbolic regressor has learnt specific behaviour. For example in Section 6.3.1, we identified that the insulin sensitivity of a model was incorrect from the k_{xgi} constant. Since the symbolic regressor does not have such a structure with specific constants for specific behaviours, understanding why specific behaviours are exhibited becomes more difficult.

The neural network is not an explainable model. Unlike the other two models, there is no practical way to examine why a neural network's inputs result in its outputs. As a result, a user would not be able to reason about the learned behaviour or trace back insulin interventions through the model to understand their cause.

The lack of explainability also made it difficult to determine the exact reason for the neural network's bad performance. Typically, factors such as over-fitting can be accounted for through splitting the data into test and training datasets [165]. However, our clinical dataset presented a couple of challenges regarding this. Each trace represented blood glucose-

insulin dynamics independent from the other traces. As a result, we could not use some traces for training and others for testing. Also, due to blood glucose-insulin dynamics changing over time, our traces were made up of 120 data points (every 5 minutes across 2 hours). The short traces made each data point necessary for training, meaning removing some for testing would have resulted in less accurate training.

Table 5.: The means and Shapiro-Wilk p-values for the distributions of data presented in Figure 16. Values are given to 5.d.p.

	Mean (Adapted Model)	Mean (Symbolic Regressor)	Mean (Neural Network)	Shapiro-Wilk p-value (Adapted Model)	Shapiro-Wilk p-value (Symbolic Regressor)	Shapiro-Wilk p-value (Neural Network)
Outliers Included	14.47301	15739.67252	876199045.27327	1.82872e-38	7.52140e-56	5.018337e-56
Outliers Removed	14.47301	27.82883	39.75195	1.82872e-38	2.654642e-35	8.7910162e-31

To further ensure that our evaluation of the distributions of data in Figure 16 demonstrates a significant difference between our approach and the others compared, we perform a statistical analysis. In this analysis, we assumed a p-value of less than 0.05 to be statistically significant [169].

Before applying our statistical tests, we must first determine which tests are applicable to our distributions. Table 5 presents the mean values and the p-value results of a Shapiro Wilk (SW) test [66] for each distribution. The SW test attempts to refute the hypothesis that a distribution follows a normal distribution. The resulting p-values can then be used to determine whether parametric statistical tests are suitable for this analysis.

Since the p-values for the SW tests across all the distributions are below 0.05, we can hypothesise that the distributions are not normal. Therefore, non-parametric statistical tests should be used when analysing these distributions [53].

We calculate these values with and without outliers. As per Figure 16, we define outliers as an RMSE greater than 200. Since some of the neural network and symbolic regressor executions failed to converge, we see a large disparity between including and excluding these outliers. Since we defined an accurate model to have an RMSE of less than 20, we assume that a model resulting in an RMSE of over 200 to not be viable.

Table 6.: Mann-Whitney U-test p-values and Cohen's D d-values for the distributions of results in Figure 16. Values are given to 5.d.p.

	Mann-Whitney U-test p-value (Adapted Model, Symbolic Regressor)	Mann-Whitney U-test p-value (Adapted Model, Neural Network)	Cohen's D d-value (Adapted Model, Symbolic Regressor)	Cohen's D d-value (Adapted Model, Neural Network)
Outliers Included	4.26850e-69	5.24559e-84	-0.06707	-0.04643
Outliers Removed	1.46850e-56	3.11923e-58	-0.58259	-0.80619

Table 6 presents the results of both Mann-Whitney U-test [110], henceforth referred to as a u-test, p-values and Cohen's D [32] d-values. These values were calculated comparing our adapted model to both the symbolic regressor and the neural network.

The u-test is used to present statistical significance between two distributions. Since the p-values from our u-test were less than 0.05, we can assume that there is a statistically

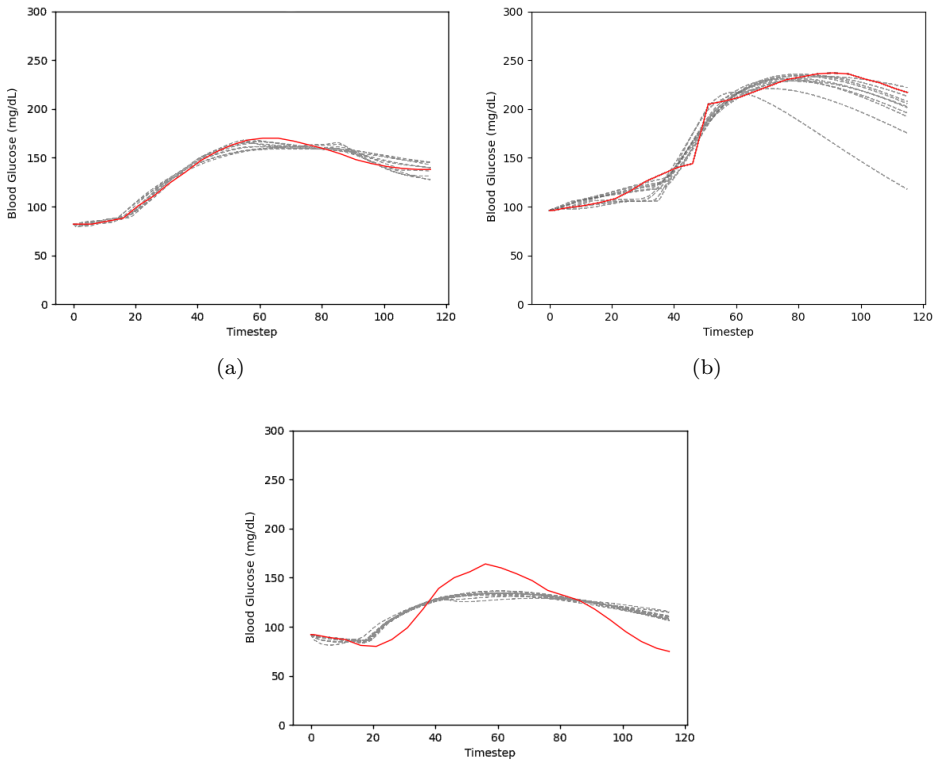


Figure 17.: Data traces showing fitted blood glucose dynamics. The real observational data from the clinical traces is shown in red, each of the 10 model attempts at learning are shown in grey. (a) presents a trace which all model attempts were able to represent, (b) presents a trace which some model attempts were able to represent, and (c) presents a trace which the model struggled to represent.

significant difference between the distributions. This significance emphasises the difference in distributions found in Figure 16. We did observe a difference between including and removing outliers in this test, but both results were statistically significant.

Cohen's D is used to measure the effect size between two distributions. For this test, we observed a large difference between including and excluding outliers. When including outliers, we found an insignificant effect between the distributions, since the absolute value was less than 0.2. However, by excluding the outliers, we observed a medium effect (absolute d-value of 0.5) when our approach was compared to the symbolic regressor and a large effect (absolute d-value of 0.8) when compared to the neural network.

This disparity in d-values is not surprising since calculating Cohen's D is largely dependant on distribution mean values. Since the outliers of both the symbolic regressor and neural network drastically increase the mean values of their respective distributions, the calculated d-values also greatly change.

To better understand the causes of accurate and inaccurate model dynamics, we can use the explainability of the adapted model. For this, we examine the RMSE values of model

outputs (grey dotted lines) with respect to their observational data trace (red solid lines), shown in Figure 17.

Figure 17 presents 3 examples of how the adapted model can accurately represent blood glucose dynamics as well as how that accuracy can vary. (a) presents a trace for which the model was able to represent the blood glucose dynamics across all 10 fitting attempts. This represents a trace which resulted in a low RMSE. For (b), the majority of the 10 attempts to fit the model accurately captured the blood glucose dynamics of the person with T1DM. One attempt, however, resulted in a large RMSE. We were able to trace this back through the model to find that the value for k_{xgi} was significantly larger than the other attempts. This caused the blood glucose level to decrease at a much greater rate than the observational data. (c), however, resulted in a large RMSE for all model fitting attempts. The model appears to over generalise the blood glucose dynamics, resulting in model outputs which do not follow the trace.

Summary: These results show how the stochastic elements of the fitting algorithm can allow for successful navigation of the solution space. However, non-determinism combined with the complexity of the solution space can sometimes produce sub-optimal solutions. Fitting the model a single time may not be optimal so the model should instead be fitted multiple times and the values of $K_x(T)$ that result in the least error used to simulate a user's blood glucose dynamics. In our case, we fitted the model 10 times as it presented accurate solutions for traces where not all model outputs had a low RMSE while not being too computationally expensive. This ensured that an accurate environment was simulated by the digital twin when performing configuration testing. Further work could be performed to understand the optimal fitting strategies and iterations required for this kind of problem.

Our results also found both of the data-driven approaches less suitable for modelling blood glucose-insulin dynamics than our adapted, domain knowledge-driven model. The underlying domain knowledge of blood glucose-insulin physiology of our approach simplified the task of learning personalised dynamics. The purely data-driven approaches did not have this underlying knowledge to provide a structure, resulting in less accurate models.

6.3.2. CS-RQ2 - Prediction Accuracy

This CS-RQ presents the ability for a digital twin to predict unknown body dynamics in order to aid in configuration testing. We use this CS-RQ to further evaluate the application of Step 4 in the testing procedure outlined in Section 3.5.3.

Methodology

For this CS-RQ, we only used traces in the lowest 50% of RMSE values from CS-RQ1, providing they had an RMSE ≤ 20 and are therefore shown to be accurate. This may have introduced bias into our results for this CS-RQ. However, traces that resulted in inaccurate models would have led to inaccurate predictions. Therefore, we chose to evaluate the predictive capabilities of our accurate models for this CS-RQ.

To investigate this CS-RQ, we trained the adapted model on the first 2 hours of each

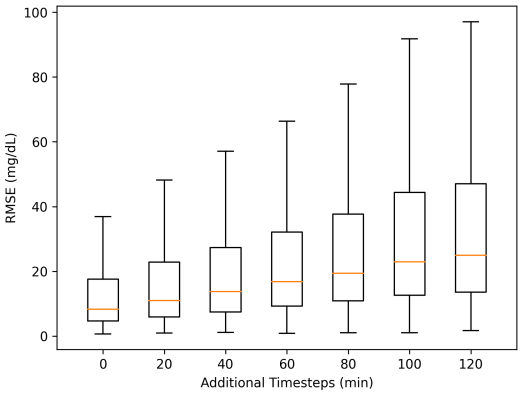


Figure 18.: Box plot presenting the distribution of errors as the level of extrapolation is increased across the 465 most accurate traces.

candidate trace. Then, we executed the model with additional timesteps in order to predict the remaining 2 hours of data points.

For each of the traces, we increased the additional timesteps from 0 to 120 minutes in steps of 20 minutes, measuring the RMSE at each. These additional timesteps are a continuation of the trained observational data. From this, we observed the accuracy of the model during extrapolation, providing insight into the reliability of the static set $K_x(T)$.

We used the RMSE values to find traces in which error did and did not increase with extrapolation. Using the transparency of the adapted model, we traced through these interesting extrapolation behaviours in order to find the causes of accurate and inaccurate predictions.

Findings

For this CS-RQ, we explored how blood glucose dynamics for a person with T1DM change over time. We used this question to understand to what extent our digital twin can predict blood glucose behaviour as the internal dynamics of the person change over time.

To answer this question, we used the RMSE metric to measure model accuracy when adding additional observed timesteps outside of the time bound captured by the training data. Figure 18 presents the distribution of RMSE across model outputs when extrapolating. From no extrapolation to 120 mins of extrapolation, the average RMSE increased from a value of 8.3 to 24.9. From this, we observed an increase in RMSE as the number of additional timesteps is increased. In CS-RQ1, we defined a prediction with an RMSE of 20 as accurate as it would ensure any incorrect predictions outside the safe blood glucose range are not severe. From this, Figure 18 presents how the adapted model can accurately predict 80 minutes of blood glucose dynamics outside the training data.

Similarly to CS-RQ1, we do not have a benchmark to compare the 80 minutes of accurate prediction to. However, for the traces that resulted in accurate models, this provides an insight into the applicability of the predictions used in configuration testing for our context.

To further explore blood glucose prediction, we identified models which originally had

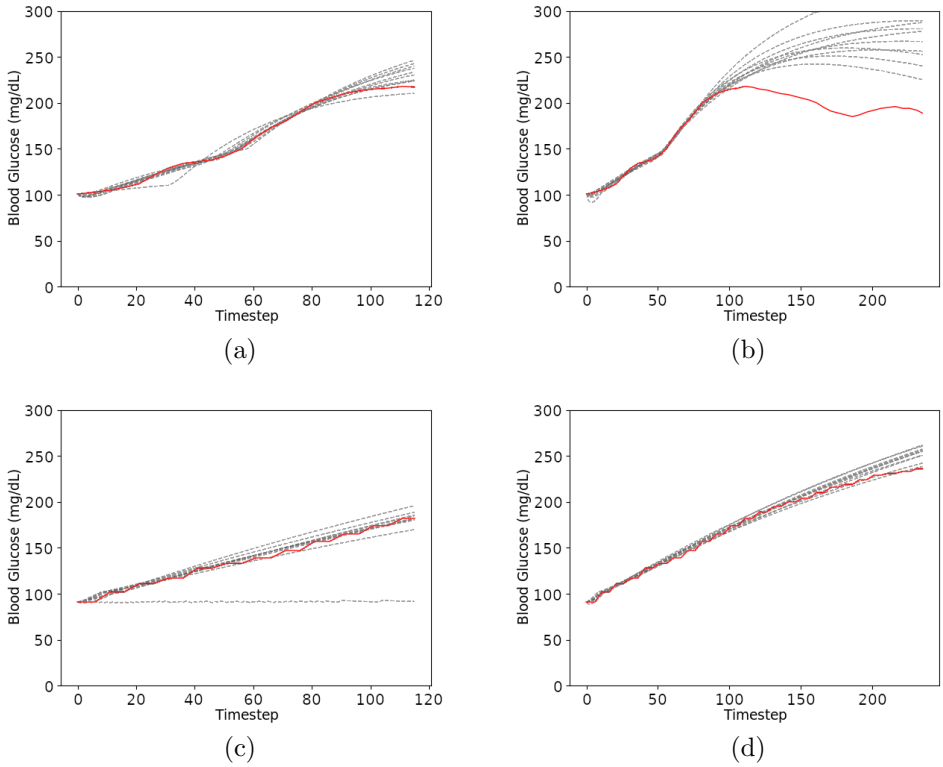


Figure 19.: Traces showing reliability of extrapolation. (a) and (b) present a trace showing an increase in RMSE due to extrapolation. (c) and (d) present a trace showing no increase in RMSE due to extrapolation.

low RMSE values but increased in RMSE after extrapolation. Figures 19(a) and 19(b) shows a single trace with 120 timesteps in Figure 19(a) and 240 timesteps in Figure 19(b). We use this example to observe the difference between no extrapolation and 120 minutes of extrapolation. Figure 19(a) presents a set of accurate model outputs where the blood glucose dynamics were correctly captured by the model. This led to a low RMSE in CS-RQ1. Figure 19(b) shows behaviour being extrapolated, which no longer followed the observed data. The trend of the behaviour was captured, but the nuances were not.

By splitting up this trace and only training on the second half, we were able to observe a completely different set of $K_x(T)$ to the original trace. This represented a change in blood glucose behaviour. The transparency of the adapted model allows for these sets of constants to be compared.

Some traces, however, did not increase their RMSE through extrapolation. Figures 19(c) and 19(d) presents a set of traces in which the extrapolated behaviour matched that of the trace data. Similar to Figures 19(a) and 19(b), the original trace for CS-RQ1 had a low RMSE. Since the blood glucose dynamics did not fluctuate during the extrapolation, this resulted in a low RMSE for Figure 19(d).

Training the model on both the first and second halves of this trace produced very similar $K_x(T)$ values. Since this set did not change much, the adapted model was able to accurately

predict the untrained blood glucose behaviour. From this, we observed that the adapted model's prediction is more accurate given stable blood glucose dynamics.

Summary: From these results, we suggest that the adapted model can be used to predict blood glucose dynamics up to 80 minutes outside of the time bounded by the training data. For continuously accurate representation, the values of $K_x(T)$ should be re-learnt outside this boundary to ensure accurate predictions. This would ensure predictions made by the digital twin would produce accurate and reliable behaviour for configuration testing.

6.3.3. CS-RQ3 - Configuration Testing

This CS-RQ aims to evaluate whether our digital twin can facilitate configuration testing of the blood glucose target setting of oref0 without clinical trials. We use this to evaluate the application of Step 5 of the testing procedure outlined in Section 3.5.3.

Methodology

We interfaced the model of a person with T1DM with oref0 to observe APS behaviour. To apply configuration testing, we need to select a small number of expected configurations, as described in Section 3.1.3. With this in mind, we identified 3 different blood glucose targets for oref0 which were widely used within the OpenAPS Data Commons: 100 mg/dL (5.6 mmol/L), 120 mg/dL (6.7 mmol/L) and 140 mg/dL (7.8 mmol/L). Section 3.3.1 identifies how the misconfiguration of blood glucose targets can result in dangerous scenarios. For this CS-RQ, we test these expected configurations with all other oref0 configurations unchanged.

For each of the 930 candidate traces, we fitted the model to represent the blood glucose dynamics for that given scenario, as seen in CS-RQ1. Given these dynamics and the initial carbohydrate, blood glucose and insulin values found in the trace, we removed the original insulin interventions and simulated the effects of using the different oref0 configurations for that scenario. Each configuration of oref0 was executed 10 times, allowing for any non-deterministic elements. Since the body does not react immediately to the interventions of oref0, the full 4 hour traces, as used in CS-RQ2, were required to capture the resulting behaviour.

For this analysis, we used the TIR metric, outlined in Section 6.2.3, to present the efficacy of each oref0 configurations for a given scenario. Battelino et al. [17] defined TIR as a percentage of time spent in a blood glucose range of 70–180 mg/dL (3.9–10.0 mmol/L). From this, we performed configuration testing on oref0 and evaluated each configuration based on its resulting TIR.

For this CS-RQ, we used oref0 0.7.1, which is not the same as that found in the OpenAPS Data Commons. To compare the performance of this version to the control algorithm in the original data, we compared the TIR to two other implementations: the original interventions in the training data, and the learned blood glucose-insulin dynamics without insulin intervention. The TIR from the original data can be calculated from the original data, finding the percentage of time within the safe glycaemic range for all of the original OpenAPS Data Commons traces. For no insulin interventions, we use the adapted model to learn the blood

Table 7.: Mean time in range (TIR) for different interventions across each 930 traces to observe oref0 effectiveness.

Intervention	BG Target (mg/dL)	Mean TIR (%)
No Intervention	N/A	63.08
OpenAPS (2014-2021)	Various	73.57
oref0 (v0.7.1)	100	87.69
oref0 (v0.7.1)	120	89.06
oref0 (v0.7.1)	140	88.10

glucose-insulin dynamics of the same traces. This time, we execute the model without the coupling to oref0, allowing for uncontrolled blood glucose-insulin dynamics to be observed.

Findings

Table 7 presents the TIR across different oref0 configurations averaged across each candidate trace. No intervention presents the TIR across after using the adapted model to learn the blood glucose-insulin dynamics of the original traces and removing all insulin interventions so that no insulin is provided to the user. OpenAPS (2014-2021) provides the TIR of each of the original traces in the OpenAPS Data Commons (preserving the original insulin interventions). oref0 (v0.7.1) provides the TIR observed when learning the blood glucose-insulin dynamics from the original traces and then using the digital twin to interface with oref0 across different blood glucose target configurations.

"No intervention" provided the lowest TIR across each implementation. This was not surprising as people with T1DM struggle to naturally regulate their blood glucose levels [76]. The high mean TIR was observed to be due to most of the traces starting within range, increasing the TIR for uncontrolled blood glucose.

"OpenAPS (2014-2021)" presented how controlling blood glucose levels increases TIR. This represents the increase in quality of life found by Litchman et al. [100] when using OpenAPS. Comparing this to no interventions is an unfair comparison, so we use this metric to compare the performance of older versions of oref0, present in the OpenAPS Data Commons, with the latest version at time of writing (oref0 v0.7.1).

"oref0 (v0.7.1)" provided a significant increase in mean TIR compared to older versions of oref0. From this, we were able to show an increase in user safety when using more modern APS control algorithms. The difference in average TIR between the different oref0 blood glucose configurations, however, was not noticeable.

To explore oref0 configurations further, Figure 20 presents two scenarios that show the behaviour of oref0 interventions across different blood glucose targets. For both of these scenarios, a large amount of carbohydrate ($\geq$60g) was consumed causing uncontrolled blood glucose levels to rise out of the safe blood glucose range, defined in Section 6.2.3. We used the explainability of the adapted model to explore how oref0 adapts to these scenarios.

Figure 20(a) presents a scenario in which oref0 successfully controlled the user's blood glucose within a safe range. From this, we observed how the different configurations resulted in different resulting blood glucose behaviours. Depending on the scenario, such as exercise,

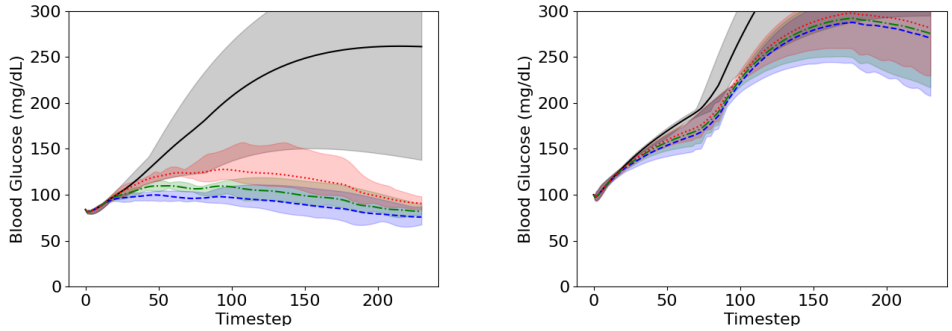


Figure 20.: Trace showing the effectiveness of oref0 against uncontrolled blood glucose. No intervention (black solid), oref0 targeting 100 (blue dashed), 120 (green dash dotted), 140 (red dotted). The mean lines are plotted for 10 runs with the ranges shown.

different targets and glucose behaviours are required. In this scenario, we see how the target of 140mg/dL sometimes produced undesirable behaviour, allowing the blood glucose to rise above a safe level. Our digital twin allowed for these configurations and their variability to be trialled in a safe environment.

Figure 20(b) presents a scenario in which the TIR was low for all configurations. We can see from the graph that there is minimal difference in system behaviour between the configurations. By investigating the model execution at each timestep, we found that the APS algorithm was acting correctly, providing insulin as expected to try and lower blood glucose levels. However, the model's fitted blood glucose dynamics were inaccurate. A lack of granularity of the insulin data in the observational trace led to the model consistently learning an infeasibly low insulin sensitivity value (k_{xgi}). As a result, the insulin interventions provided by oref0 had virtually no effect on blood glucose levels, making the insights gained from this configuration testing being misleading.

To further evaluate the applicability of configuration testing, we performed a temporal analysis of each stage of the testing procedure[5]. Figure 21 presents the different time expense distributions for each stage of the configuration testing process across the 930 traces. We do note that the times found in this evaluation are those required for 4 hours of oref0 execution.

These results illustrate how the training of the model is the most computationally expensive task of the procedure. Due to the large number of constants required when fitting the model, as described in Section 6.1.3, this is not surprising. However, the results of CS-RQ1 have shown the efficacy of this procedure in ensuring accurate models. Also, compared to the 4 hours of oref0 execution, the time required is still minimal

.

Figure 21 shows a large disparity between executing the model with and without oref0. The model execution alone took on average 0.002 seconds, whereas also executing oref0 took

[5]Since we are measuring time in this evaluation, it should be noted that the results are subjective to the machine the evaluation was executed on. Regardless of hardware, the ratios between the execution timings should be similar. This process is mainly CPU intensive so we note the CPU hardware: 2nd generation AMD EPYC 7R32 with 96 vCPUs

Figure 21.: Distribution of times taken for fitting the model, executing the model and executing oref0.

on average 38.7 seconds. This difference demonstrates the computational efficiency of using a differential equations model compared to the rest of the testing procedure. The difference also demonstrated the computational expense of running an APS. We observed how the expense of running the APS is comparable to fitting the model itself.

Summary: From this CS-RQ, we were able to evaluate the TIR of different settings of an APS control algorithm through configuration testing. Configurations could be trialled for different people to ensure an APS control algorithm is set up correctly, without the need for clinical trials. We also observed the limitations of using real world data, which will be explored further in Section 6.4.4.

We observed that fitting the model was the most computationally expensive part of the testing procedure. We also demonstrated the efficiency of our model and the extent to which interacting with oref0 increased the temporal expense required for configuration testing.

6.4. Case Study Discussion

In this section, we discuss the finding of our case study to evaluate the applicability of using a digital twin to evaluate an APS while decoupled from a human-in-the-loop. We use the SWOT analysis framework [96] to approach this discussion from the perspective of strengths, weaknesses, opportunities and threats.

6.4.1. Strengths

Medical devices present a challenge for configuration as humans in the loop can result in dangerous scenarios when the system is misconfigured. During our case study, we were able to develop a digital twin to alleviate this difficulty and observe executions of an APS control algorithm without putting users at risk.

The process outlined by Corral-Acero et al. [37], presented in Section 3.5.3, provided a framework for enabling the configuration testing of medical devices without requiring clinical trials. By following this process, we were able to develop a digital twin to represent

individuals and interface it with an APS control algorithm to observe different APS behaviours for different system configurations across a multitude of people. From this, we were able to perform configuration testing for the oref0 control algorithm for scenarios that were potentially dangerous, without putting any users at risk. We propose that the process outlined by Corral-Acero et al. [37] presented a compelling framework for other medical device configuration testing challenges.

6.4.2. Weaknesses

We did, however, encounter some difficulties when applying configuration testing to an APS through the use of a digital twin. Such a complex physiological phenomenon as blood glucose-insulin dynamics required an equally intricate model. CS-RQ1 found that the complex solution space of the adapted model sometimes presented a challenge when fitting it to data.

We further encountered this in CS-RQ2 where the digital twin sometimes struggled to predict more dynamic blood glucose-insulin behaviours. We suggest that this may be a sign of model over-fitting as our explainable model could not adapt its dynamics. However, more adaptive models, such as that presented by Edington et al. [47], present a potential solution to this challenge by generating a digital twin comprised of multiple models. A combination of data-driven and physics-driven models allow for a better representation of systems whose internal dynamics change over time.

6.4.3. Opportunities

The above weakness provides an opportunity for evaluating different modelling techniques for body dynamics. Future work could investigate the use of models that can better represent dynamics behaviours, such as machine learning based modelling [172], and how this could potentially improve the reliability of digital twin-based configuration testing in this context. Such an approach, however, would have reduced the explainability of the model and introduced the social challenge, presented by Corral-Acero et al., of users finding it difficult to trust a 'black-box' for clinical decisions [37].

Curated data sets from future work would allow for more usable data [85]. The OpenAPS Data Commons, however, demonstrates the unique challenge of the APS's clinical setting as such curated data does not typically exist. As a result, we found that only 2.37% of the data being suitable as a significant challenge concerning future evaluations. In which case, we suggest that more open-source datasets are made available from future clinical trials designed specifically for learning blood glucose-insulin dynamics. However curated datasets may reduce the realism of applying such techniques to a real user's data due to the intrinsic difficulties that come with using clinical datasets, as described in Section 6.2.2.

6.4.4. Threats

We found that the greatest threat posed to configuration testing using a digital twin was the use of clinical data. In Section 6.2.2 we present the OpenAPS Data Commons as our

data source. At the time of writing, this was the largest and best documented data set for people with T1DM [152]. Regardless of this, a large quantity of rejected traces led to only a 2.37% acceptance rate, as presented in Section 6.2.2. This was due to the data containing inconsistent or non-physiological behaviour, human error in data collection or not containing a recorded carbohydrate consumption that was required for training the model.

These factors are real characteristics of clinical data. Similarly to APSs, data for other medical devices may contain similar characteristics and, therefore, reduce the reliability of applying configuration testing using past observational data. Performing future evaluations for both APSs and other medical devices may be affected by these data challenges, reducing the ability to evaluate the generalisability of our approach.

A potential solution for increasing the number of accepted traces would be to interpolate missing values in the data. As a result, more of the data would be compatible with training the digital twin. Therefore, more behaviours could be trained, allowing for configuration testing of more contexts. However, this could introduce a potential threat. Interpolated values could result in behaviour that is not representative of the person's real glucose-insulin dynamics. Subsequent configuration testing may be impacted by using training data and, as a result, may be less reliable.

Even after rejecting 97.63% of the individual data points, some accepted traces did not contain enough granularity in the data, leading to misinterpretation of blood glucose-insulin dynamics (as seen in CS-RQ3). This meant other human factors in the data may have still introduced erroneous data into the model. This would produce potentially incorrect predictions from the model, which could lead to inaccurate assumptions about APS behaviours.

For digital twins to accurately predict the consequences of medical interventions, more accurate and consistent data sources or techniques to account for pre-existing data sources are required. Being able to account for the clinical data would allow for more models to be generated and an increase in their accuracy. By alleviating the challenges associated with clinical data, the applicability for modelling and testing medical devices would greatly improve.

6.5. Chapter Summary

In this chapter, we described the development of a digital twin of a person with T1DM interacting with an APS. We developed a model of a user's blood-insulin dynamics by adapting an existing model, trained it on a clinical dataset and coupled it with an APS. Our digital twin enabled us to investigate the behaviour of different APS configurations in potentially dangerous scenarios while being decoupled from the user. We validated the resulting digital twin through an evaluative case study, showing it to be clinically accurate to the original training data, able to perform predictions outside of the training data, and able to use predictions to enable configuration testing without interacting directly with the user.

Our case study also revealed the challenges associated with pre-existing clinical datasets. A large amount of the OpenAPS Data Commons was incompatible with our approach due

to missing values and a lack of consistent data capture. As a result, we note the difficulties associated with the intrinsic nature of clinical data during testing. In Section 2.2.1, we motivated this thesis by stating how curating datasets for system testing may not always be feasible, and how pre-existing datasets may be used instead. In the following chapter, we show how our causal surrogate model is able to alleviate some of the challenges associated with this dataset.

At the start of this chapter, we likened the use of configuration testing using a digital twin to the simulation-based evaluation step of surrogate-assisted testing (Section 3.4.5). In the following chapter, we follow this similarity by using our digital twin as a simulation environment for the surrogate-assisted testing of the APS, oref0.

Chapter 7.
Evaluation - Testing the Artificial Pancreas System

For the second evaluation, we explore the potential violations of the requirements for the artificial pancreas control algorithm, oref0 [98]. We have shown how surrogate-assisted testing (Section 3.4) and our digital twin simulation environment (Chapter 6) alleviate the *human-interaction* and *complex input space* testing difficulties of CPSs. In Section 2.2.1, we highlighted two further difficulties related to testing an APS: unmeasurable external factors, making it difficult to define correct behaviour, and the unavailability of curated datasets. As a result, finding incorrect system behaviours becomes difficult.

In this chapter, we build upon the evaluation in Chapter 5 by investigating further how our approach alleviates these difficulties. To facilitate this, we show that our approach can reduce bias in pre-existing datasets and answer causal questions about the relationships between system variables. We investigate the impact of using both causal and associative surrogate models to explore system behaviour during surrogate-assisted testing and the effects of the resulting combined dataset (described in Section 4.1.1). We also perform statistical tests on the results of our evaluations to investigate the significance of our results.

Specifically, we aim to answer the following research questions:

RQ2: To what extent can using causal surrogates increase the effectiveness of surrogate-assisted CPS testing?

RQ3: To what extent does using causal surrogates affect the search procedure's ability to explore system behaviours during surrogate-assisted CPS testing?

RQ4: To what extent can using **only** a causal surrogate model increase the effectiveness of surrogate-assisted CPS testing?

7.1. Evaluation Methodology

We this evaluation, we follow the same technique used for the ADS evaluation in Section 5. The remainder of this section outlines the requirements for surrogate-assisted testing of oref0, the baseline surrogate model we compare our approach to, and the metrics required to answer the RQs.

As with our initial evaluation, we follow Figure 7 for our approach. We described our methodology for evaluating surrogate-assisted CPS testing in Section 5.1. We provide a summary of this methodology as follows:

To perform surrogate-assisted testing, we require initial datasets, a causal DAG to specify

domain knowledge, a set of system violations, and a simulation environment. We alternate between generating an associative surrogate model and a causal surrogate model with each testing iteration. The chosen surrogate model is then used as a fitness function for a meta-heuristic search algorithm to find the scenario most likely to cause a system violation. This scenario is then evaluated on a high-fidelity simulator to observe if a violation occurs. If no violation occurs, this data is fed back into the original dataset and used to retrain the surrogate models as the testing procedure is repeated.

Algorithm 2 describes how the causal surrogate models are generated and Algorithm 3 describes how they are used to search for inputs for which causal relationships do not hold. The code used in this evaluation can also be found in our replication package (see Section 1.1.2).

This evaluation did require an additional step since the simulator needed to be configured based on the data. Since each dataset represented different blood glucose-insulin dynamics, the simulator was trained to represent these dynamics from the dataset before the testing procedure. We further discuss this additional evaluation step in Sections 7.1.1 and 8.5.

For RQ2, similarly to RQ1, we execute each testing technique to determine its effectiveness in finding system violations for each initial dataset. We determine whether our technique can identify scenarios in which SUT violations occur that are not detected by traditional approaches.

For RQ3, we investigate how the surrogate model affects the generated test inputs that are fed back into training future surrogate models (see Figure 7). Since our evaluation of oref0 requires a larger number of search iterations than in the ADS evaluation in Chapter 5, we are able to observe how the combined dataset (described in Section 4.1.1) evolves over the testing procedure. As a result, we investigate how the surrogate model affects the exploration of system behaviours and the resulting distribution of training data throughout the testing procedure.

For RQ4, we perform the same evaluation as in RQ2 but also evaluate the surrogate-assisted testing approach with *only* the causal surrogate model. As a result, we can determine to what extent the causal surrogate model can improve surrogate-assisted testing effectiveness without the help of an associative model and combined dataset. We can also assess the extent to which our hybrid approach relies on its associative surrogate model compared to the causal surrogate model.

7.1.1. Simulator and Violations

To enable the simulation of oref0 in scenarios of interest, we use the digital twin described in Chapter 6. For this evaluation, we personalise the digital twin to the blood glucose-insulin dynamics of the initial trace using the fitting procedure described in Section 6.1.3.

We define violations for oref0 as scenarios that result in severe hypoglycaemia or severe hyperglycaemia. Since these conditions could be dangerous to the user [81, 150]. These scenarios could occur if the APS behaves incorrectly by providing too much insulin (severe hypoglycaemia) or too little insulin (severe hyperglycaemia), allowing the user to enter an

unsafe blood glucose range. We define severe hypoglycaemia as a blood glucose level of below 50mg/dL (2.8 mmol/L) and severe hyperglycaemia as a blood glucose level of above 200mg/dL (11.1 mmol/L) [17].

7.1.2. Datasets

We evaluate our testing technique using pre-collected data from the OpenAPS Data Commons [41, 97, 153]. Unlike the pseudo-random generation performed in the ADS evaluation in Section 5.1.2, this evaluation uses real pre-existing data with approximately 10 million data points across 156 real users.

We performed the data cleaning process outlined in Section 6.2.2 [158] to obtain 924 initial datasets that could be used with the digital twin. Each dataset contains blood glucose, insulin and carbohydrate data points at 5-minute intervals for 2 hours. This selection provides a large variety of blood-insulin dynamics from real-world data that invoked different APS behaviours.

We use 924 datasets in this evaluation, as opposed to the 930 used in Chapter 6. 6 of the original traces resulted in blood glucose-insulin dynamics in which there was no possibility that the APS could stop the person from leaving the safe glycaemic range [17]. This was due to limitations in the data, identified in Sections 7 and 8.6, such as not enough insulin absorption information leading to the simulator learning an insulin sensitivity so low that oref0 interventions had no effect. We decided to remove these traces as they would have resulted in system violations, regardless of the technique. We further discuss this in Section 8.5.

From our 924 traces, we used the digital twin, described in Chapter 6, to learn the blood glucose-insulin dynamics of the user and extend each dataset. For each trace, we used the digital twin to predict 300 different scenarios for these blood glucose-insulin dynamics. Each scenario was generated from a set of input variables which were randomly sampled in accordance with the values in Table 8. As a result, we had 924 datasets, each representing that specific user's behaviour across 300 randomly selected scenarios.

As part of this data generation, we ensured that the APS had a chance of preventing hyperglycaemia or hypoglycaemia. For this, we removed scenarios in which the person started with too much injected insulin. Since the APS can only increase blood insulin levels, these scenarios resulted in a decrease of blood glucose levels to an unsafe level regardless of APS behaviour. If a scenario with this behaviour was generated, it was discarded and re-executed with a maximum of 100 reruns before the trace was discarded. This process uncovered the 6 traces for which the APS has no possibility of controlling the blood glucose levels, as described earlier in this section.

Table 8 presents the scenario input variables, output variables, and their potential values. Each scenario consisted of 3 input variables and 3 output variables. This produces datasets inline with those presented in our previous evaluation (Section 5.1.2) and Haq et al. [70]. The scenario consisted of initial values for blood glucose, carbohydrates and blood insulin, taken from the original trace. The amount of insulin prescribed to the user throughout the

Table 8.: The input variables and potential violation variables for oref0. Each input variable has a range that it must be in. The output variables do not have specified ranges, but are output as floats.

Variable Name	Potential Values
start_bg	[70 - 180]
start_cob	[100 - 300]
start_iob	[0 - 150]
open_aps_output	Float
hyper	Float
hypo	Float

scenario is calculated from the trace (*open_aps_output*). We also note the maximum (*hyper*) and minimum (*hypo*) blood glucose values from the trace for use in the associative surrogate model. From these values, the surrogate models can provide a simplified representation of the digital twin to be used as part of the meta-heuristic search.

As with Section 5.1.2, the same 924 datasets were used across each of the testing techniques in this evaluation. Each technique was also executed with the same initial random seed. As a result, the only difference between each execution was the testing technique.

7.1.3. Causal DAG

To use our causal surrogate model we must first generate a causal DAG. As with our previous evaluation in Section 5.1.3, generating this DAG is non-trivial. We use the datasets identified in the previous section when identifying the nodes and expected causal relationships of this causal DAG.

Figure 22 presents the causal DAG for oref0 in a simulation environment. We generated this DAG by identifying variables visible in the pre-existing datasets (see Section 3.6.1) and used domain knowledge to represent how they should affect one another. This DAG illustrates the causal relationships within oref0 (solid edges), the interactions of the APS on the simulator (dotted edges), and the potential relationships introduced by the search algorithm (dashed edges).

We constructed the DAG by using domain knowledge of the APS and its operating environment. We assume that the initial blood glucose (BG), carbohydrates (COB) and insulin (IOB) should affect the algorithmic output of oref0. We also assume these initial values will be affected by the search process, as with the previous evaluation. The initial values and the algorithmic output of oref0 should affect the likelihood of hyperglycaemia (Max BG) and hypoglycaemia (Min BG).

For the purposes of testing, we are only interested in the causal relationships within the SUT, which in this case are the relationships that cause a change in the amount of insulin prescribed by the APS control algorithm. It is still important to include the other edges in the DAG as they can still aid in mitigating spurious associations. Since the data for these variables are available in our dataset, we can use causal inference techniques to adjust

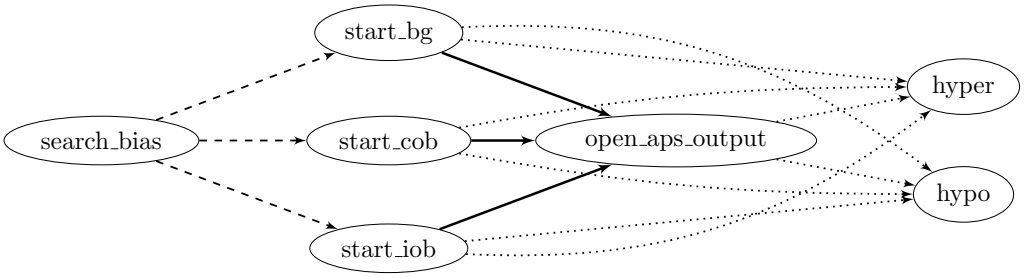


Figure 22.: Causal DAG of oref0. We highlight 3 causal relationships of interest (solid lines) and also include relationships present in the search procedures (dashed lines) and the simulator (dotted lines).

estimations based on the entire causal structure.

For the APS to work as intended, we expect that increasing the initial blood glucose (BG) or initial carbohydrates on board (COB) should increase oref0's insulin output. Conversely, we expect that increasing the initial insulin on board (IOB) should require less insulin to be output from oref0. By using our surrogate model to find scenarios that contradict these expected causal relationships, we aim to identify incorrect system behaviour that will potentially result in system violations.

7.1.4. Baseline

We compare our approach against the same ensemble surrogate technique described in Section 3.4.1 and our previous evaluation (Section 5.1.4). We also present the results of the random search to provide a testing baseline across 924 different random seeds, matching the number of initial datasets. We used the same meta-heuristic algorithm, PyGAD [60], and settings, as described in Section 5.1.4 and Table 2.

We imposed a computational budget of 200 search iterations. Unlike the ADS evaluation in Chapter 5, prior work has not been performed for finding violations in an APS using surrogate-assisted testing. Therefore, we did not have prior results from which to choose a computational budget. As a result, we used preliminary experiments to find a plateau in initial violations to devise our computational budget. These experiments were performed by observing the search iterations at which initial violations were found for a subset of the initial dataset. As a result of these experiments, we identified when each technique plateaued and subsequently derived a computational budget.

For this evaluation, we required a larger computational budget to find system violations. The original ensemble would have required ∼90 days of execution time, which was not feasible in our time budget. As the result of some preliminary experiments (see Appendix A), we chose to remove the neural network. This did affect the accuracy of the ensemble model slightly, but we determined it was worth the cost. To ensure this adaptation was fair, we mirrored this change in both the Ensemble surrogate model and the associative surrogate model of the Causally-Assisted technique.

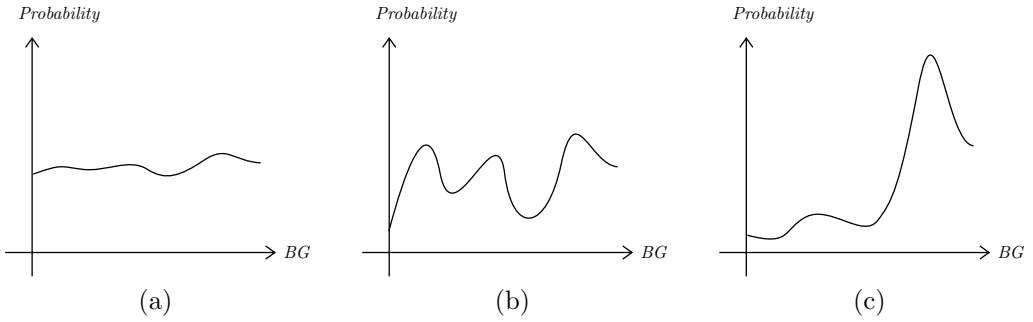


Figure 23.: Example distributions showing the probability density function of the the blood glucose variable (BG) in the training data of the surrogate models and how they relate to D_{KL}. (a) presents a uniform distribution of training data across BG resulting in a low D_{KL}. (b) presents 3 points of convergence in the training data for BG, resulting in a higher D_{KL}. (c) presents a single point of convergence in the training data for BG, resulting in a very high D_{KL}.

7.1.5. Metrics

To answer RQ2, we measure the effectiveness of each testing technique in terms of the number of different initial datasets that result in a violation. From this measure, we can quantify each technique's ability to explore the system's solution space and uncover violations.

To answer RQ3, we measure the extent to which each testing technique is able to explore system behaviours. To accomplish this, we measure the diversity of data as it is fed back into the surrogate model throughout the testing procedure (see Figure 4). This diversity can be measured using the Kullback-Leibler divergence (D_{KL}) [93] (Equation (7.1)) between each input variable of the data used to build the surrogate models and a uniform distribution. D_{KL} allows for the divergence of two distributions to be quantified. D_{KL} has been used in a similar way in prior works [48, 108] to represent the diversity of a test set.

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{7.1}$$

Using D_{KL}, we aim to identify the trajectory and convergence of input variables throughout the search procedure. In our case, we set the reference distribution (Q) to a uniform distribution and the test distribution (P) to the surrogate model's training data distribution for a given search iteration. We can then quantify the surrogate's divergence from a uniform data distribution as more data is fed back into the surrogate model (Section 3.4.4).

Figure 23 presents 3 potential distributions for the blood glucose (BG) input variable, each resulting in a different D_{KL} (with respect to a uniform distribution). (a) presents a relatively uniform distribution of values across BG. Such a distribution would result in a low D_{KL} being calculated. If surrogate-assisted testing resulted in this distribution, it suggests that BG was explored across all values equally when feeding data back with each iteration. (b) illustrates a less uniform distribution where there is more data for some specific values of BG than others. Since this distribution is less uniform, it would result in a higher D_{KL} being calculated, suggesting a dataset which has converged on some specific values during

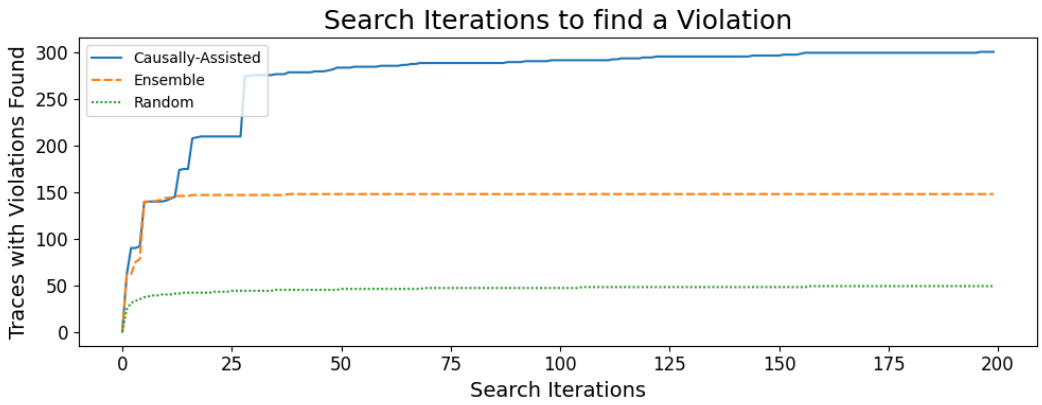


Figure 24.: Total number of traces where an oref0 violation was found and the search iteration at which it was executed on the simulator.

the testing procedure. (c) presents a distribution where a specific value of BG exists in the dataset considerably more than other values. This distribution would result in an even higher D_{KL} being calculated. Such a distribution suggests that surrogate-assisted testing is feeding back the same input value continuously, without exploring other system behaviours.

Although D_{KL} has been used in prior works to measure test set divergence [48], it should be noted that measuring the convergence of a machine learning model's training data distribution D_{KL} is a novel use. This metric allowed us to quantify the uniformity of training data throughout the surrogate-assisted testing procedure. This provides an opportunity for future studies on surrogate-assisted testing to measure the data flow back into the surrogate model.

7.2. RQ2 Results: Effectiveness

Figure 24 presents the total number of initial datasets for which violations are found and the number of search iterations required to find them. For this RQ, we use the same reporting technique as used in RQ1 in Section 5.2.

Our approach found violations in 301 of the initial 924 datasets, whereas the associative approach was only able to find violations in 148. The random search was only able to find violations in 49 datasets within the computational budget, illustrating a larger difference than when evaluating Pylot. We discuss the initial increase of the random search in Section 8.5. These results show how both testing approaches were able to find system violations much more efficiently than the random search, but, ultimately, our causal surrogate was able to find system violations more effectively than the ensemble surrogate.

Unlike our previous evaluation, we did not find violations in all initial datasets for oref0 and required many more iterations. As stated in Section 2.2, an APS acting correctly is paramount to the user's safety. Therefore, it is unsurprising that finding scenarios which resulted in system violations took longer for each testing technique.

If we take a closer look at the difference between the ensemble approach and our causality-

assessed approach, the former finds the majority of its violations within the first few simulator iterations. Since the ensemble approach is highly associative, this is not surprising as associations between inputs and real violations in the initial datasets would lead to this approach finding violations quickly.

However, we observed a plateau in the number of violations found after this initial increase. This means that datasets in which the violation was not found quickly were unlikely to result in a violation being found. Such a quick plateau suggests that the technique struggled to explore outside of potentially giving rise to spurious associations. Since the data in this evaluation (the OpenAPS Data Commons) is pre-existing data, there is no guarantee that spurious associations are not inferred.

Our causally-assisted approach shows a similar initial increase in violations to the ensemble approach. This indicates that our approach still exploits the associative properties in the data. As a result, the technique found inputs highly correlated with violations, similar to the ensemble technique. Our approach, however, also demonstrates a more exploratory approach to the system's behaviour. Thanks to the different perspective (refuting causal relationships), data from both associative and causal approaches are fed back into the surrogate models (Section 4.1.1). This allows for a more exploratory search process which was able to account for datasets biases and represent more nuance in system behaviour. As a result, the technique was able to identify more violations at later search iterations.

We observe that our causally-assisted approach continues to find violations towards the end of our computational budget. There is potential that this approach may have found additional violations in the remaining 623 initial traces given a larger computational budget. However, since each search iteration required an execution of the simulator and the ensemble approach has reasonably plateaued, we believe our choice of computational budget to be sensible. From the results to this RQ, we are able to demonstrate the applicability of our approach with respect to the state-of-the-art.

Summary: From RQ2, we found an increase in the effectiveness of finding CPS violations by using our causal surrogate model. For oref0, our approach was able to find violations in over double (153 more) the number of pre-existing traces than the state-of-the-art technique within the computational budget. Our approach was able to account for the non-curated nature of the data and more successfully explore the complex behaviour of oref0.

7.3. RQ3 Results: Diversity

Figure 25 presents the D_{KL} for the distribution of each input variable in the surrogate models for the oref0 evaluation. The shaded area represents the variation across each initial dataset at the given timestep. For both approaches, associative and causally-assisted, we see a general increase in D_{KL} as the search process introduces more data into the surrogate. This suggests that the search process generally converges on specific areas of interest when suggesting candidate test cases.

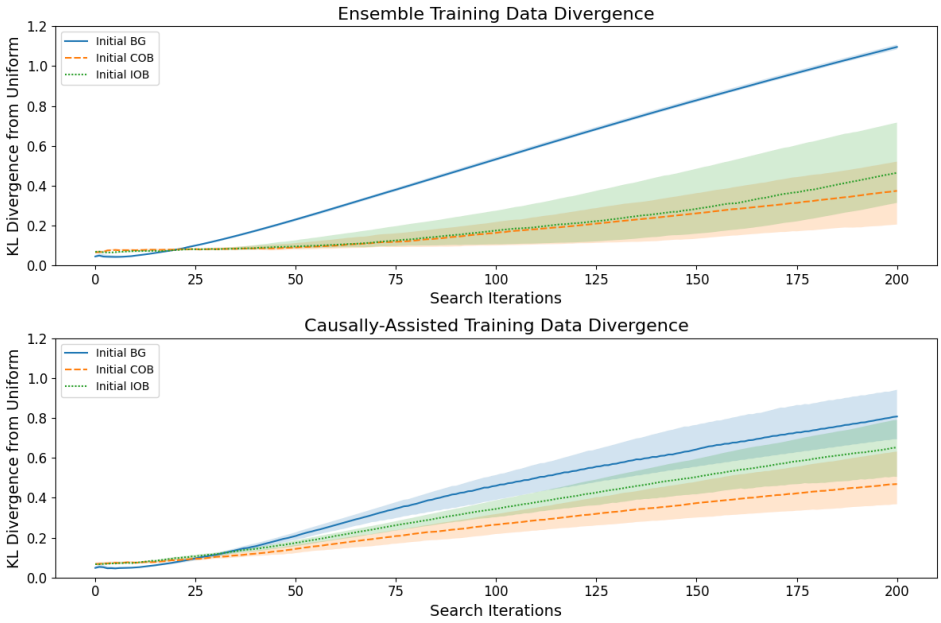


Figure 25.: Training data D_{KL} of each variable used to generate the oref0 surrogate model. The first and third quartiles are highlighted to illustrate the distribution across the datasets.

From this figure, however, we observe how the associative method prioritises the initial blood glucose (BG) of the scenario. This can be seen by the sharp increase in D_{KL} of the blood glucose variable with a very low variation between the different traces. The other variables increase in D_{KL} throughout the testing procedure but not nearly to the extent of blood glucose. Such a high D_{KL} in the initial blood glucose suggests a very high convergence of the search procedure on a specific blood glucose input value (eg. Figure 23 (c)).

A high D_{KL} for blood glucose suggests an association between blood glucose and violations in the initial datasets. However, with pre-existing data, such association may be spurious, as discussed in Section 3.6.3. By converging the search algorithm based on a spurious association, many more search iterations would be needed, increasing the time and resources required to find a violation. This is supported by our findings for RQ2, where an initially large number of violations were found, while only a few were found throughout the remainder of the testing procedure.

Our approach, in contrast, can be seen not to only focus on one specific variable. The D_{KL} of all three input variables increase without one variable increasing considerably more than the rest. We do see that the blood glucose variable, ultimately, ends with a higher D_{KL}, but remains more consistent with the other input variables when compared to the purely associative method.

We also observe how the distribution of D_{KL}s for each variable are more similar in our approach. The different perspectives of the system's behaviour allowed for more of an exploratory approach when reintroducing data to train the surrogate models. These factors ensured that all variables are explored, resulting in a more uniform distribution of data being used to build the surrogate model. From these results, we observe how combining data from

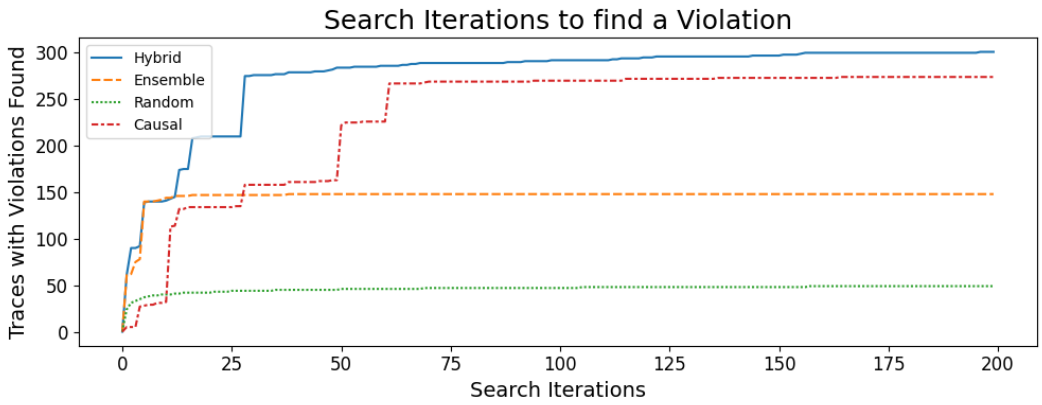


Figure 26.: Total number of traces where an oref0 violation was found and the search iteration at which it was executed on the simulator.

both the associative and causal surrogate models in our approach allows for exploration and exploitation to complement one another.

Summary: From RQ3, we found how causal surrogate models explore the search space more evenly than the purely associative approach. We observe how combining data from associative approaches and causally-assisted techniques improves our approaches ability to explore more during the search procedure. As a result, our approach inhibits the search process from focusing on a single area of the search space by providing a more exploratory perspective than the state-of-the-art.

7.4. RQ4 Results: Ablation Study

RQ2 and RQ3 do raise the question as to how the causal surrogate model would perform outside of the hybrid approach. To answer this question, we performed an ablation study [155] in which we perform the same evaluation as the previous two research question with the associative surrogate model removed from our approach. For this RQ, we refer to the causal model on its own as *Causal* and our original approach, which uses both surrogate models, as *Hybrid* so the two approaches can be compared.

Figure 26 presents the same results as RQ2 (Figure 24) with the addition of the Causal approach. Our original Hybrid approach found violations in 301 traces within the computational budget whereas the Causal approach only found violations in 274 traces. The causal surrogate model on its own was therefore not as effective as our Hybrid approach, but was still able to outperform the state-of-the-art Ensemble approach (148 traces).

However, the trajectory of the Causal approach illustrates the benefits of both the causal surrogate model and the applicability of the Ensemble approach. We observe how both the Hybrid and Ensemble approaches have a similar initial increase. This increase was not exhibited by the Causal approach. We, therefore, highlight how associative surrogate models

are very effective at finding initial violations and how the approach lacking this associative aspect, the Causal approach, was not as effective in the short term.

The Causal approach does find more violations over time than the Ensemble approach. This suggest that, even without a combined dataset, the Causal approach was more exploratory. An exploratory approach allowed this technique to find more violations after the Ensemble approach had plateaued.

Ultimately, our original Hybrid approach was the most effective approach. However, this RQ does highlight the importance of the Ensemble model to find initial violations and the applicability of the Causal model's ability to explore when finding additional violations.

Summary: From RQ4, we highlighted the importance of both exploiting the associative properties of the data and the exploratory nature of the causal surrogate model. As a result, we showed how a causal surrogate model on its own is more effective than a purely associative approach. However, a combination of both causal and associative surrogate models is more effective at finding system violations than each separately.

7.5. Statistical Significance

Table 9.: The number of oref0 violations found and the resulting Pearson's chi squared values calculated for them at different search iterations. Chi Squared values outlined in **bold** indicate an inability to demonstrate independence. Values are given to 5.d.p.

Search Iteration	Violations Found (Ensemble)	Violations Found (Causally-Assisted)	Pearson's Chi-Square Value
10	142	140	**0.01674**
20	147	210	13.77962
30	147	275	50.31413
40	148	279	52.26640
50	148	284	55.87696
60	148	286	57.34836
70	148	289	59.58429
80	148	289	59.58429
90	148	290	60.33724
100	148	292	61.85455
110	148	292	61.85455
120	148	295	64.15885
130	148	296	64.93447
140	148	296	64.93447
150	148	297	65.71384
160	148	300	68.07429
170	148	300	68.07429
180	148	300	68.07429
190	148	300	68.07429
200	148	301	68.86852

To ensure our evaluation demonstrates a significant difference between the state-of-the-art approach and our own, we performed a statistical significance analysis [10]. We focused this analysis on the results of RQ2 since they compare the two approaches. In this section, we assumed a p-value of less than 0.05 to be statistically significant [169].

Following from our statistical analysis in our previous evaluation in Section 5.3, we perform a Pearson's Chi Squared test [24], henceforth referred to as the cs-test, on our results. This statistical test enables the testing of a null hypothesis for categorical datasets, such as that

Table 10.: The means, Shapiro-Wilk p-values, Mann-Whitney U-test p-values and Cohen's D d-values for the distributions of results from RQ2 at different search iterations. Values outlined in **bold** indicate where statistical tests have failed. For Cohen's D, **bold** indicates negligible effect. Values are given to 5.d.p.

Search Iteration	Mean (Ensemble)	Mean (Causally-Assisted)	Shapiro-Wilk p-value (Ensemble)	Shapiro-Wilk p-value (Causally-Assisted)	Mann-Whitney U-test p-value	Cohen's D d-value
10	0.15367	0.15151	0.17495	0.19614	**0.69578**	**0.03403**
20	0.15909	0.22727	0.29894	0.61137	0.00078	-1.00576
30	0.15909	0.29761	0.06237	0.07250	2.41114e-06	-1.50329
40	0.16017	0.30194	0.26162	**0.03239**	1.45526e-09	-2.43912
50	0.16017	0.30735	**0.04648**	0.27001	2.51277e-09	-2.37874
60	0.16017	0.30952	**0.00759**	0.05529	5.27301e-09	-2.48491
70	0.16017	0.31277	0.21410	0.79506	2.33261e-08	-2.10915
80	0.16017	0.31277	**0.00890**	0.37400	6.70651e-07	-1.70360
90	0.16017	0.31385	**0.04799**	0.20062	1.04505e-09	-2.46731
100	0.16017	0.31601	0.10659	0.12446	5.79982e-07	-1.76097
110	0.16017	0.31601	0.22944	0.46135	4.50633e-08	-1.95269
120	0.16017	0.31926	0.57594	0.23620	2.53691e-07	-1.77448
130	0.16017	0.32034	**0.03104**	0.25941	4.19871e-07	-1.77552
140	0.16017	0.32034	0.06300	0.70033	4.07518e-08	-2.12351
150	0.16017	0.32142	**0.02176**	0.25378	2.02556e-08	-2.25621
160	0.16017	0.32467	0.14745	0.33746	1.76372e-08	-2.26049
170	0.16017	0.32467	**0.00307**	**0.03431**	4.75513e-10	-2.61099
180	0.16017	0.32467	0.14074	0.38019	4.93045e-09	-2.21816
190	0.16017	0.32467	0.58180	0.37906	4.32424e-08	-2.05335
200	0.16017	0.32575	0.10119	0.09582	3.12331e-09	-2.47617

produced in our evaluation. The null hypothesis in this evaluation is that the results are not independent and therefore there is no statistical significance between our technique and the state-of-the-art.

We performed the cs-test at different iterations throughout the testing procedure, starting at iteration 10 and increasing by steps of 10 until iteration 200. As a result, we were able to determine how the statistical significance of our approach changed throughout the testing procedure.

Table 9 presents the cs-test values at different timesteps across our evaluation of oref0. From these results, we observed that our null hypothesis is between timesteps 10 and 20, presenting a statistical significance after these timesteps. This observation was achieved with the use of a cs-test p-value table [145] with a p-value of 0.05. The resulting pattern can be seen mirrored in the results of RQ2 in Figure 24 as this is the point at which our approach diverges from the state of the art.

Since this evaluation has more data than that in Chapter 5, we can perform additional statistical tests to reinforce our technique's statistical significance. Due to the additional data, we were able to generate distributions from our binary data by following the existing technique of simple random sampling [123]. For each surrogate modelling approach, the results of the 924 traces were randomly assigned to 28 samples, each consisting of 33 binary outcomes as to whether a violation was found. Each sample was then averaged, producing a value between 0 and 1. As a result, the 924 results were processed into 28 values that made up a distribution for each surrogate modelling approach.

Similar to the cs-test, we repeated the process above at different search iterations to determine how the statistical significance of our approach changed throughout the testing procedure. For each, we assigned the binary outcome as to whether the testing procedure

had found a system violation by the given search iteration.

However, before choosing a statistical test to apply, we first had to determine whether our distributions were normal distributions. Since, some statistical tests require normal distributions [53], it was important to first analyse the distributions individually to ensure the correct test was chosen. To accomplish this, we use the Shapiro-Wilk (SW) statistical test [66]. This test attempts to refute the hypothesis that the distribution follows a normal distribution.

Table 10 presents the mean and SW p-value for each of the distributions at each search iteration. For both testing approaches, some iterations resulted in an SW p-value less than 0.05. Therefore, the null hypothesis of the SK test cannot be refuted for all iterations. In which case, we must use a non-parametric statistical test when comparing our two distributions.

To evaluate statistical significance, we used a Mann-Whitney U-test [110], henceforth referred to as a u-test. This test identifies whether two distributions reject a null hypothesis. In our case, the distributions are the outcomes from the two approaches, and the null hypothesis is that the underlying distributions are the same. Therefore, we aim to prove that the results from our approach are statistically significantly different from that of the state-of-the-art.

Table 10 presents the p-values generated as the results of the u-tests at different search iterations. Note that, as stated earlier in this section, a p-value of less than 0.05 is statistically significant [169]. We observe that our approach becomes statistically significant between iteration 10 and 20. From iteration 20, we observe p-values considerably less than 0.05, demonstrating the statistical significance of our results.

We also measure the effect size to determine the expected difference between the two approaches [57]. To measure effect size, we calculate the non-parametric effect size Cohen's D [32], which measures the difference between the mean values of two distributions. D-values, resulting from calculating Cohen's D, can be interpreted as a small effect (0.2), a medium effect (0.5) and a large effect (0.8).

Similar to statistical significance, we measure the effect size of the approach distributions at different search iterations. By doing so, we measure the difference between the mean values of the different approaches as the testing procedure progresses.

Table 10 presents the d-values calculated from Cohen's D when comparing the distributions at different search iterations. We observe a very small effect size at iteration 10 and then a much larger negative effect size from iteration 20 onwards. This pattern follows the results of RQ2, since the two approaches diverge between iterations 10 and 20.

From these results, we demonstrate that the effect size is much less than 0.2 before the state-of-the-art plateaued and, therefore, there is minimal difference in effect between our approach and the state-of-the-art. However, after the state-of-the-art's plateau, the effect size is large (the absolute values are much greater than 0.8), demonstrating a significant difference in effect between the two approaches.

7.6. Threats to Validity

Similarly to our prior evaluation in Chapter 5, we use this section to explore potential threats to validity for this evaluation. We, as in Section 5.4, focus specifically on the internal validity of this evaluation. We discuss further threats to the validity of our approach, focusing on both internal and external validity, in Section 8.6.

In RQ2, we mention that our approach may find more system violations given a larger computational budget. Therefore, a potential threat to validity is that the computational budget assigned for this is too low. However, our computational budget does demonstrate a large difference in violations found between the different approaches. The Ensemble method is seen to plateau, with the final violation found at search iteration 38. Our approach does continue to find violations through to iteration 196. However, the difference between the two approaches by this iteration is definitely apparent.

A potential threat to validity is the choice of data used in this evaluation. To alleviate this, we used the largest open source dataset of T1DM available at the time of writing. This dataset was derived from real world use of oref0, providing a pre-existing dataset from which to test oref0. Since one of our motivations for testing an APS was a lack of curated data, this pre-existing dataset exhibited all the factors that makes CPS testing difficult, such as potential sensor error, a lack of consistency, and a high likelihood of human error [52, 94]. This is further discussed in Section 6.2.2. This dataset allowed us to provide a realistic evaluation of our testing technique based on the difficulties of real life data.

It can also be argued that, although we use a real dataset, by expanding the initial dataset using digital twin executions it is no longer realistic. Since each of the original traces was from different people in different environments, learning a surrogate model across all of them would have resulted in a representation of no individual user. Instead, the surrogate model would represent an average of all users, where since each user's reaction to insulin may be drastically different, this may not have been representative of oref0 behaviour.

In Section 7.1.2, we took each of the 924 initial traces and used the digital twin to learn the blood glucose-insulin dynamics before simulating 300 different scenarios for each different set of dynamics. New scenarios were generated from random sampling of the input variables in Table 8, presenting a range of scenarios from which to observe oref0 behaviour. By doing so, each of the expanded 924 datasets, derived from the initial 924 traces, was each representative of a single user at a single time. Therefore, blood glucose-insulin dynamics would be static for each of the initial traces, allowing for only the behaviour of oref0 to be evaluated.

Another potential threat to validity is that we developed the simulator used in this evaluation. Therefore, it may have been developed in a way that biased our findings. This contrasts the use of CARLA in Chapter 5, which is widely used across studies [62, 70]. Unfortunately, there did not exist a high-fidelity simulator in which we could observe oref0 behaviour. We, instead, developed our own simulator in the form of a digital twin, described in Chapter 6. To ensure our simulator was accurate, we performed an evaluative case study to evaluate its accuracy, prediction and ability to facilitate different system configurations.

We used the same simulator and configurations were used across our evaluation to ensure the only difference in our evaluation was the technique being evaluated.

7.7. Chapter Summary

In this Chapter, we evaluated our causal surrogate model against the state-of-the-art approach when testing a real APS with a pre-existing clinical dataset. Our approach found over double the number of violation than the state-of-the-art when using pre-existing data. We discovered that this was due to the causal surrogate model's ability to explore outside of the potentially spurious associations.

By combining both causal and ensemble models, our approach was able to find more violations than each surrogate model individually. We also highlighted the strengths of each individual surrogate model, emphasising the applicability of their combination in our hybrid approach. In the following chapter, we further discuss these results and the implications our approach may have for CPS testing.

Chapter 8. Discussion

In the evaluations in Chapters 5 and 7, we showed how our causal surrogate model improved the effectiveness of surrogate-assisted CPS testing. In this section, we explore what implications this may have for CPS testing, discuss our methodology, and describe some potential threats to validity. We discuss our evaluative case study in Section 6.4.

8.1. RQ1 and RQ2 - Effectiveness

Both RQ1 and RQ2 evaluated the effectiveness of the causal surrogate model against different subject systems. By evaluating against both systems, we provided a variety of domains and present the applicability of our approach across CPSs.

The results of RQ1 showed an increase in effectiveness when using our causal surrogate model when replicating an existing study. Our evaluation found that our causal surrogate enabled surrogate-assisted testing to find system violations in fewer search iterations than the state-of-the art (Figure 10). As a result, the high-fidelity simulator was used fewer times, requiring less computational expense.

The results of RQ2 showed a similar increase in effectiveness when testing an open-source APS using our causal surrogate model. However, not only did our approach find violations with fewer search iterations, our approach was able to find more violations in datasets where the state-of-the-art was unable (Figure 24). From these results, we can see that testing the APS was difficult for the purely associative technique, as it struggled to explore outside of what were potentially spurious associations inferred from the training data. These results also presented the applicability of our causal surrogate model and the strengths of accounting for pre-existing datasets and evaluating causal relationships.

8.2. RQ3 - Diversity

In RQ3, we explored how the causal surrogate model affected the data being fed back as part of surrogate-assisted testing. We measured the divergence of this data from a uniform distribution using the Kullback-Leibler divergence (D_{KL}). Using this metric, we aimed to understand how our approach explored system inputs and how this search behaviour differed from the state-of-the-art technique.

The results to RQ3 showed how our approach was more exploratory than the associative approach. The state-of-the-art was seen to quickly increase the D_{KL} of the blood glucose input, suggesting a single value of interest was being attempted repeatedly. Such a pattern may suggest potential spurious associations being learnt by the associative model. These

incorrect associations would result in no violation when evaluated in the simulator, as seen by our results to RQ2.

Our approach, however, did not follow this trend. Instead, each input variable distribution increased its D_{KL} at a slower rate throughout the testing procedure. This pattern suggested that many different values of each variable were being used in testing. However, there was still an overall increase to each variable, insinuating areas of interest. Again, we compared these results to the results of RQ2 to observe how more exploration led to the discovery of more system violations.

8.3. RQ4 - Ablation Study

RQ4 evaluated the use of **only** the causal surrogate model, foregoing its interaction with the associative model and the resulting combined dataset. From this RQ, we aimed to understand to what extent our surrogate model contributed to the improvement in finding violations seen in the results of RQ2.

The results of RQ4 show how the causal surrogate model alone increased the effectiveness of surrogate-assisted testing, but also highlighted the strengths of the associative surrogate model. The causal surrogate was able to find more violations than the state-of-the-art. However, without assessing the associative properties of the training data, the causal surrogate model required more search iterations. The overall increase in found violations reinforced our interpretation that the increase in effectiveness in RQ2 was due to the integration of the causal surrogate model.

The lack of initial increase in the causal only approach highlighted the strength of the associative surrogate. The traditional approach was able to find many system violations from non-spurious associations in the first few search iterations. We, therefore, found the importance of combining both techniques in our hybrid technique. The hybrid technique assessed associations in the data for a large initial increase in finding system violations, and then relied on the combined dataset and causal relationships to further explore system behaviour.

8.4. Implications for CPS Testing

We found that testing causal relationships of a CPS's behaviour enabled developers to more effectively find violations and, therefore, better validate system behaviour. This is particularly important for CPSs that lack observability [54] in testing, such as an APS interacting with human-in-the-loop, as described in Definition 4. Such systems struggle with the oracle problem (Definition 7) [15] as unmeasurable external factors make defining correct behaviour challenging. By evaluating causal relationships instead of specific input values, our approach enabled the tester to answer causal questions ("what if...?") to identify causal relationships between variables that did not hold.

Causal surrogate models also enabled testers to more effectively test CPSs where curated datasets may not be available. Testers may be required to use pre-existing datasets instead,

since curated data may be computationally expensive or dangerous to obtain [49]. The ability to use pre-existing datasets is particularly important for CPSs that lack controllability [54] (Definition 3 and Section 2.2.1). For example, when testing oref0 we relied on pre-existing clinical data, as obtaining curated data may have been unethical or time consuming. As seen from our evaluation, causal surrogate models increased the effectiveness of using pre-existing data where spurious associations may be inferred.

Our technique is system agnostic, meaning testers can apply it to CPSs across domains. We demonstrate this through evaluating our technique on both an ADS and an APS. Our evaluations, also, highlighted how different domains demonstrate differing benefits from our technique. For Pylot, our technique found violations within fewer search iterations, and for oref0, our technique found additional violations that the state-of-the-art surrogate model was unable to.

8.5. Methodology Discussion

In Chapters 5 and 7, we perform our evaluations using our causal surrogate model. In this section, we explore different aspects of our approach and evaluations to explore how they may have affected the testing procedure.

Variable Visibility We found our technique's ability to control biases is determined by the extent to which the CPS exposed its internal variables. Causal identification (Section 3.6.3) adjusts causal model estimations based on the relationships between exposed system variables. By only exposing inputs and violations, Pylot resulted in a DAG which may not have been detailed enough to identify potential biases in the data (lack of *controllability*). However, we still observed the advantage of evaluating complex behaviour through refuting causal relationships (lack of *observability*) by the evaluation's increase in effectiveness.

For oref0, more variables were visible to the tester, as seen in the DAG (Figure 22). Due to the data availability, we were able to better account for dataset biases when estimating the causal relationships between variables (lack of *controllability*) in addition to accounting for the system's complex behaviours (lack of *observability*).

Exploration and Exploitation Our technique used both exploitation and exploration [177] to search the complex behaviour of CPSs. Our technique outperformed both an exploitative method (the state-of-the-art ensemble) and a purely exploratory method (random search) in both evaluations. We observe in RQ3 how combining both techniques explored system behaviours instead of focusing on specific inputs. By implementing exploitative and exploratory techniques in our approach, we better explored the complex behaviour of systems such as oref0 while still exploiting causal relationships to find system violations.

Single Objective Optimisation We evaluated the surrogate models using a single-objective search algorithm. However, this raises the question as to whether single-objective search is

sufficient for complex CPS behaviours. In our evaluations, a single-objective search effectively found contradictions in expected behaviour. Our results show that it is appropriate for our technique to maximise for a single strong contradiction as opposed to many weak contradictions [121].

Single-objective search contrasts prior evaluations, which used multi-objective search [62, 70] to maximise the likelihood of multiple violations. In our approach, given the 96 causal relationships in our initial evaluation (Chapter 5), we assumed a single causal relationship being strongly contradicted was more helpful for finding system violations than several weaker causal relationships being contradicted. However, future work could investigate this effect by using multi-objective search algorithms alongside a causal surrogate model.

Initial Violation Evaluation Unlike previous works [70], our evaluations aimed to evaluate the surrogate model based on the first violation found. As a result, there is no guarantee that we found a distribution of different violations. Since our work focuses on the effectiveness of the surrogate models, we assume the distribution of violations to be out of the scope of this thesis.

However, while our work provided an insight into the benefits of using causal inference, uncovering different violations provides an opportunity for future work. More system behaviours may be uncovered after additional search iterations feed more data back into the surrogate model. As a result, future work may evaluate how using causal inference affects the frequency and distribution of system violations.

DAG Oracle Problem Causal DAGs are generated by domain experts manually and could potentially be incorrect. Since CPSs are inherently complex [49], ensuring causal DAGs are correctly defined can be non-trivial. As with any model-based approach, they are susceptible to the oracle problem [15] (Definition 7). Since our approach relies on refuting the expected relationships in the DAG, an incorrectly specified DAG may reduce the ability of the proposed technique to find violations.

Approaches, such as causal discovery [63], have been proposed to alleviate this by generating the DAG from data instead of relying on domain expertise. Such techniques, however, rely on curated datasets of correct system behaviour. In Section 2.1.1, we explain why such datasets are not always available for a CPS.

Simulator Correctness As well as affecting surrogate models, we found that pre-existing datasets can affect the configuration of the simulator in surrogate-assisted testing. For our evaluation in Chapter 7, we configured the digital twin of a person using an APS through constants derived from the clinical data. In Section 6.4.2, we highlighted how using clinical data may, however, not always capture the correct glucose-insulin dynamics [158].

For a few traces, we observed the simulator was executed with an insulin sensitivity that was non-physiologically low. The simulation, therefore, resulted in hyperglycaemia regardless of the APS's output. When evaluating random search, this resulted in an initial increase in violations, suggesting a potential limitation when insufficient data is available.

We found this same limitation when validating our digital twin, as described in Section 6.4.4.

8.6. Threats to Validity

In this section, we explore the threats to the validity of our study. We build upon the threats to validity explored in Sections 5.4 and 7.6. We discuss the threats in terms of internal and external validity.

8.6.1. Internal Validity

A potential threat is that the baseline chosen for our evaluation does not present a sufficient comparison. To mitigate this, we evaluated our technique against the current state-of-the-art surrogate modelling-based testing approach [64, 70]. We also compared both techniques with random search, showing how effective the approaches are for each CPS domain. We did, however, encounter an issue with the applicability of the state-of-the-art to oref0, requiring an additional ∼90 days of computation (described in Appendix A). This highlights the difficulty of applying the state-of-the-art technique to more complex, safety-critical systems.

Another potential threat is that the data used in our evaluations may not be extensive enough. To mitigate this threat, we used two sets of data to evaluate our approach, pseudo-random for the initial and real-world data for the main evaluation. We chose to use pseudo-random data to evaluate Pylot because the original evaluation used purely synthetic data. Controlled data may not be available for such systems, as described in Section 2.2.1, so the pseudo-random synthetic data represented pre-existing data. The dataset used to evaluate oref0 was real pre-existing clinical data, where there is no guarantee that spurious associations may not be inferred [97]. By using pre-existing data, it may give rise to spurious associations [27] and it has no guarantee that violations are present in the data. This dataset contains all the potential CPS testing challenges outlined by a lack of *controllability*, as defined in Section 2.1.1.

The choice of metric used to measure the divergence and distributions of the surrogate model training data in RQ3 (Section 7.3) could be a potential threat to validity. To mitigate this, we use the metric D_{KL} to measure the test set divergence of the surrogate model's training data throughout the search procedure. This metric has been used in prior works [48, 108, 157] to represent the divergence of a test set. We interpret this metric in Chapter 7 based on its prior use, which is described in Section 7.1.5.

Another potential threat to the validity to our evaluations is that the code used to generate the results could contain bugs. Incorrect code could have resulted in incorrect results that were not representative of our technique and may have potentially biased our findings. This is an inherent problem with code written by a human due to human error [146]. To minimise this threat, the code used throughout this thesis was executed initially on subsets of data. In doing so, we uncovered unexpected behaviours in the digital twin, CARLA and our evaluation code. For example, a misinterpretation of PyGAD's configurations resulted in the boolean values in Table 1 being held as false. We were able to identify this incorrect

code and ensure those faults did not propagate through to our results.

8.6.2. External Validity

A potential external threat would be that of generalisability across other CPSs. Our study demonstrates an initial evaluation and a more in-depth evaluation using two CPSs from different domains. We used Pylot because of its use in prior evaluations [62, 70], and oref0 because it is open-source and has extensive data availability. Other CPSs also provide open source datasets, such as CARLA Garage [75], but are typically curated, and therefore not relevant to this study. Some other datasets are only high-level, such as those produced by driving authority organisations [180], and are, therefore, not detailed enough to perform system testing. Nevertheless, future work could evaluate the use of causal surrogates in more CPS domains.

It can be argued that our evaluations are not reproducible and can, therefore, not be applied to other systems. To alleviate this issue, we provide a detailed account of our methodology in both Chapters 5 and 7. The datasets used in both evaluations are either open-source or generated as part of our replication package and are, therefore, available for future evaluations. We also ensured that static random seeds were used across the evaluations. The results of our RQs can, therefore, be replicated, regardless of stochastic factors. Our reproducibility package is described in Section 1.1.2.

Chapter 9.
Related Works

In this Chapter, we describe existing works that are related to this thesis. We use these works to compare our approach to the current state-of-the-art techniques and highlight how we build upon them.

9.1. Surrogate-Assisted CPS Testing

Surrogate-assisted CPS testing is a novel technique for reducing the compuatational expense of testing CPSs [120]. In Section 3.4, we described how our approach follows the surrogate-assisted testing approach described in Haq et al. [70]. Their approach uses associative surrogates to suggest test cases likely to cause ADS violations and then evaluates the resultant test cases on a high-fidelity simulator. They used local and global searches to improve the effectiveness of the search algorithm. We expand on this approach by integrating our causal surrogate model. We used this approach as the baseline for our evaluations.

Menghi et al. [116] introduce a similar approach that uses surrogate models to find system violations in a satellite. Their approach, however, uses surrogate models that approximate each timestep of a simulation of system behaviour. This surrogate model creates a less computationally expensive simulation that can be used in the search process. This contrasts with Haq et al. [70] and our approach, which estimates the likelihood of violations based on scenario inputs. Each time step provides a more nuanced representation of the system's behaviour but is more computationally expensive than our approach. However, these surrogate models do present potential future work to identify the benefits of applying causal inference to timestep-based surrogate models.

9.2. Causal Inference in Testing

Causal inference is an emerging concept in software testing. Podgurski et al. [91,141] demonstrate a source code fault localisation technique, aided by the bias-reducing techniques of causal inference. This approach works by evaluating the code base, which is not always possible for CPSs. CPSs can contain black-box, multi-domain elements for which source code analysis is not possible [49].

Clark et al. [27] use causal inference to test computational models. Similar to our approach, regression models are used to represent the subject system due to the expensive runtime and large input scales of the computational models. By using causal inference, pre-existing data could be used to approximate the behaviour of systems, while accounting for potential spurious associations. This work also uses the causal DAG as a test oracle, allowing a developer to test the expected causal relationships of a system. Such techniques

have also been applied to CPSs [159]. We build upon this work by using the expected causal relationships of a DAG to guide the discovery of incorrect behaviour. From this, we not only test causal relationships but attempt to find inputs that lead to system behaviours that do not satisfy these relationships.

Causal inference has been applied to reduce bias in surrogate-assisted CPS testing in prior works [62]. This technique, however, relies on causal discovery [63], which learns causal relationships and generates the causal DAG from data. Causal discovery works under the assumption that the causality can be detected from statistical dependencies [183]. Due to the lack of controllability of CPS datasets, outlined in Section 2.1.1, this may not always be possible. Data that covers all behaviours in the system's specification are not guaranteed. In our technique, we derive the causal DAGs from domain knowledge to act as our test oracle. By encoding the expected causal relationships, instead of just those learnt from the data, we use these expectations of system behaviour to guide the search for incorrect behaviour.

9.3. Boundary Analysis Testing

Dobslaw et al. [44] introduce a boundary detection technique to find the boundaries between behaviours in software. Similar to our approach, they use the gradient of the system's outputs to find these boundaries. The boundaries are then used as part of a test selection technique to cover different behaviours and find inputs for which behaviours change unexpectedly.

In this thesis, we describe a causal surrogate model, which also finds unexpected behaviours from program derivatives. However, we expand on boundary analysis testing by using a test oracle (the causal DAG) to guide test selection. By using a causal DAG as the system oracle, we not only use the gradient to guide the discovery of system behaviours but also use this gradient to refute our expectation of system behaviour. As a result, we do not aim to find the boundaries between different behaviours, as seen in Dobslaw et al. [44], but instead identify inputs that contradict our expectation of system behaviour the most.

9.4. Digital Twins in Healthcare

Digital twins have been proposed to be implemented across the medical domain to better inform life saving decisions without costly clinical trials. Kiagias et al. [86] present a simulation-based approach which predicts treatment responses for people with tuberculosis. They show how *in silico* experiments speed up the ability for medical interventions to be delivered, without compromising safety and effectiveness. Our digital twin, whose development is described in Chapter 6, capitalises on such notions by simulating the medical interventions of system in a safe environment to ensure the system is behaving correctly for a given configuration.

The Cardiac Physiome Project [31,117] argue the importance of simulation based medicine for a better understanding of cardiac tissues, whilst outlining challenges surrounding the representation of tissue behaviour due to model resolution and approximation. This presents

an ongoing challenge for digital twins in the medical domain. Corral-Acero et al. [37] propose a solution to this through a digital twin for precision cardiology. This would allow simulations to be patient specific, influence clinical decisions and be updated with both population and individual data over time.

Corral-Acero et al. do, however, highlight an important social challenge with the adoption of such technology, as clinicians may find it difficult to trust a 'black-box' for clinical decisions [37]. We use this notion to further motivate the use of an explainable model when developing our digital twin in Section 6.1.

9.5. Digital Twin-based CPS Testing

In Chapter 6 we described the development of a digital twin to be used to validate APS behaviour as part of surrogate-assisted testing. In this section, we describe how digital twins have been implemented to test CPSs and discuss the similarities and differences from our approach. This section is an adaptation of a published literature review [160], P1 from Section 1.1.1.

9.5.1. Digital Twin Testing Areas

Section 3.5.3 described how digital twins have been proposed to test healthcare devices. In this section, we discuss other CPS domains which have had digital twin-based testing and highlight their differences from healthcare applications.

Manufacturing and other industrial applications [74] make up the majority of digital twin-based testing applications. These systems tend to be complex, modular and work in high risk environments [49]. The influence of Industry 4.0 has increased the number of digital twins in manufacturing, providing more informed behaviour for increased safety and reliability [72]. Digital twin-based testing can be found in other sectors, such as the utilities sector in the form of wind turbines [6, 137, 182] and the information technology sector in the form of drones [68].

Similar to our application, these domains tend to be human-interacting [49, 72]. However, unlike medical devices, these CPSs exist in environments that contains humans, as opposed to being integrated human behaviour. Douthwaite et al. [46] designed a digital twinning platform specifically for human-collaborative environments. Ensuring behaviour across such environments is paramount to these systems' use, requiring the ability to test CPS functionality.

9.5.2. Digital Twin Oracles

In Section 3.5.2, we described how a digital twin has been proposed to be used as a test oracle. In this section we explore digital twin oracle implementations and compare them to our technique.

Our technique uses the expected causal relationships of a system as an oracle when defining correct and incorrect behaviours. These relationships are derived as part of a causal DAG

from domain expertise. Barr et al. [15] define this kind of oracle as "specified" as it is based on the expected specification of the system.

Unlike our technique, digital twins themselves have been used as test oracles in prior works. Simulation based software, such as MATLAB Simulink [113], or via "off the shelf" technology solutions [68], have been used to simulate the expected behaviours of its physical counterpart. These approaches are also defined by Barr et al. [15] as "specified" oracles.

Prior works have learnt an oracle digital twin from past execution data of the system. Such "derived" oracles [15] learn the digital twin model from data using historic runtime data from the system under test when it is both running correctly and incorrectly [61]. The model can then be compared to the physical system under test to examine if it is behaving correctly [179].

As we found in the case study from Chapter 6, obtaining data that is curated for medical devices is not always possible. The intrinsic nature of clinical datasets, as described in Section 6.2.2, means that deriving an oracle from clinical data may not be feasible.

9.5.3. Digital Twin Explainability

In Section 3.5.3, we described how the explainability of a digital twin's model is useful for providing confidence in the predictions of medical interventions. We used this notion when adapting the model of our digital twin in Section 6.1.1. We use this section of the related works to explore how other digital twin-based testing applications have approached this requirement for explainability.

Digital twins in testing have been categorised in terms of their explainability as "white-box", "grey-box" and "black-box" [160]. The explainability of the digital twin presents the ability to observe and infer information within the model of a digital twin [172]. For testing, these modelling terms can be used to classify how simple information can be extracted from the model in order to better understand test failures or the causes of incorrect behaviour. We use this section to describe the modelling techniques used and how they compare to our approach.

White-box models are those that have high information visibility. An expert in the field can understand its inner workings and infer information about the model executions [105]. Digital twin-based testing uses white-box models defined "where the equations of motion have been derived from the underlying physics of the problem and the model parameters have direct physical meanings" [174]. Simulation based digital twins are the most common white-box approach found in testing CPSs, providing strong visual insight into the architecture of the system under test. White-box models highlight the emergence of specialised techniques with electronics based CPSs being modelled using mathematical methods as opposed to the more popular simulation approaches [138].

Black-box models are those with low information visibility, not allowing for an expert in the field to understand the inner workings of the system [105]. In the context of digital twins, these models are "derived entirely from measured data, with no assumed knowledge of the physics at all" [172]. Black-box models provide the most popular implementation

of digital twins found in CPS testing, neural networks. These models allow for the correct behaviour of a system to be trained and tested against a live system [137,179]. This does not provide the same granularity as found in white-box approaches, but can more realistically represent system behaviour as it is derived from historical data.

Grey-box models provide partial visibility into the inner workings of the model as well as its inputs and outputs [174]. This allows for experts in the field to gain partial explanations for the occurrence of outcomes from given inputs. In the context of digital twins, these combine both the physics based approaches seen in white-box models and the data driven approaches seen in black-box models [172]. This is mostly done through enhancing a white-box simulation through the use of black-box fault classification to infer more information from test failure data [6].

Inline with these terminologies, the digital twin described in Chapter 6 is developed as white-box. Section 3.5.3 highlights the importance of explainable treatments for medical digital twins, informing our decision to ensure an explainable T1DM model. Corral-Acero et al. [37] do, however, propose the use of black-box statistical models for medical devices when the underlying physiology is not well understood. The use of white-box models for physiological behaviour mirrors that of other CPS domains where the physical behaviours are well understood [174].

Chapter 10. Conclusion

CPSs are becoming more prevalent across human-interacting and medical domains. As a result, ensuring that their behaviour is correct is paramount to their use and the safety of the users. However, finding system violations in CPSs can be challenging due to unmeasurable factors affecting their behaviour and the difficulty of obtaining controlled datasets. Existing techniques, such as associative surrogate modelling, present a way of representing this behaviour in a computationally efficient manner. However, such techniques can struggle to evaluate system behaviour affected by *unmeasurable factors* and require *curated* datasets to accurately represent system behaviour.

In this thesis, we introduced the causal surrogate model, which integrates with existing surrogate-assisted testing techniques. Our surrogate model accounts for pre-existing datasets that may give rise to spurious associations while using the expected causal relationships of variables in the system to guide CPS testing to find system violations. We performed an initial evaluation by replicating a previous evaluation of surrogate-assisted ADS testing using causal surrogates. We also evaluated our technique on a safety-critical APS to further investigate the effectiveness of our approach. We then investigated how using causal surrogates allows for the navigation of the system's complex behaviours and highlighted the strengths and weaknesses of both our causal surrogate model and the state-of-the-art associative surrogate model.

Our results showed that our technique found violations in pre-existing data more effectively than the state-of-the-art associative surrogate models when applying surrogate-assisted CPS testing to an ADS. For the safety-critical APS, our results showed how our technique found over double (153 additional) the number of violations. Our evaluations demonstrated our causal surrogate's ability to account for the potential of spurious associations being inferred from pre-existing data and efficiently explore the solution space of the system. As a result, we show how causal surrogates alleviated the requirement for developers to curate datasets to effectively perform CPS testing and allowed causal questions to be answered to evaluate system behaviour that may appear inconsistent.

To enable our second evaluation, we performed an evaluative case study on the development of a digital twin of a person with T1DM using an APS. This digital twin was then used as a simulation environment during surrogate-assisted testing. From this case study, we found a proposed framework for testing medical devices using digital twins to be applicable in the domain of an APS. The framework allowed for accurate digital twin-based blood glucose-insulin predictions to enable configuration testing of an open-source APS. We evaluated simulations and predictions of the digital twin to validate the simulation environ-

ment used for configuration testing. Our digital twin enabled the execution of different APS configurations in this simulated environment, resulting in the identification of behaviours that could be unsafe to a user. Configurations were observed without requiring physical trials, reducing physical risk to the user.

Our case study also identified difficulties associated with simulating human interaction with a CPS. Model misrepresentation of a user's blood glucose-insulin dynamics led to distorted system behaviour being presented. We alleviated this challenge by fitting the digital twin multiple times and confining its predictions to a 2 hour time window due to changing glucose-insulin dynamics over time.

Using real clinical data also presented a substantial threat to the approach. Human factors and data inconsistency led to a large proportion of the data being impractical for training the digital twin, resulting in only 2.37% of the pre-existing clinical dataset being usable. Since the difficulties associated to the data used in this thesis are inherent to clinical data, it's unlikely that other datasets or additional data would have yielded different results. The challenge of data availability, also, does not only apply to clinical data, meaning it would be interesting to observe the effects of controllability in obtaining data across more CPS domains.

In future work, we aim to examine more CPS domains to examine further testing limitations and, therefore, the impact of causal inference in CPS testing. Investigating the effects of controllability and observability across other CPS domains would solidify the applicability of our approach. Ensuring correct causal DAG representation of complex system components, such as neural networks and non-deterministic behaviour, would also allow more efficient testing of complex CPSs. In addition, we aim to explore the use of additional aspects of causal inference, such as accounting for time-based and unmeasured variable biases, to further improve surrogate model estimations for adaptive systems.

A potential approach to this could be the application of quantitive bias analysis (QBA) [25]. QBA is a technique of causal inference that quantifies how much an unmeasured bias-inducing variable would need to affect a causal relationship for the measured causal effect to be nullified. Such a technique could be used to derive confidence intervals for the causal relationships made by the surrogate models. Preliminary work into this topic was done as part of this research but more time would have been required to obtain meaningful results.

Further exploration into how using a causal surrogate model may affect the other testing objectives highlighted but not covered in this thesis (fragility, robustness) would inform the wider applicability of our approach. By alleviating the lack of controllability and observability in CPSs, using a causal surrogate model may help to test these additional objectives. For example, asking causal questions about minor deviations in environmental factors may allow for incorrectly large system reactions to be identified during system fragility testing.

Future work could also extend the evaluative case study of the applicability of digital twins for testing medical devices. Further evaluation of the blood glucose-insulin model would present more evidence of its real-world applicability. We performed a temporal analysis to identify which aspects of the testing procedure were the most expensive. A further sensitivity

analysis may provide information required to streamline the most temporally expensive segment of testing, fitting the model constants. Future stability analysis and resource-based evaluation would also present evidence as to whether such a testing procedure could be applied in real time.

Since APSs and medical devices in general are safety critical, future work could be performed to conduct a safety case [164] for our technique. It would be important to ensure cross-domain collaboration, through both software testing and medicine, to ensure the requirements of our technique, as well as the user of an APS, are accounted for in such research.

Bibliography

[1] Odd Olai Aalen, Kjetil Røysland, Jon Michael Gran, Roger Kouyos, and Tanja Lange. Can we believe the dags? a comment on the relationship between causal dags and mechanisms. *Statistical methods in medical research*, 25(5):2294–2314, 2016.

[2] Houssam Abbas, Bardh Hoxha, Georgios Fainekos, and Koichi Ueda. Robustness-guided temporal logic testing and verification for stochastic cyber-physical systems. In *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*, pages 1–6, 2014.

[3] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11), 2018.

[4] Syed Haris Ahmed, David L Ewins, Jane Bridges, Alison Timmis, Nicola Payne, Cormac Mooney, and Claire MacGregor. Do-It-Yourself (DIY) artificial pancreas systems for type 1 diabetes: Perspectives of two adult users, parent of a user and healthcare professionals. *Advances in Therapy*, 37(9):3929–3941, September 2020.

[5] Jameela Al-Jaroodi, Nader Mohamed, and Eman Abukhousa. Health 4.0: On the way to realizing the healthcare of the future. *IEEE Access*, 8:211189–211210, 2020.

[6] Amin Amini, Jamil Kanfound, and Tat Hean Gan. An ai driven real-time 3-d representation of an off-shore wt for fault diagnosis and monitoring. In *PervasiveHealth: Pervasive Computing Technologies for Healthcare*, pages 162–165. ICST, 10 2019.

[7] Kamilla Egedal Andersen, Simon Köslich, Bjarke Kristian Maigaard Kjær Pedersen, Bente Charlotte Weigelin, and Lars Christian Jensen. Do we blindly trust self-driving cars. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-robot Interaction*, pages 67–68, 2017.

[8] Dimitrios Angelis, Filippos Sofos, and Theodoros E Karakasidis. Artificial intelligence in physical sciences: Symbolic regression trends and perspectives. *Archives of Computational Methods in Engineering*, 30(6):3845–3865, 2023.

[9] Hugo L. S. Araujo, Gustavo Carvalho, Morteza Mohaqeqi, Mohammad Reza Mousavi, and Augusto Sampaio. Sound conformance testing for cyber-physical systems: Theory and implementation. *Sci. Comput. Program.*, 162:35–54, 2018.

[10] Andrea Arcuri and Lionel Briand. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verification and Reliability*, 24(3):219–250, 2014.

[11] Behrad Bagheri, Shanhu Yang, Hung-An Kao, and Jay Lee. Cyber-physical systems architecture for self-aware machines in industry 4.0 environment. *IFAC-PapersOnLine*, 48(3):1622–1627, 2015. 15th IFAC Symposium onInformation Control Problems in-Manufacturing.

[12] Radhakisan Baheti and Helen Gill. Cyber-physical systems. *The impact of control technology*, 12(1):161–166, 2011.

[13] Naviyn Prabhu Balakrishnan, Gade Pandu Rangaiah, and Lakshminarayanan Samavedham. Review and analysis of blood glucose (bg) models for type 1 diabetic patients. *Industrial & Engineering Chemistry Research*, 50(21):12041–12066, 2011.

[14] Elias Bareinboim and Judea Pearl. Controlling selection bias in causal inference. In Neil D. Lawrence and Mark Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 100–108, La Palma, Canary Islands, 21–23 Apr 2012. PMLR.

[15] Earl T Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. The oracle problem in software testing: A survey. *IEEE transactions on software engineering*, 41(5):507–525, 2014.

[16] Yogesh Barve, Pranav Karve, Aniruddha Gokhale, and Sankaran Mahadevan. Research challenges in the design and composition of surrogate models for robust cps: position paper. In *Proceedings of the Workshop on Design Automation for CPS and IoT*, Destion '21, page 26–29, New York, NY, USA, 2021. Association for Computing Machinery.

[17] Tadej Battelino et al. Clinical Targets for Continuous Glucose Monitoring Data Interpretation: Recommendations From the International Consensus on Time in Range. *Diabetes Care*, 42(8):1593–1603, 06 2019.

[18] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[19] Wayne Biever, Linda Angell, and Sean Seaman. Automated driving system collisions: Early lessons. *Human factors*, 62(2):249–259, 2020.

[20] Bergthor Björnsson et al. Digital twins to personalize medicine. *Genome Medicine*, 12(1):4, Dec 2019.

[21] Mercedes J. Burnside et al. Open-source automated insulin delivery in type 1 diabetes. *New England Journal of Medicine*, 387(10):869–881, 2022. PMID: 36069869.

[22] T. Y. Chen, S. C. Cheung, and S. M. Yiu. Metamorphic testing: A new approach for generating next test cases, 2020.

[23] Runxiang Cheng, Lingming Zhang, Darko Marinov, and Tianyin Xu. Test-case prioritization for configuration testing. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2021, page 452–465, New York, NY, USA, 2021. Association for Computing Machinery.

[24] Herman Chernoff and E. L. Lehmann. The Use of Maximum Likelihood Estimates in χ^2 Tests for Goodness of Fit. *The Annals of Mathematical Statistics*, 25(3):579 – 586, 1954.

[25] Carlos Cinelli and Chad Hazlett. Making sense of sensitivity: Extending omitted variable bias. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(1):39–67, 2020.

[26] Mehmet Hazar Cintuglu, Osama A. Mohammed, Kemal Akkaya, and A. Selcuk Uluagac. A survey on smart grid cyber-physical system testbeds. *IEEE Communications Surveys Tutorials*, 19(1):446–464, 2017.

[27] Andrew G. Clark, Michael Foster, Benedikt Prifling, Neil Walkinshaw, Robert M. Hierons, Volker Schmidt, and Robert D. Turner. Testing causality in scientific modelling software, 2023.

[28] Andrew G. Clark, Michael Foster, Richard Somers, Christopher Wild, Farhad Allian, Robert M. Hierons, Nick Latimer, David Wagg, and Neil Walkinshaw. CITCOM Software Release. 11 2023.

[29] Andrew G. Clark, Michael Foster, Neil Walkinshaw, and Robert M. Hierons. Metamorphic testing with causal graphs. In *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 153–164, 2023.

[30] William L Clarke. The original clarke error grid analysis (ega). *Diabetes technology & therapeutics*, 7(5):776–779, 2005.

[31] R.H. Clayton, O. Bernus, E.M. Cherry, H. Dierckx, F.H. Fenton, L. Mirabella, A.V. Panfilov, F.B. Sachse, G. Seemann, and H. Zhang. Models of cardiac tissue electrophysiology: Progress, challenges and open questions. *Progress in Biophysics and Molecular Biology*, 104(1):22–48, 2011. Cardiac Physiome project: Mathematical and Modelling Foundations.

[32] Jacob Cohen. *Statistical power analysis for the behavioral sciences*. routledge, 2013.

[33] Myra B. Cohen, Joshua Snyder, and Gregg Rothermel. Testing across configurations: implications for combinatorial testing. *SIGSOFT Softw. Eng. Notes*, 31(6):1–9, nov 2006.

[34] Sebastián Contreras, David Medina-Ortiz, Carlos Conca, and Álvaro Olivera-Nappa. A novel synthetic model of the glucose-insulin system for patient-wise inference of physiological parameters from small-size ogtt data. *Frontiers in Bioengineering and Biotechnology*, 8, 2020.

[35] Genevieve Coorey, Gemma A. Figtree, David F. Fletcher, and Julie Redfern. The health digital twin: advancing precision cardiovascular medicine. *Nature Reviews Cardiology*, 18(12):803–804, Dec 2021.

[36] Alessandra Corrado, Giuseppe Scidà, Marilena Vitale, Benedetta Caprio, Giuseppina Costabile, Eric Annuzzi, Giuseppe Della Pepa, Roberta Lupoli, and Lutgarda Bozzetto. Eating habits and sleep quality in individuals with type 1 diabetes on continuous glucose monitoring and insulin pump. *Nutrition, Metabolism and Cardiovascular Diseases*, 34(7):1703–1711, 2024.

[37] Jorge Corral-Acero et al. The 'Digital Twin' to enable the vision of precision cardiology. *European Heart Journal*, 41(48):4556–4564, 03 2020.

[38] Anthony Corso, Robert Moss, Mark Koren, Ritchie Lee, and Mykel Kochenderfer. A survey of algorithms for black-box safety validation of cyber-physical systems. *Journal of Artificial Intelligence Research*, 72:377–428, 2021.

[39] Ermira Daka and Gordon Fraser. A survey on unit testing practices and problems. In *2014 IEEE 25th International Symposium on Software Reliability Engineering*, pages 201–211, 2014.

[40] Mark R Davies, Ken Wang, Gary R Mirams, Antonello Caruso, Denis Noble, Antje Walz, Thierry Lavé, Franz Schuler, Thomas Singer, and Liudmila Polonchuk. Recent developments in using mechanistic cardiac modelling for drug safety evaluation. *Drug Discov Today*, 21(6):924–938, February 2016.

[41] Isabella Degen, Kate Robson Brown, and Zahraa S Abdallah. Studying insulin needs in type 1 diabetes by analysing the openaps data commons. *International Journal of Population Data Science*, 8(3), 2023.

[42] Nilanjan Dey, Amira S Ashour, Fuqian Shi, Simon James Fong, and João Manuel RS Tavares. Medical cyber-physical systems: A survey. *Journal of medical systems*, 42:1–13, 2018.

[43] NHS Digital. Diabetes in england and wales by type 2022/23. `https://www.statista.com/statistics/386742/individuals-with-diabetes-by-type-in-england-and-wales/`. Accessed: 2024-10-7.

[44] Felix Dobslaw, Robert Feldt, and Francisco Gomes de Oliveira Neto. Automated black-box boundary value detection. *PeerJ Computer Science*, 9, 2023.

[45] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[46] J A Douthwaite, B Lesage, M Gleirscher, R Calinescu, J M Aitken, R Alexander, and J Law. A modular digital twinning framework for safety assurance of collaborative robotics. *Front Robot AI*, 8:758099, December 2021.

[47] Lara Edington, Nikolaos Dervilis, Anis Ben Abdessalem, and David Wagg. A time-evolving digital twin tool for engineering dynamics applications. *Mechanical Systems and Signal Processing*, 188:109971, 2023.

[48] Islam T Elgendy, Robert M Hierons, and Phil McMinn. A systematic mapping study of the metrics, uses and subjects of diversity-based testing techniques. *Software Testing, Verification and Reliability*, 35(2):e1914, 2025.

[49] Waguih ElMaraghy, Hoda ElMaraghy, Tetsuo Tomiyama, and Laszlo Monostori. Complexity in engineering design and manufacturing. *CIRP Annals*, 61(2):793–814, 2012.

[50] Tolga Erol, Arif Furkan Mendi, and Dilara Doğan. The digital twin revolution in healthcare. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–7, 2020.

[51] Jonathan Eyre et al. Untangling the requirements of a digital twin. Technical report, Advanced Manufacturing Research Centre, 2021.

[52] Andrea Facchinetti, Simone Del Favero, Giovanni Sparacino, Jessica R. Castle, W. Kenneth Ward, and Claudio Cobelli. Modeling the glucose sensor error. *IEEE Transactions on Biomedical Engineering*, 61(3):620–629, 2014.

[53] Michael P Fay and Michael A Proschan. Wilcoxon-Mann-Whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules. *Stat Surv*, 4:1–39, 2010.

[54] R.S. Freedman. Testability of software components. *IEEE Transactions on Software Engineering*, 17(6):553–564, 1991.

[55] Zhicheng Fu, Chunhui Guo, Shangping Ren, Yizong Ou, and Lui Sha. Modeling and integrating human interaction assumptions in medical cyber-physical system design. In *2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 373–378, 2017.

[56] Aidan Fuller, Zhong Fan, Charles Day, and Chris Barlow. Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, 8:108952–108971, 2020.

[57] David C Funder and Daniel J Ozer. Evaluating effect size in psychological research: Sense and nonsense. *Advances in methods and practices in psychological science*, 2(2):156–168, 2019.

[58] Carlo A Furia, Richard Torkar, and Robert Feldt. Towards causal analysis of empirical software engineering data: The impact of programming languages on coding competitions. *ACM Transactions on Software Engineering and Methodology*, 33(1):1–35, 2023.

[59] Clara Furió-Novejarque et al. Modeling the effect of glucagon on endogenous glucose production in type 1 diabetes: On the role of glucagon receptor dynamics. *Computers in Biology and Medicine*, 154:106605, 2023.

[60] Ahmed Fawzy Gad. Pygad: An intuitive genetic algorithm python library. *Multimedia Tools and Applications*, pages 1–14, 2023.

[61] Chuanchao Gao, Heejong Park, and Arvind Easwaran. An anomaly detection framework for digital twin driven cyber-physical systems. In *ICCPS 2021 - Proceedings of the 2021 ACM/IEEE 12th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2021)*, pages 44–54. Association for Computing Machinery, Inc, 5 2021.

[62] Luca Giamattei, Antonio Guerriero, Roberto Pietrantuono, and Stefano Russo. Causality-driven testing of autonomous driving systems. *ACM Transactions on Software Engineering and Methodology*, 33(3):1–35, 2024.

[63] Clark Glymour and Kun Zhang. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:418407, 2019.

[64] Tushar Goel, Raphael T. Haftka, Wei Shyy, and Nestor V. Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33(3):199–216, Mar 2007.

[65] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Matthew A Wright, Joseph E Gonzalez, and Ion Stoica. Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8806–8813. IEEE, 2021.

[66] Elizabeth González-Estrada and Waldenia Cosmes. Shapiro–wilk test for skew normal distributions based on data transformations. *Journal of Statistical Computation and Simulation*, 89(17):3258–3272, 2019.

[67] Michael Grieves. *Product Lifecycle Management: Driving the next generation of lean thinking*. McGraw-Hill Professional, 2005.

[68] Nasos Grigoropoulos and Spyros Lalis. Simulation and digital twin support for managed drone applications. In *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8, 2020.

[69] Kyeonghye Guk, Gaon Han, Jaewoo Lim, Keunwon Jeong, Taejoon Kang, Eun-Kyung Lim, and Juyeon Jung. Evolution of wearable devices with real-time disease monitoring for personalized healthcare. *Nanomaterials*, 9(6), 2019.

[70] Fitash Ul Haq, Donghwan Shin, and Lionel Briand. Efficient online testing for dnn-enabled systems using surrogate-assisted and many-objective optimization. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, page 811–822, New York, NY, USA, 2022. Association for Computing Machinery.

[71] Miguel Hernan and James Robins. *Causal Inference: What If*. Boca Raton: Chapman & Hall/CRC, 2020.

[72] Yuling Hsu, Jing-Ming Chiu, and John S. Liu. Digital twins for industry 4.0 and beyond. In *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 526–530, 2019.

[73] Jiehui Huang, Lishan Lin, Fengcheng Yu, Xuedong He, Wenhui Song, Jiaying Lin, Zhenchao Tang, Kang Yuan, Yucheng Li, Haofan Huang, Zhong Pei, Wenbiao Xian, and Calvin Yu-Chian Chen. Parkinson's severity diagnosis explainable model based on 3d multi-head attention residual network. *Computers in Biology and Medicine*, 170:107959, 2024.

[74] Morgan Stanley Capital International. https://www.msci.com/our-solutions/indexes/gics. *GICS - Global Industry Classification Standard*, 1999.

[75] Bernhard Jaeger, Kashyap Chitta, and Andreas Geiger. Hidden biases of end-to-end driving models. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023.

[76] Andrzej S. Januszewski et al. Insulin micro-secretion in type 1 diabetes and related microrna profiles. *Scientific Reports*, 11(1):11727, Jun 2021.

[77] Philip M. Jedrzejewski, Ioscani Jimenez Del Val, Antony Constantinou, Anne Dell, Stuart M. Haslam, Karen M. Polizzi, and Cleo Kontoravdi. Towards controlling the glycoform: A model framework linking extracellular metabolites to antibody glycosylation. *International Journal of Molecular Sciences*, 15(3):4492–4522, 2014.

[78] Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering*, 37(5):649–678, 2011.

[79] R. Jones and S Dawson. Strategies for ensuring safety with industrial robot systems. *Omega*, 14(4):287–297, 1986.

[80] Paul C. Jorgensen. *Software Testing: a Craftsman's Approach*. Auerbach Publications, fourth edition, 2013.

[81] Nadya Kagansky, Shmuel Levy, and Hilla Knobler. The Role of Hyperglycemia in Acute Stroke. *Archives of Neurology*, 58(8):1209–1212, 08 2001.

[82] Maged N Kamel Boulos and Peng Zhang. Digital twins: from personalised medicine to precision public health. *Journal of Personalized Medicine*, 11(8):745, 2021.

[83] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, Feb 2021.

[84] Luke Keele. The statistics of causal inference: A view from political methodology. *Political Analysis*, pages 313–335, 2015.

[85] Sagheer Khan, Aaesha Alzaabi, Tharmalingam Ratnarajah, and Tughrul Arslan. Novel statistical time series data augmentation and machine learning based classification of unobtrusive respiration data for respiration digital twin model. *Computers in Biology and Medicine*, 168:107825, 2024.

[86] Dimitrios Kiagias, Giulia Russo, Giuseppe Sgroi, Francesco Pappalardo, and Miguel A Juárez. Bayesian augmented clinical trials in tb therapeutic vaccination. *Frontiers in Medical Technology*, 3:719380, 2021.

[87] Anna H. Kingston, Amy J. Morgan, Anthony F. Jorm, Kate Hall, Laura M. Hart, Claire M. Kelly, and Dan I. Lubman. Helping someone with problem drug use: a delphi consensus study of consumers, carers, and clinicians. *BMC Psychiatry*, 11(1):3, Jan 2011.

[88] Keiichi Kodama, Damon Tojjar, Satoru Yamada, Kyoko Toda, Chirag J. Patel, and Atul J. Butte. Ethnic Differences in the Relationship Between Insulin Sensitivity and Insulin Response: A systematic review and meta-analysis. *Diabetes Care*, 36(6):1789–1796, 05 2013.

[89] Boris P. Kovatchev, Linda A. Gonder-Frederick, Daniel J. Cox, and William L. Clarke. Evaluating the Accuracy of Continuous Glucose-Monitoring Sensors: Continuous glucose–error grid analysis illustrated by TheraSense Freestyle Navigator data . *Diabetes Care*, 27(8):1922–1928, 08 2004.

[90] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11):1016–1022, 2018.

[91] Yigit Kucuk, Tim A. D. Henderson, and Andy Podgurski. Improving fault localization by integrating value and predicate based causal inference techniques, 2021.

[92] D Richard Kuhn, Itzel Dominguez Mendoza, Raghu N Kacker, and Yu Lei. Combinatorial coverage measurement concepts and applications. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, pages 352–361. IEEE, 2013.

[93] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[94] Choong Ho Lee and Hyung-Jin Yoon. Medical big data: promise and challenges. *Kidney Res Clin Pract*, 36(1):3–11, March 2017.

[95] Mengyun Lei, Beisi Lin, Ping Ling, Zhigu Liu, Daizhi Yang, Hongrong Deng, Xubin Yang, Jing Lv, Wen Xu, and Jinhua Yan. Efficacy and safety of android artificial pancreas system use at home among adults with type 1 diabetes mellitus in china: protocol of a 26-week, free-living, randomised, open-label, two-arm, two-phase, crossover trial. *BMJ Open*, 13(8):e073263, August 2023.

[96] Doug Leigh. *SWOT Analysis*, chapter 5, pages 115–140. John Wiley & Sons, Ltd, 2009.

[97] Dana Lewis. Openaps data commons. `https://OpenAPS.org/data-commons`, 2023.

[98] Dana Lewis. Openaps oref0. `https://github.com/openaps/oref0`, 2023.

[99] Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. Many-objective evolutionary algorithms: A survey. *ACM Comput. Surv.*, 48(1), sep 2015.

[100] Michelle L Litchman, Heather R Walker, Caroline Fitzgerald, Mariana Gomez Hoyos, Dana Lewis, and Perry M Gee. Patient-driven diabetes technologies: sentiment and personas of the #wearenotwaiting and #openaps movements. *Journal of diabetes science and technology*, 14(6):990–999, 2020.

[101] Mengnan Liu, Shuiliang Fang, Huiyue Dong, and Cunzhi Xu. Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, 58:346–361, 2021. Digital Twin towards Smart Manufacturing and Industry 4.0.

[102] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[103] Yu Liu. Overview of some theoretical approaches for derivation of the monod equation. *Applied Microbiology and Biotechnology*, 73(6):1241–1250, Jan 2007.

[104] Yiling Lou, Junjie Chen, Lingming Zhang, and Dan Hao. Chapter one - a survey on regression test-case prioritization. In Atif M. Memon, editor, *Advances in Computers*, volume 113, pages 1–46. Elsevier, 2019.

[105] Octavio Loyola-Gonzalez. Black-box vs. White-Box: Understanding their advantages and weaknesses from a practical point of view. *IEEE Access*, 7:154096–154113, 2019.

[106] Lin Lu, Jiayao Zhang, Yi Xie, Fei Gao, Song Xu, Xinghuo Wu, and Zhewei Ye. Wearable health devices in health care: Narrative systematic review. *JMIR Mhealth Uhealth*, 8(11):e18907, Nov 2020.

[107] D. Maclay. Simulation gets into the loop. *IEE Review*, 43(3):109–112, 1997.

[108] Nathaniel R Maddux, Austin L Daniels, and Theodore W Randolph. Microflow imaging analyses reflect mechanisms of aggregate formation: comparing protein particle data sets using the kullback–leibler divergence. *Journal of pharmaceutical sciences*, 106(5):1239–1248, 2017.

[109] Dani Mann. Omnipod 5 insulin system reports decimal point glitch, Dec 2023.

[110] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50 – 60, 1947.

[111] Alberto Maran et al. Continuous Subcutaneous Glucose Monitoring in Diabetic Patients: A multicenter analysis. *Diabetes Care*, 25(2):347–352, 02 2002.

[112] Edwin P. Martens, Wiebe R. Pestman, Anthonius de Boer, Svetlana V. Belitser, and Olaf H. Klungel. Instrumental variables: Application and limitations. *Epidemiology*, 17(3), 2006.

[113] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[114] Phil McMinn. Search-based software test data generation: a survey. *Software Testing, Verification and Reliability*, 14(2):105–156, 2004.

[115] Phil McMinn. Search-based software testing: Past, present and future. In *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, pages 153–163, 2011.

[116] Claudio Menghi, Shiva Nejati, Lionel Briand, and Yago Isasi Parache. Approximation-refinement testing of compute-intensive cyber-physical models: An approach based on system identification. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 372–384, New York, NY, USA, 2020. Association for Computing Machinery.

[117] Gary R. Mirams, Pras Pathmanathan, Richard A. Gray, Peter Challenor, and Richard H. Clayton. Uncertainty and variability in computational and mathematical models of cardiac physiology. *The Journal of Physiology*, 594(23):6833–6847, 2016.

[118] John D. Musa. The operational profile. In Süleyman Özekici, editor, *Reliability and Maintenance of Complex Systems*, pages 333–344, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[119] Brady Neal. Introduction to causal inference. *Course Lecture Notes (draft)*, 2020.

[120] Shiva Nejati, Lev Sorokin, Damir Safin, Federico Formica, Mohammad Mahdi Mahboob, and Claudio Menghi. Reflections on surrogate-assisted search-based testing: A taxonomy and two replication studies based on industrial adas and simulink models. *Information and Software Technology*, 163:107286, 2023.

[121] Patrick Ngatchou, Anahita Zarei, and A El-Sharkawi. Pareto multi objective optimization. In *Proceedings of the 13th international conference on, intelligent systems application to power systems*, pages 84–91. IEEE, 2005.

[122] Steven A. Niederer, Joost Lumens, and Natalia A. Trayanova. Computational models in cardiology. *Nature Reviews Cardiology*, 16(2):100–111, Feb 2019.

[123] Shagofah Noor, Omid Tajik, and Jawad Golzar. Simple random sampling. *International Journal of Education & Language Studies*, 1(2):78–82, 2022.

[124] Everton Note Narciso, Márcio Delamaro, and Fátima Nunes. Test case selection: A systematic literature review. *International Journal of Software Engineering and Knowledge Engineering*, 24:653–676, 05 2014.

[125] OpenAPS. Facebook theloopedgroup `https://www.facebook.com/groups/TheLoopedGroup`.

[126] OpenAPS. Issues openaps/oref0 `https://github.com/openaps/oref0/issues`.

[127] OpenAPS. Loop and learn `https://www.loopnlearn.org/starting-loop/`.

[128] OpenAPS. Openaps dev `https://groups.google.com/g/openaps-dev?pli=1`.

[129] OpenAPS. Openaps gitter `https://app.gitter.im/#/room/#nightscout_intend-to-bolus:gitter.im`.

[130] Openaps. Openaps.org - #wearenotwaiting to reduce the burden of type 1 diabetes `https://openaps.org/`.

[131] OpenAPS. Understanding your preferences and safety settings `https://openaps.readthedocs.io/en/latest/docs/While%20You%20Wait%20For%20Gear/preferences-and-safety-settings.html#oref1-related-preferences`, 2017.

[132] Judea Pearl. causal graphs for empirical research. *Biometrika*, 82(4):669–688, 1995.

[133] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009.

[134] Judea Pearl. On measurement bias in causal inference, 2012.

[135] Judea Pearl et al. Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146, 2009.

[136] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.

[137] Chao Chung Peng and Yi Ho Chen. Digital twins-based online monitoring of tfe-731 turbofan engine using fast orthogonal search. *IEEE Systems Journal*, 2021.

[138] Yingzhou Peng, Shuai Zhao, and Huai Wang. A digital twin based estimation method for health indicators of dc-dc converters. *IEEE Transactions on Power Electronics*, 36:2105–2118, 2 2021.

[139] Yingzhou Peng, Shuai Zhao, and Huai Wang. A digital twin based estimation method for health indicators of dc–dc converters. *IEEE Transactions on Power Electronics*, 36(2):2105–2118, 2021.

[140] Luis Piardi, Paulo Leitão, and André Schneider de Oliveira. Fault-tolerance in cyber-physical systems: Literature review and challenges. In *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 29–34, 2020.

[141] Andy Podgurski and Yiğit Küçük. Counterfault: Value-based fault localization by modeling and predicting counterfactual outcomes. In *2020 IEEE International Conference on Software Maintenance and Evolution*, pages 382–393. IEEE, 2020.

[142] Jay W Porter. *Sleep, exercise, and insulin sensitivity.* PhD thesis, University of Missouri-Columbia, 2020.

[143] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: The next computing revolution. In *Design Automation Conference*, pages 731–736, 2010.

[144] Paul Ralph, Sebastian Baltes, Domenico Bianculli, Yvonne Dittrich, Michael Felderer, Robert Feldt, Antonio Filieri, Carlo Alberto Furia, Daniel Graziotin, Pinjia He, Rashina Hoda, Natalia Juristo, Barbara A. Kitchenham, Romain Robbes, Daniel Méndez, Jefferson Seide Molléri, Diomidis Spinellis, Miroslaw Staron, Klaas-Jan Stol, Damian A. Tamburri, Marco Torchiano, Christoph Treude, Burak Turhan, and Sira Vegas. ACM SIGSOFT empirical standards. *CoRR*, abs/2010.03525, 2020.

[145] Rakesh Rana and Richa Singhal. Chi-square test and its application in hypothesis testing. *Journal of Primary Care Specialties*, 1(1):69–71, 2015.

[146] James Reason. *Human error.* Cambridge university press, 1990.

[147] Danilo J. Rezende, Ivo Danihelka, George Papamakarios, Nan Rosemary Ke, Ray Jiang, Theophane Weber, Karol Gregor, Hamza Merzic, Fabio Viola, Jane Wang, Jovana Mitrovic, Frederic Besse, Ioannis Antonoglou, and Lars Buesing. Causally correct partial models for reinforcement learning, 2020.

[148] P Runeson, M Höst, A Rainer, and B Regnell. *Design of the Case Study*, chapter 3, pages 23–45. John Wiley & Sons, Ltd, 2012.

[149] Radhya Sahal, Saeed H. Alsamhi, and Kenneth N. Brown. Personal digital twin: A close look into the present and a step towards the future of personalised healthcare industry. *Sensors (Basel, Switzerland)*, 22(15):5918, 2022.

[150] Elizabeth R. Seaquist, John Anderson, Belinda Childs, Philip Cryer, Samuel Dagogo-Jack, Lisa Fish, Simon R. Heller, Henry Rodriguez, James Rosenzweig, and Robert Vigersky. Hypoglycemia and Diabetes: A Report of a Workgroup of the American Diabetes Association and The Endocrine Society. *Diabetes Care*, 36(5):1384–1395, 04 2013.

[151] Sergio Segura, Gordon Fraser, Ana B Sanchez, and Antonio Ruiz-Cortés. A survey on metamorphic testing. *IEEE Transactions on software engineering*, 42(9):805–824, 2016.

[152] Arsalan Shahid and Dana M. Lewis. Large-scale data analysis for glucose variability outcomes with open-source automated insulin delivery systems. *Nutrients*, 14(9), 2022.

[153] Arsalan Shahid and Dana M Lewis. Large-scale data analysis for glucose variability outcomes with open-source automated insulin delivery systems. *Nutrients*, 14(9):1906, 2022.

[154] Kfir Sharabi, Clint D J Tavares, Amy K Rines, and Pere Puigserver. Molecular pathophysiology of hepatic glucose production. *Mol Aspects Med*, 46:21–33, November 2015.

[155] Sina Sheikholeslami. Ablation programming for machine learning, 2019.

[156] Akhil Shetty, Mengqiao Yu, Alex Kurzhanskiy, Offer Grembek, Hamidreza Tavafoghi, and Pravin Varaiya. Safety challenges for autonomous vehicles in the absence of connectivity. *Transportation Research Part C: Emerging Technologies*, 128:103133, 2021.

[157] Qingkai Shi, Zhenyu Chen, Chunrong Fang, Yang Feng, and Baowen Xu. Measuring the diversity of a test set with distance entropy. *IEEE Transactions on Reliability*, 65(1):19–27, 2015.

[158] Richard Somers, Neil Walkinshaw, Robert Hierons, Jackie Elliott, Ahmed Iqbal, and Emma Walkinshaw. Configuration testing of an artificial pancreas system using a digital twin. 2024.

[159] Richard J. Somers, Andrew G. Clark, Neil Walkinshaw, and Robert M. Hierons. Reliable counterparts: efficiently testing causal relationships in digital twins. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, MODELS '22, page 468–472, New York, NY, USA, 2022. Association for Computing Machinery.

[160] Richard J. Somers, James A. Douthwaite, David J. Wagg, Neil Walkinshaw, and Robert M. Hierons. Digital-twin-based testing for cyber–physical systems: A systematic literature review. *Information and Software Technology*, 156:107145, 2023.

[161] Matt Staats, Michael W. Whalen, and Mats P.E. Heimdahl. Programs, tests, and oracles: the foundations of testing revisited. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, page 391–400, New York, NY, USA, 2011. Association for Computing Machinery.

[162] ISO Standards. Iso/iec/ieee international standard - systems and software engineering–vocabulary. *ISO/IEC/IEEE 24765:2017(E)*, pages 1–541, 2017.

[163] Agostino Sturaro, Simone Silvestri, Mauro Conti, and Sajal K Das. A realistic model for failure propagation in interdependent cyber-physical systems. *IEEE Transactions on Network Science and Engineering*, 7(2):817–831, 2018.

[164] Mark A. Sujan, Ibrahim Habli, Tim P. Kelly, Simone Pozzi, and Christopher W. Johnson. Should healthcare providers do safety cases? lessons from a cross-industry review of safety case practices. *Safety Science*, 84:181–189, 2016.

[165] Jimin Tan, Jianan Yang, Sai Wu, Gang Chen, and Jake Zhao. A critical look at the current train/test split in machine learning. *arXiv preprint arXiv:2106.04525*, 2021.

[166] Meseret N Teferra. ISO 14971-medical device risk management standard. *International Journal of Latest Research in Engineering and Technology (IJLRET)*, 3(3):83–87, 2017.

[167] Julian L. Tekaat, Harald Anacker, and Roman Dumitrescu. The paradigm of design thinking and systems engineering in the design of cyber-physical systems: A systematic literature review. In *2021 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–8, 2021.

[168] Hood Thabit and Roman Hovorka. Coming of age: the artificial pancreas for type 1 diabetes. *Diabetologia*, 59(9):1795–1805, June 2016.

[169] Matthew S Thiese, Brenden Ronna, and Ulrike Ott. P value interpretations and considerations. *J Thorac Dis*, 8(9):E928–E931, September 2016.

[170] Guilherme Luz Tortorella, Flávio Sanson Fogliatto, Alejandro Mac Cawley Vergara, Roberto Vassolo, and Rapinder Sawhney. Healthcare 4.0: trends, challenges and research directions. *Production Planning & Control*, 31(15):1245–1260, 2020.

[171] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[172] D. J. Wagg, K. Worden, R. J. Barthorpe, and P. Gardner. Digital Twins: State-of-the-Art and Future Directions for Modeling and Simulation in Engineering Dynamics Applications. *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, 6(3), 05 2020. 030901.

[173] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, page 3645–3649. IEEE Press, 2017.

[174] K. Worden, R.J. Barthorpe, E.J. Cross, N. Dervilis, G.R. Holmes, G. Manson, and T.J. Rogers. On evolutionary system identification with applications to nonlinear benchmarks. *Mechanical Systems and Signal Processing*, 112:194–232, 2018.

[175] Keith Worden, E. J. Cross, P. Gardner, R. J. Barthorpe, and D. J. Wagg. On digital twins, mirrors and virtualisations. In *Model Validation and Uncertainty Quantification, Volume 3*, pages 285–295. Springer, 2020.

[176] Zekai Wu, Sihui Luo, Xueying Zheng, Yan Bi, Wen Xu, Jinhua Yan, Daizhi Yang, and Jianping Weng. Use of a do-it-yourself artificial pancreas system is associated with

better glucose management and higher quality of life among adults with type 1 diabetes. *Therapeutic Advances in Endocrinology and Metabolism*, 11:2042018820950146, 2020. PMID: 32922721.

[177] Junqin Xu and Jihui Zhang. Exploration-exploitation tradeoffs in metaheuristics: Survey and analysis. In *Proceedings of the 33rd Chinese Control Conference*, pages 8633–8638, 2014.

[178] Tianyin Xu and Owolabi Legunsen. Configuration testing: Testing configuration values as code and with code. *arXiv preprint arXiv:1905.12195*, 2019.

[179] Yan Xu, Yanming Sun, Xiaolong Liu, and Yonghua Zheng. A digital-twin-assisted fault diagnosis using deep transfer learning. *IEEE Access*, 7:19990–19999, 2019.

[180] Song Yan, Chunxi Huang, and Dengbo He. A comparison of patterns and contributing factors of adas and ads involved crashes. *Journal of Transportation Safety & Security*, pages 1–28, 2023.

[181] Srikanth Yoginath, Varisara Tansakul, Supriya Chinthavali, Curtis Taylor, Joshua Hambrick, Philip Irminger, and Kalyan Perumalla. On the effectiveness of recurrent neural networks for live modeling of cyber-physical systems. In *Proceedings - IEEE International Conference on Industrial Internet Cloud, ICII 2019*, pages 309–317. Institute of Electrical and Electronics Engineers Inc., 11 2019.

[182] Qingmin Yu, Ying Huang, Yang Liu, Sicong Yu, and Shijie Wang. Research on application of information model in wind turbine fault diagnosis. In *2021 2nd International Conference on Artificial Intelligence in Electronics Engineering*, AIEE 2021, page 67–74, New York, NY, USA, 2021. Association for Computing Machinery.

[183] Zhalama, Jiji Zhang, and Wolfgang Mayer. Weakening faithfulness: some heuristic causal discovery algorithms. *International Journal of Data Science and Analytics*, 3(2):93–104, Mar 2017.

[184] Xin Zhou, Xiaodong Gou, Tingting Huang, and Shunkun Yang. Review on testing of cyber physical systems: Methods and testbeds. *IEEE Access*, 6:1–1, 09 2018.

[185] Zexuan Zhu, Chao Liu, and Xun Xu. Visualisation of the digital twin data in manufacturing by using augmented reality. *Procedia CIRP*, 81:898–903, 2019. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.

Appendix A.
Appendix - Evaluation

Table 11.: Surrogate model training times between the modified and unmodified ensemble models.

Surrogate	1	2	3	4	5	6	7	8	9	10	Mean (s)	Required (s)
Modified	0.033	0.035	0.025	0.026	0.018	0.022	0.025	0.063	0.022	0.026	0.029	5486
Unmodified	38.035	37.858	40.446	41.278	38.755	37.068	39.318	38.075	40.178	34.949	38.596	7722048

For our oref0 evaluation in Chapter 7, we decided to make a modification to the associative model used in both the ensemble and hybrid approaches. Since the oref0 evaluation was so large, the neural network aspect of the associative model became computationally infeasible. We demonstrate this in Table 11, which shows that training the neural network for each iteration across all 924 traces would have added ∼90 days of GPU computation time.

For our study, we found this infeasible. Instead, we modified the ensemble model by removing the neural network for this study. To ensure this adaptation was fair and did not skew the comparison of techniques, we ensured that the entire study used this modification.

Appendix B.
Appendix - Digital Twin

B.1. Model Modification

The model implementation by Contreras et al. [34] produced negative hepatic blood glucose (G_{prod}) at blood glucose levels less than the steady-state blood glucose level (G_b). We modified the original model by introducing a Monod equation [103] for G_{prod} with respect to $-G$. This is inline with prior work that has used Monod equations to represent hepatic glucose production [77] as this approach a simple yet effective representation of this behaviour [154]. Equation (B.1) introduces the hepatic glucose growth rate (k_μ), the hepatic glucose limit (k_λ) and the hepatic glucose production (G_{prod0}) at the glucose steady-state (G_b).

$$G_{prod} = \frac{k_\lambda (G_b - G)}{k_\mu + (G_b - G)} + G_{prod0} \tag{B.1}$$

Table 12.: The meaning of each constant from $K_x(T)$ representing the blood glucose-insulin dynamics of a person.

Constant	Meaning
k_{js}	Rate of carbohydrate movement from stomach to jejunum
k_{gj}	Rate of carbohydrate absorption from jejunum to blood
k_{jl}	Rate of carbohydrate movement from jejunum to ilium
k_{gl}	Rate of carbohydrate absorption from ilium to blood
k_{xg}	Rate of blood glucose basal uptake
k_{xgi}	Rate of blood glucose insulin sensitivity uptake
k_{xi}	Rate of insulin degradation
τ	Time taken for ilium to receive glucose from jejunum
η	Bioavailability of glucose absorbed in the intestine
k_λ	Hepatic glucose production limit
k_μ	Hepatic glucose production growth rate
G_{prod0}	Hepatic glucose production at the steady state

B.2. Fitting Algorithm Choice

When deciding the technique for fitting the digital twin in Chapter 6, we evaluated 4 different meta-heuristic algorithms. Each algorithm was fitted using the same 10 randomly selected datasets and executed 10 times to account for randomness. Figure 27 presents the fitness values for each meta-heuristic algorithm. It can be seen that the generic algorithm resulted in the highest average fitness across traces. We use this to justify our choice of fitting algorithm. The code used to generate this figure can be found in *fitness_checker.py* in our replication package (S2 in Section 1.1.2).

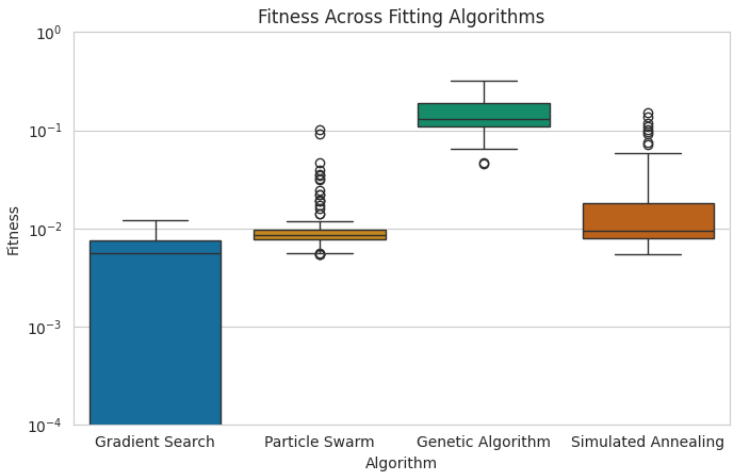


Figure 27.: The fitness values when fitting the digital twin across 4 different fitting algorithms. Note, the scale of fitness is logarithmic.