

37th International Conference on Principles of Diagnosis and Resilient Systems

DX 2026, September 1–3, 2026, Cork, Ireland

Edited by

Ingo Pill

Marina Zanella

Gregory Provan



Editors

Ingo Pill 

Graz University of Technology, Austria
ingo.pill@gmail.com

Marina Zanella 

University of Brescia, Italy
marina.zanella@unibs.it

Gregory Provan 

University College Cork, Ireland
gprovan@cs.ucc.ie

ACM Classification 2012

Computing methodologies → Knowledge representation and reasoning

ISBN 978-3-95977-455-0

Published online and open access by

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <https://www.dagstuhl.de/dagpub/978-3-95977-455-0>.

Publication date

October, 2026

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists all publications of this volume in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <https://portal.dnb.de>.

License

This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0):
<https://creativecommons.org/licenses/by/4.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/OASlcs.DX.2026.0

ISBN 978-3-95977-455-0

ISSN 1868-8969

<https://www.dagstuhl.de/oasics>

OASlcs – OpenAccess Series in Informatics

OASlcs is a series of high-quality conference proceedings across all fields in informatics. OASlcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

Editorial Board

- Daniel Cremers (TU München, Germany)
- Barbara Hammer (Universität Bielefeld, Germany)
- Marc Langheinrich (Università della Svizzera Italiana – Lugano, Switzerland)
- Dorothea Wagner (*Editor-in-Chief*, Karlsruher Institut für Technologie, Germany)

ISSN 1868-8969

<https://www.dagstuhl.de/oasics>

Contents

Preface	
<i>Ingo Pill, Marina Zanella, and Gregory Provan</i>	0:vii
Conference Organization	
.....	0:ix
List of Authors	
.....	0:xi

Regular Papers

A Realization-Theoretic Foundation for Fault Diagnosis, with Diagnosability Equivalence Predictions	
<i>Gregory Provan</i>	1:1–1:20
An Explainable GNN Framework for Component-Level Anomaly Diagnosis	
<i>Sena Ozgunay, Louise Travé-Massuyès, Jean-Michel Loubes, and Raul Sena Ferreira</i>	2:1–2:17
Dissecting NetDoktor’s Online Questionnaire with Entropy, Tree Simplification & Logical Abduction	
<i>Alexander Perko, Emir Mujić, and Franz Wotawa</i>	3:1–3:20
Early Failure Prediction from Near-Anomaly Detection: A Proactive Approach	
<i>Léa Billet, Louise Travé-Massuyès, Elodie Chanthery, and Alexandre Gaffet</i>	4:1–4:18
FELIX: Model Repair in Numeric Planning	
<i>Nir Aharoni, Yarin Benyamin, Shiwali Mohan, and Roni Stern</i>	5:1–5:18
Geometric Feature Selection for Interpretable SVM Classifiers in Cyber-Physical Systems	
<i>Leonie Hatte, Pauline Ribot, and Elodie Chanthery</i>	6:1–6:20
HSU TwinFlow: A Living Benchmark for Evaluating Anomaly Detection and Diagnosis Methods in Cyber-Physical Production Systems Using Digital-Twin-Generated Data	
<i>Nemanja Hranisavljevic, Alexander Diedrich, Lukas Moddemann, Domenic Schaeffer, Frank Marek, Ingo Pill, and Oliver Niggemann</i>	7:1–7:14
Hybrid Fault Detection and Isolation for the Fuel Turbopump Subsystem of the LOX/LNG Expander-Bleed Rocket Engine LUMEN	
<i>Eldin Kurudzija, Marcos Quinones-Grueiro, Kai Dresia, Austin Coursey, Gautam Biswas, Jan Deeken, and Günther Waxenegger-Wilfing</i>	8:1–8:20
On the Practical Utility of Diagnoses	
<i>Ingo Pill and Johan de Kleer</i>	9:1–9:20
Planning Domain Repair Using Preferred Plans and Background Knowledge	
<i>Thomas Eckstein and Gerald Steinbauer-Wagner</i>	10:1–10:22

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

QSIM-Guided Weak Supervision for Fault Detection in Cyber-Physical Systems <i>Ankita Das, Roxane Koitz-Hristov, and Franz Wotawa</i>	11:1–11:20
Reinforcement Learning-Driven Predictive Maintenance Planning via Timed Automata Modeling <i>Ferhat Tamssaouet, Pauline Ribot, and Yannick Pencolé</i>	12:1–12:20
Statistical Foundations of Local Geometric Fault Detection: The Curvature Channel <i>Gregory Provan</i>	13:1–13:20
Using SLMs and LLMs for Diagnosis <i>Sophia Plank and Franz Wotawa</i>	14:1–14:17
DX Competition	
Physics-Informed ML Diagnostic Engine: DXC’26 LiU-ICE Track Solution <i>Antoni Rogowski and Anna Sztzyber-Betley</i>	15:1–15:20
Short Papers	
Counterfactual Fault Responsibility in a Dynamic Structural Causal Model of a DC Motor <i>Emir Mujić and Franz Wotawa</i>	16:1–16:14
When Dynamic Slicing Helps and Hurts Spectrum-Based Fault Localization: Evidence from an Expanded TCAS Benchmark <i>Iulia Nica and Franz Wotawa</i>	17:1–17:15
Extended Abstracts	
Summary Of: Modelling Cyber-Physical Systems for Fault Diagnosis <i>Alexander Diedrich, Mattias Krysander, René Heesch, and Oliver Niggemann</i>	18:1–18:6
Uncertainty-Aware Fault Diagnosis of Unknown Faults Using Ensemble-Based NODE Residuals <i>Daniel Jung and Theodor Westny</i>	19:1–19:6

■ Preface

In 2026, the 37th International Conference on Principles of Diagnosis and Resilient Systems (DX'26) was hosted by University College Cork from September 1st to 3rd in Cork, Ireland. The DX conference has been a leading forum to present and discuss the latest research, experience reports, and emerging ideas related to diagnosis and resilient systems. Since DX is application-agnostic, the research presented covers a wide range of systems, from physical via biological to computational ones, represented as either abstract or detailed models. Starting with the 2024 edition, DX not only evolved from a workshop to a full conference, but also expanded its focus to include resilience systems. Directly connected to diagnostic issues, resilience in a system refers to a system's inherent ability to maintain its essential operations when faced with both expected and unforeseen challenges. Related aspects of interest to DX include resilient design as well as approaches that enable operational resilience. The topics covered in the DX conference thus include, but are not limited to:

- Monitoring, detection, diagnosis and mitigation of faults, unexpected issues and anomalies
- Formal theories and symbolic, sub-symbolic, as well as hybrid approaches to diagnosis
- Data-driven and learning-enabled methods for monitoring and diagnosis
- Connections between diagnosis and other techniques such as decision-making, (re-)planning, (re-)configuration, control, formal verification, and testing
- Theoretical concepts for design and operational resilience
- Designing, developing, and operationalizing resilient systems
- Hardware and software instrumentation for dependable data acquisition and probing
- Development, learning, abstraction, transformation, analysis, optimization, maintenance and transfer of diagnosis models
- Diagnosis in resilient, intelligent, and autonomous systems
- Diagnosis in a distributed, hierarchical, system-of-systems, or multi-agent context

For our 2026 edition, we received 29 submissions, ranging from extended abstracts to full papers and targeting the regular, DX Competition, and PhD panel tracks. This number includes two submissions that have been re-evaluated. For publication in these proceedings, we accepted 19 submissions, which entails an acceptance rate of 65.5%.

The manuscripts in these proceedings demonstrate the wide and innovative scope of research on diagnosis, prognosis, and resilient systems. Contributions include advances in methodology – such as temporal reasoning and planning or uncertainty considerations – as well as the incorporation of large language models for developing diagnostic models. The application areas are just as varied, covering control software, cyber-physical systems, or combustion and space rocket engines. Several studies highlight benchmarking and evaluation frameworks, like the revived DX Competition, a benchmark for fault isolation and diagnosis in production plants, and the systematic consideration or analysis of diagnostic algorithms and derived diagnoses. All these papers exemplify the theoretical depth as well as practical applications of diagnosis research, pointing towards the future integration of AI, multimodal reasoning, and resilience-by-design for enabling complex engineered systems.

A special highlight this year is that TIME, i.e., the International Symposium on Temporal Representation and Reasoning, organized by *Sophie Pinchinat* and *Andrea Orlandini*, chose to co-locate with DX. To honor the co-location, the program on September 2nd was shared between the two events, starting with a keynote given by *Alessandro Cimatti* from FBK, Italy,

with the title *From Runtime Verification to Model-Based Diagnosis*. For the DX program, we could also secure an invited talk on *Engineering Resilience for Critical Infrastructure* given by *Utz Roenig* from UCC, Ireland.

Organizing a conference is always a team effort, so we would like to thank a few people. We start with the program committee members and thank them for their reviews that helped us select the right papers for our program and the subset accepted for these proceedings. We would like to specifically mention also our keynote speaker *Alessandro Cimatti*, the invited speaker *Utz Roedig*, the DX Competition committee (see below), as well as the DX Steering Committee (see below) for their essential contributions to the DX community and this conference. Further thanks go to the DX'26 award committee members, *Franz Wotawa*, *Oliver Niggemann* and *Marina Zanella*, who selected the awardees from the following four full papers that were nominated for receiving a best paper award (in a.o.):

- Early Failure Prediction from Near-Anomaly Detection: A Proactive Approach, *by Léa Billet, Louise Travé-Massuyès, Elodie Chantry and Alexandre Gaffet*
- Hybrid Fault Detection and Isolation for the Fuel Turbopump Subsystem of the LOX/LNG Expander-Bleed Rocket Engine LUMEN, *by Eldin Kurudzija, Kai Dresia, Günther Waxenegger-Wilfing, Marcos Quinones-Grueiro, Austin Coursey and Gautam Biswas*
- On the Practical Utility of Diagnoses, *by Ingo Pill and Johan de Kleer*
- Reinforcement Learning-Driven Predictive Maintenance Planning via Timed Automata Modeling, *by Ferhat Tamssaouet, Pauline Ribot and Yannick Pencolé*

We congratulate the winners of the issued award(s) who were not known at the time of writing this preface. Special thanks go to Dagstuhl, our publisher, and the editors of the OASIs series for collaborating with us to make our technical program available through open-access proceedings. Finally, we thank the authors for contributing to the success of this conference and the high quality of our archival proceedings. Without their work and their choice to present their most recent research at DX, this event would not have been possible.

The DX'26 chairs: Ingo Pill, Gregory Provan, Marina Zanella

■ Conference Organization

DX 2026 Organizing Committee (in a.o.)

- Johan de Kleer, C-Infinity, USA
- Ingo Pill (chair), Graz University of Technology, Austria
- Gregory Provan (local chair), University College Cork, Ireland
- Marcos Quinones-Grueiro, Vanderbilt University, USA
- Marina Zanella (chair), Università di Brescia, Italy

DX Steering Committee (in a.o.)

- Gautam Biswas, Vanderbilt University, Vanderbilt, USA
- Johan de Kleer, C-Infinity, USA
- Meir Kalech, Ben Gurion University of the Negev, Israel
- Oliver Niggemann, Helmut-Schmidt-Universität Hamburg, Germany
- Ingo Pill (chair), Graz University of Technology, Austria
- Louise Travé-Massuyès, LAAS-CNRS / Université de Toulouse, France
- Franz Wotawa, Graz University of Technology, Austria
- Marina Zanella, Università di Brescia, Italy

DX Competition 2026 Committee (in a.o.)

- Johan de Kleer (co-chair, chair COMIC Benchmark), C-Infinity, USA
- Erik Frisk, Linköping University, Sweden
- Daniel Jung (chair LiU-ICE Benchmark), Linköping University, Sweden
- Mattias Krysander, Linköping University, Sweden
- Ingo Pill (chair), Graz University of Technology, Austria
- Michal Syfert (chair SLIDE Benchmark), Warsaw University of Technology, Poland
- Anna Szttyber-Betley, Warsaw University of Technology, Poland

DX 2026 Program Committee (in a.o.)

- Elodie Chantry, LAAS-CNRS / Université de Toulouse, France
- Alessandro Cimatti, Fondazione Bruno Kessler, Italy
- Johan de Kleer, C-Infinity, USA
- Alexander Diedrich, Helmut-Schmidt-Universität Hamburg, Germany
- Görschwin Fey, Technische Universität Hamburg, Germany
- Daniel Jung, Linköping University, Sweden
- Martin Leucker, University of Lübeck, Germany
- Avraham Natan, Ben Gurion University of the Negev, Israel
- Yannick Pencolé, LAAS-CNRS / Université de Toulouse, France
- Ingo Pill (chair), Graz University of Technology, Austria
- Gregory Provan (chair), University College Cork, Ireland
- Belarmino Pulido, Universidad de Valladolid, Spain
- Marcos Quinones-Grueiro, Vanderbilt University, USA
- Pauline Ribot, LAAS-CNRS / Université de Toulouse, France
- Patrick Rodler, Alpen-Adria Universität Klagenfurt, Austria
- Indranil Roychoudhury, Schlumberger Menlo Park, USA

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).
Editors: Ingo Pill, Marina Zanella, and Gregory Provan



OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- Martin Sachenbacher, Technical University of Applied Sciences Regensburg, Germany
- Gerald Steinbauer-Wagner, Graz University of Technology, Austria
- Roni Stern, Ben Gurion University of the Negev, Israel
- Peter Struss, Technical University of Munich, Germany
- Michal Syfert, Warsaw University of Technology, Poland
- Anna Sztyber-Betley, Warsaw University of Technology, Poland
- Stefano Tonetta, Fondazione Bruno Kessler, Italy
- Gianluca Torta, Università di Torino, Italy
- Louise Travé-Massuyès, LAAS-CNRS / Université de Toulouse, France
- Günther Waxenegger-Wilfing, DLR and Julius Maximilian University Würzburg, Germany
- Franz Wotawa, Graz University of Technology, Austria
- Marina Zanella (chair), Università di Brescia, Italy

■ List of Authors

Nir Aharoni  (5)

Ben-Gurion University of the Negev,
Beer Sheva, Israel

Yarin Benyamin  (5)

Ben-Gurion University of the Negev,
Beer Sheva, Israel

Léa Billet  (4)

Schaeffler, Toulouse, France; LAAS-CNRS,
University of Toulouse, INSA, France

Gautam Biswas  (8)

Institute for Software Integrated Systems,
Vanderbilt University, Nashville, TN, USA

Elodie Chantry  (4, 6)

LAAS-CNRS, University of Toulouse, INSA,
UPS, France

Austin Coursey  (8)

Institute for Software Integrated Systems,
Vanderbilt University, Nashville, TN, USA

Ankita Das  (11)

Institute of Software Engineering and Artificial
Intelligence, Graz University of Technology,
Austria

Johan de Kleer  (9)

c-infinity, Mountain View, CA, USA

Jan Deeken  (8)

Institute of Space Propulsion, German
Aerospace Center (DLR), Hardthausen,
Germany;
Institute of Jet Propulsion and Turbomachinery,
RWTH Aachen, Germany

Alexander Diedrich  (7, 18)

Helmut Schmidt University, Hamburg, Germany

Kai Dresia  (8)

Institute of Space Propulsion, German
Aerospace Center (DLR), Erlenbach, Germany

Thomas Eckstein (10)

Institute of Software Engineering and Artificial
Intelligence, Graz University of Technology,
Austria

Raul Sena Ferreira  (2)

Airbus Protect, Toulouse, France

Alexandre Gaffet  (4)

Schaeffler, Toulouse, France

Leonie Hatte  (6)

LAAS-CNRS, Université de Toulouse, INSA,
UPS, France

René Heesch  (18)

Helmut-Schmidt-University, Hamburg, Germany

Nemanja Hranisavljevic  (7)

Helmut Schmidt University, Hamburg, Germany

Daniel Jung  (19)

Linköping University, Sweden

Roxane Koitz-Hristov  (11)

Institute of Software Engineering and Artificial
Intelligence, Graz University of Technology,
Austria

Mattias Krysander  (18)

Linköping University, Sweden

Eldin Kurudzija  (8)

Institute of Space Propulsion, German
Aerospace Center (DLR), Köln, Germany

Jean-Michel Loubes  (2)

INRIA, Toulouse, France

Frank Marek (7)

Novatio Solutions GmbH, Monheim, Germany

Lukas Moddemann  (7)

Helmut Schmidt University, Hamburg, Germany

Shiwali Mohan  (5)

Future Concepts Division, SRI International,
Menlo Park, CA, USA

Emir Mujić  (3, 16)

Institute of Software Engineering and Artificial
Intelligence, Graz University of Technology,
Austria

Iulia Nica  (17)

Institute of Software Engineering and Artificial
Intelligence, Graz University of Technology,
Austria

Oliver Niggemann  (7, 18)

Helmut Schmidt University, Hamburg, Germany

Sena Ozgunay  (2)

LAAS-CNRS, Université de Toulouse, France

Yannick Pencolé  (12)

LAAS-CNRS, CNRS, Université de Toulouse,
France

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan



OpenAccess Series in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Alexander Perko  (3)

Institute of Software Engineering and Artificial
Intelligence, Graz University of Technology,
Austria

Ingo Pill  (7, 9)

Institute of Software Engineering and Artificial
Intelligence, TU Graz, Graz, Austria

Sophia Plank (14)

Graz University of Technology, Austria

Gregory Provan  (1, 13)

School of Computer Science and IT,
University College Cork, Ireland

Marcos Quinones-Grueiro  (8)

Institute for Software Integrated Systems,
Vanderbilt University, Nashville, TN, USA

Pauline Ribot  (6, 12)

LAAS-CNRS, Université de Toulouse, INSA,
UPS, France

Antoni Rogowski  (15)

Warsaw University of Technology, Poland

Domenic Schaeffer (7)

Digital Twin Factory GmbH,
Bad Oeynhausen, Germany

Gerald Steinbauer-Wagner  (10)

Institute of Software Engineering and Artificial
Intelligence, Graz University of Technology,
Austria

Roni Stern  (5)

Ben-Gurion University of the Negev, Israel

Anna Sztyber-Betley  (15)

Warsaw University of Technology, Poland

Ferhat Tamssaouet  (12)

LAAS-CNRS, Université de Toulouse, France

Louise Travé-Massuyès  (2, 4)

LAAS-CNRS, Université de Toulouse, CNRS,
France

Günther Waxenegger-Wilfing  (8)

Institute of Space Propulsion, German
Aerospace Center (DLR), Hardthausen,
Germany;
Institute of Computer Science, University of
Würzburg, Germany

Theodor Westny  (19)

Linköping University, Sweden

Franz Wotawa  (3, 11, 14, 16, 17)

Institute of Software Engineering and Artificial
Intelligence, Graz University of Technology,
Austria

A Realization-Theoretic Foundation for Fault Diagnosis, with Diagnosability Equivalence Predictions

Gregory Provan 

School of Computer Science and IT, University College Cork, Ireland

Abstract

Fault diagnosis encompasses a wide range of paradigms, including observer-based fault detection and isolation, consistency-based diagnosis, Bayesian diagnosis, geometric diagnosis, and structured machine-learning approaches. Despite their shared objective, these paradigms are formulated in different mathematical languages and are rarely analyzed within a common framework. This paper introduces a realization-theoretic foundation for diagnosis based on the Diagnostic System Specification (DSS), a representation that separates behavioural models, realizations, measurement fields, and inference procedures.

We define diagnostic invariants as symmetry-preserving quantities of realized system structures and show that detectability is fundamentally an orbit separation property in the realization's symmetry space. Building on this observation, we develop a theory of realizations in which diagnostic models appear as objects equipped with invariant families and nominal symmetry groups. We show that diagnostic paradigms differ not in their underlying architecture but in the realizations they induce and the invariants they expose.

Our main result addresses the question: is there a single mathematical condition of which diagnosis approaches are equivalent in terms of diagnosability? We show that using orbit separation on a paradigm-appropriate realization shows the following: faults are distinguishable iff they lie in different orbits of the realization's nominal symmetry group. We accomplish this by constructing a minimal sufficient realization associated with a measurement field and characterize it by a universal property. This yields a structural notion of diagnostic equivalence and provides a principled basis for comparing methods across traditionally separate paradigms.

2012 ACM Subject Classification Hardware → Fault tolerance; Theory of computation → Automata extensions

Keywords and phrases Model-based Diagnosis, Diagnosability, System Realization Theory

Digital Object Identifier 10.4230/OASICS.DX.2026.1

Funding Funded by Research Ireland grant 13/RC/2094_P2.

1 Introduction

Fault diagnosis is a foundational capability of engineered systems. Modern diagnostic methods span a wide spectrum of paradigms, including observer-based fault detection and isolation (FDI), consistency-based diagnosis, Bayesian diagnosis, geometric diagnosis, discrete-event diagnosis, and machine-learning-based approaches [8, 9, 12, 13]. Although these paradigms pursue the same goal – detecting and isolating abnormal behaviour – they are formulated in markedly different mathematical languages. Observer methods reason over residuals, consistency-based methods over logical theories and conflicts, Bayesian methods over probability distributions, discrete-event methods over event languages, and machine-learning methods over learned representations. As a result, diagnostic theories are largely developed in isolation, making principled comparison across paradigms difficult.

This fragmentation raises a basic question: what structure is common to diagnostic methods independently of the formalism in which they are expressed? We argue that the central object is neither the inference algorithm nor the modelling language, but the



© Gregory Provan;

licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 1; pp. 1:1–1:20

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

realization through which a method views a system. Every diagnostic method begins with a behavioural specification, constructs a structure on which reasoning is performed, evaluates quantities defined on that structure, and maps those quantities to diagnostic conclusions. The mathematical form of these steps varies, but the pattern itself is remarkably consistent across paradigms.

To make this observation precise we introduce the Diagnostic System Specification (DSS), a representation that separates behavioural models, realizations, measurement fields, and inference procedures. The DSS provides a common language in which observer-based, logical, probabilistic, geometric, discrete-event, and learning-based diagnostic methods can be expressed without privileging any one modelling framework.

Building on the DSS, we develop a realization-theoretic framework for diagnosis. The key observation is that diagnostic measurements typically evaluate quantities that are invariant under transformations preserving nominal behaviour. Residual norms, conflict signatures, posterior distributions over exchangeable components, and geometric quantities all exhibit this property. This leads to a symmetry-based view of diagnosis in which distinguishability is determined by the orbit structure induced by a realization's nominal symmetry group.

Our main result establishes a common criterion underlying diagnosability across paradigms. We show that, relative to an appropriate realization, faults are distinguishable precisely when they occupy different invariant-separated orbits of the nominal symmetry group. We formalize this notion through the construction of a minimal sufficient realization, characterized by a universal property, and show that diagnostic capability depends only on this object. This yields a structural notion of diagnosable-equivalence: two diagnostic methods are equivalent whenever they induce the same minimal sufficient realization relative to the fault set of interest.

The framework has two consequences. First, it recovers established diagnosability conditions as special cases, including observer-based rank tests, Reiter-style conflict diagnosability, and Sampath diagnosability for discrete-event systems. Second, it allows diagnostic methods from different paradigms to be compared through a common structural criterion. In particular, it becomes possible to predict whether two methods possess identical diagnosability without executing either method.

2 The Diagnostic System Specification

We first define the DSS in a single canonical form.

► **Definition 1** (Diagnostic System Specification). *A Diagnostic System Specification is a tuple $\mathcal{D} = (X, B, F, O, P, h, R, M, I, \Delta)$, whose components are:*

X the state space – continuous, hybrid, symbolic, or propositional;

B the behavioural specification – the model of nominal (and possibly faulty) evolution, e.g. $\dot{x} = f(x, u)$, a hybrid automaton $(Q, f, G, R_, \text{Inv})$, a system description SD , a transition kernel $P(x_{t+1} \mid x_t)$, or a learned predictor f_θ ;*

F the fault space – the set of admissible faults (parametric, component, structural, topological, sensor, actuator);

O the observation space – sensor readings, residuals, propositions, events, or features;

P the probabilistic structure $P = (p(x), p(o \mid x), p(f), p(x_{t+1} \mid x_t, f))$; deterministic methods are the degenerate case in which these collapse to Dirac measures;

$h : X \rightarrow O$ the observation map;

$R : B \mapsto S$ the realization map, producing the structure $S = R(B)$ on which invariants are computed (Definition 3);

$M : S \times O \rightarrow \mathbb{R}^r$ the measurement field, whose components evaluate diagnostic invariants of S using observations;

$I : \mathbb{R}^r \rightarrow \Delta$ the inference procedure;

Δ the diagnosis space – the set of admissible diagnostic verdicts (single faults, fault sets, or posteriors over them).

The reset map of a hybrid behavioural model is written R_* to distinguish it from the realization map R .

► **Remark 2** (Reading the tuple as a pipeline). The components compose as

$$B \xrightarrow{R} S, \quad X \xrightarrow{h} O, \quad (S, O) \xrightarrow{M} \mathbb{R}^r \xrightarrow{I} \Delta.$$

The realization $S = R(B)$ and the observations $O = h(X)$ are the two inputs to the measurement field; inference maps the field output to a diagnosis. This pipeline, not any single paradigm, is the object the paper characterises. The probabilistic structure P is orthogonal to the pipeline shape: it enriches M and I (likelihoods, posteriors) without changing the composition, and its Dirac collapse recovers the deterministic pipeline.

► **Definition 3** (Realization map). The realization map R sends a behavioural specification to the mathematical structure on which diagnosis is performed:

$$R : \text{Beh} \longrightarrow \text{Struct}, \quad S = R(B).$$

Realizations fall into two classes. A realization is structural if S is discrete, symbolic, probabilistic, or graph-based (logical theory, finite-state machine, Bayesian network); it is geometric if S carries differentiable or stratified structure (smooth manifold, Whitney-stratified space / stratifold). The class of S – not a fixed geometric template – determines which invariants M can read; this is the sense in which the framework is realization-agnostic rather than geometry-centric.

We now define what object is *canonical* for a method. The canonical object is the coarsest realization that still supports M – the structure generated by the invariants M reads.

► **Definition 4** (Minimal sufficient realization). Let $M = (\iota_1, \dots, \iota_r)$ be a measurement field on S , with each $\iota_j : S \rightarrow V_j$. Write

$$\Sigma_M := \prod_{j=1}^r V_j$$

for the joint invariant value space of M – equivalently, the space in which M takes its values, $M : S \rightarrow \Sigma_M$. Let $N_M := \{g \in G_S : \iota_j(g \cdot s) = \iota_j(s) \ \forall j \in \{1, \dots, r\}, \ \forall s \in S\}$ be the subgroup of G_S acting trivially on Σ_M , i.e. the kernel of the action $G_S \curvearrowright \Sigma_M$ induced by M . Let $\sim_M$ be the equivalence relation on S with $s \sim_M s'$ iff $M(s) = M(s')$. The minimal sufficient realization of M is the quotient $S_M := S / \sim_M$, equipped with the induced symmetry group $G_{S_M} = G_S / N_M$, the quotient map $\pi_M : S \rightarrow S_M$, and the invariant family generated by the descended maps $\bar{\iota}_j : S_M \rightarrow V_j$ (well-defined since each ι_j is constant on $\sim_M$ -classes, and in particular on N_M -orbits). By construction M factors through S_M : $M = \bar{M} \circ \pi_M$ with $\bar{M} = (\bar{\iota}_1, \dots, \bar{\iota}_r)$.

■ **Table 1** Canonical DSS components and their instantiation across four representative paradigms. Every downstream theorem refers to this tuple; no other arity is used. Dashes mark components that collapse (deterministic P) or carry no nontrivial structure (e.g. no useful manifold for a propositional realization).

Component	Observer FDI [9]	Reiter MBD [12]	Bayesian net [5]	Geometric (GMBD [11])
B	$\dot{x} = Ax + Bu$	SD clauses	$P(x_{t+1} x_t)$	hybrid (Q, f, G, R_*)
$R(B)=S$	smooth manifold	logical theory	Bayesian network	stratifold X_H
P	Dirac (or Gaussian)	Dirac	full	Dirac (or noise model)
M	residual r_t	consistency preds.	likelihood/marginals	Geometric-signatures [11]
I	threshold	hitting set	belief propagation	geometric predicate
Δ	fault flag	minimal diagnoses	posterior over faults	topological verdict

3 Diagnostic Invariants

The framework repeatedly refers to *invariants* – surviving, consistency, geometric, diagnostic. Because the selection guarantees rest on this notion, we give it a single rigorous definition and show that the four usages are instances of it. The definition is group-theoretic: an invariant is a functional on the realized structure that is constant on the orbits of the realization’s nominal symmetry group and that separates distinct orbits. This makes “a fault changes an invariant” mean “the fault moves the state to a different orbit,” which is the precise, non-circular content behind detectability and behind the admissibility gates.

3.1 Nominal symmetry group and invariants

Let $S = R(B)$ be the realized structure (Definition of the realization map). Diagnosis must be blind to differences in S that are pure representation – relabelling of interchangeable components, choice of coordinates, observer gauge, the noise orbit at a point – and sensitive only to differences that correspond to genuine changes in system behaviour. We encode the former as a group action.

► **Definition 5** (Nominal symmetry group). *The nominal symmetry group G_S of a realization S is the group of transformations $g : S \rightarrow S$ that preserve nominal (fault-free) behaviour: permutations of interchangeable components (structural realizations), local isometries / coordinate changes preserving the dynamics (geometric realizations), observer-gauge transformations (smooth realizations), and measure-preserving maps of the observation-noise orbit. Two structural configurations $s, s' \in S$ are behaviourally equivalent, $s \sim_{G_S} s'$, iff $s' = g \cdot s$ for some $g \in G_S$; the equivalence classes are the orbits S/G_S .*

► **Definition 6** (Diagnostic invariant). *A diagnostic invariant of S is a measurable functional $\iota : S \rightarrow V$ (into a value space V) that is*

1. **G_S -invariant:** $\iota(g \cdot s) = \iota(s)$ for all $g \in G_S, s \in S$ – so ι is constant on orbits and never reports a purely representational difference; and
2. **orbit-separating on its domain of sensitivity:** there is a set of orbit-pairs $\text{Sep}(\iota) \subseteq (S/G_S)^2$ on which ι takes distinct values, $[s] \neq [s'] \in \text{Sep}(\iota) \Rightarrow \iota(s) \neq \iota(s')$ – so a change in ι certifies a genuine (cross-orbit) structural difference.

$\text{Sep}(\iota)$ is the set of structural differences ι can resolve; distinct invariants resolve different pairs.

► **Definition 7** (Measurement field as an invariant vector). *A measurement field is a finite vector of diagnostic invariants $M = (\iota_1, \dots, \iota_r)$, and its resolving power is $\text{Sep}(M) = \bigcup_j \text{Sep}(\iota_j)$: the set of orbit-pairs separated by at least one component. This is the rigorous content of the informal “ M reads invariants of S ”.*

► **Remark 8** (Measurement field/detector distinction). A measurement field is **not** a classical FDI detector, due to the DSS’s separation between measurement and inference: M assigns invariant values across S and is agnostic to any decision rule, while a detector is the composite $I \circ M$ that consumes those values to produce a verdict in Δ . Collapsing the two would erase exactly the M/I distinction the DSS is built to expose.

► **Remark 9** (The four usages are one definition). Each named invariant is Definition 6 for a specific (S, G_S) : *consistency invariant* – S a logical theory, G_S the component-relabelling group, $\iota \in \{\text{consistent}, \text{violated}\}$; *geometric invariant* – S a stratifold, G_S the group of local $O(D)$ isometries, $\iota \in \{d_{\text{eff}}, \kappa_{\text{OR}}\}$ (invariance under $O(D)$ is precisely why these are well-defined intrinsic quantities) [11]; *residual invariant* – S a smooth realization, G_S the observer-gauge group, $\iota = \|r\|$; *diagnostic invariant* – the generic term. The “surviving invariant” of the selection narrative is any ι with the target fault’s orbit-pair in $\text{Sep}(\iota)$.

3.2 Faults as orbit changes; detectability as separation

A fault is diagnostically meaningful exactly when it changes the orbit – otherwise it is a representational artifact, not a behavioural change.

► **Definition 10** (Orbit-changing fault). *A fault f is orbit-changing for S if it maps the nominal orbit $[s^0]$ to a distinct orbit $[s^f] \neq [s^0]$ in S/G_S . The pair $([s^0], [s^f])$ is the fault signature in S/G_S .*

► **Lemma 11** (Quantitative Detectability). *Fix a measurement field $M = (\iota_1, \dots, \iota_r)$ on S and a fault f with nominal and faulty orbit representatives s^0, s^f . Assume a measurement noise model in which repeated evaluations of M take the form $M(s, \cdot) = \mu(s) + \eta$, where $\mu(s) = \mathbb{E}[M(s, \cdot)] \in \mathbb{R}^r$ and the noise vector η has independent, zero-mean, sub-Gaussian coordinates with common proxy scale s_M (i.e. $\mathbb{E}[e^{t\eta_i}] \leq e^{t^2 s_M^2 / 2}$ for all $t \in \mathbb{R}$, each coordinate i), independent of the orbit. Let $\Delta_M(f) := \|\mu(s^f) - \mu(s^0)\|$. Fix a false-alarm level $\varepsilon \in (0, 1)$ and a miss level $\delta \in (0, 1)$, and set $\kappa(\varepsilon, \delta) := \sqrt{2 \ln(1/\varepsilon)} + \sqrt{2 \ln(1/\delta)}$.*

1. **(Necessity.)** *If $\Delta_M(f) = 0$, then $M(s^f, \cdot)$ and $M(s^0, \cdot)$ are identically distributed, and consequently no test (measurable decision rule) can achieve false-alarm rate and miss rate both strictly less than 1 simultaneously; in particular f is not (ε, δ) -detectable by M at any threshold, for any $\varepsilon + \delta < 1$.*
2. **(Sufficiency.)** *If $\Delta_M(f) > \kappa(\varepsilon, \delta) \cdot s_M$, there is a threshold test on M that achieves false-alarm rate $\leq \varepsilon$ and miss rate $\leq \delta$; i.e. f is (ε, δ) -detectable by M .*

Proof. (1) If $\Delta_M(f) = 0$ then $\mu(s^f) = \mu(s^0)$, so $M(s^f, \cdot) = \mu(s^0) + \eta = M(s^0, \cdot)$ in distribution: the two hypotheses induce identical laws on the observation. For any measurable rejection region A (declare-fault region), the false-alarm rate is $P(A \mid H_0) = P(M(s^0, \cdot) \in A)$ and the miss rate is $P(A^c \mid H_1) = P(M(s^f, \cdot) \notin A) = P(M(s^0, \cdot) \notin A)$ since the two distributions coincide. Hence

$$P(A \mid H_0) + P(A^c \mid H_1) = P(M(s^0, \cdot) \in A) + P(M(s^0, \cdot) \notin A) = 1$$

identically, for every choice of A : false-alarm and miss rates cannot both be driven below any pair (ε, δ) with $\varepsilon + \delta < 1$. This is the precise content of “undetectable by M at every threshold” used in Theorem 12(1).

(2) Let $u = (\mu(s^f) - \mu(s^0))/\Delta_M(f)$ be the unit vector along the mean shift, and consider the scalar statistic $Y := u^\top(M - \mu(s^0))$. Under H_0 , $Y = u^\top \eta$; under H_1 , $Y = \Delta_M(f) + u^\top \eta$. Since the coordinates of η are independent sub-Gaussian with common proxy s_M and u is a unit vector, $u^\top \eta$ is sub-Gaussian with proxy s_M (a weighted sum of independent sub-Gaussians with weights summing in squares to $\|u\|^2 = 1$), giving the two-sided tail bound

$$P(|u^\top \eta| > t) \leq 2 \exp\left(-\frac{t^2}{2s_M^2}\right) \quad \text{for all } t \geq 0.$$

(We use the one-sided form $P(u^\top \eta > t) \leq \exp(-t^2/2s_M^2)$ below.)

Take the threshold test: declare fault iff $Y > \tau$, with $\tau := s_M \sqrt{2 \ln(1/\varepsilon)}$.

False-alarm rate. Under H_0 ,

$$P(Y > \tau \mid H_0) = P(u^\top \eta > \tau) \leq \exp\left(-\frac{\tau^2}{2s_M^2}\right) = \exp(-\ln(1/\varepsilon)) = \varepsilon.$$

Miss rate. Under H_1 ,

$$P(Y \leq \tau \mid H_1) = P(u^\top \eta \leq \tau - \Delta_M(f)) = P(u^\top \eta \leq -(\Delta_M(f) - \tau)).$$

By hypothesis $\Delta_M(f) > \kappa(\varepsilon, \delta) s_M = \tau + s_M \sqrt{2 \ln(1/\delta)}$, so $\Delta_M(f) - \tau > s_M \sqrt{2 \ln(1/\delta)}$, and by the sub-Gaussian tail bound applied to $-u^\top \eta$ (also sub-Gaussian with proxy s_M),

$$P(u^\top \eta \leq -(\Delta_M(f) - \tau)) \leq \exp\left(-\frac{(\Delta_M(f) - \tau)^2}{2s_M^2}\right) < \exp(-\ln(1/\delta)) = \delta.$$

Hence this threshold test achieves false-alarm $\leq \varepsilon$ and miss $< \delta$, establishing (ε, δ) -detectability. $\blacktriangleleft$

► **Theorem 12** (Detectability equals orbit separation). *Let $M = (\iota_1, \dots, \iota_r)$ be a field on S and f an orbit-changing fault with signature $([s^0], [s^f])$. Then:*

1. (Blindness) *If $([s^0], [s^f]) \notin \text{Sep}(M)$ then $\Delta_M(f) = 0$: every component satisfies $\iota_j(s^f) = \iota_j(s^0)$, so the field has identical nominal and faulty distributions and, by Lemma 11(1), f is undetectable by M at every threshold.*
2. (Sensitivity) *If $([s^0], [s^f]) \in \text{Sep}(M)$ then some ι_j separates the orbits, $\Delta_M(f) \geq |\iota_j(s^f) - \iota_j(s^0)| > 0$, and f is (ε, δ) -detectable once this gap clears the noise floor $\kappa(\varepsilon, \delta) s_M$ (Lemma 11(2)).*

Hence f is detectable by M if and only if f 's signature lies in $\text{Sep}(M)$ and the induced gap clears the noise floor: detectability is orbit separation above noise.

Proof. (1) By G_S -invariance each ι_j is constant on orbits, so if $[s^f]$ and $[s^0]$ are not separated by any ι_j (signature outside $\text{Sep}(M)$), then $\iota_j(s^f) = \iota_j(s^0)$ for all j , whence the population field values coincide and $\Delta_M(f) = \|\mathbb{E}[M(s^f)] - \mathbb{E}[M(s^0)]\| = 0$. Equal means with common noise give identically distributed statistics under H_0 and H_1 ; the detectability theorem's necessity clause then forbids detection above the false-alarm rate. (2) If the signature is in $\text{Sep}(M)$, by Definition 6(2) some component takes distinct values on the two orbits, lower-bounding $\Delta_M(f)$ by that gap; sufficiency of the detectability theorem gives (ε, δ) -detectability once $\Delta_M(f) > \kappa s_M$. $\blacktriangleleft$

► **Remark 13** (Why this de-circularizes the gates). Theorem 12 grounds invariant-detectability in a prior object – the orbit space S/G_S , defined without reference to any method. “Fault-blind” now means precisely “signature outside $\text{Sep}(M)$,” a statement about the field and the group alone.

► **Remark 14** (Practical estimation of the separation). We can set γ and s_M per method without circularity. Theorem 12 localizes the question: what must be estimated is the orbit gap $|\iota_j(s^f) - \iota_j(s^0)|$ relative to the field's nominal dispersion s_M . Both are estimated on a *held-out nominal calibration set* (for s_M and the nominal orbit value) and a *fault library or fault-injection simulation* (for the faulty orbit value), disjoint from the run-time observations used for the actual diagnosis – exactly the split used in the two-tank study, where nominal and injected-fault trajectories estimate the loci and their spread. The margin $\gamma = \Delta_M(f)/s_M$ is then a measured quantity on calibration data, not a free parameter.

4 Recovering Classical Detectability and Isolability

The orbit-separation criterion (Theorem 12) captures detectability and isolability in general. We now show that our framework reproduces each paradigm's established diagnosability condition as a special case. In particular, we show that orbit separation recovers two canonical conditions – the LTI fault-signature rank test and Reiter conflict-based diagnosability – exactly, and verify each numerically on the two-tank system. (The discrete-event case, Sampath diagnosability, is treated separately, as it requires lifting orbit separation from states to the diagnoser's language level.)

4.1 LTI observer FDI: orbit separation is the rank test

Consider the linear realization $\dot{x} = Ax + Bu + \sum_i E_i f_i$, $y = Cx$, with fault signature directions E_i and observer residual $r = y - C\hat{x}$. The classical conditions (Ding; Gertler) are: fault f_i is *detectable* iff its residual signature $g_i := C(-A)^{-1}E_i$ is nonzero, and faults f_i, f_j are *isolable* iff their signatures are linearly independent (the fault signature matrix has full column rank).

► **Definition 15** (Observer-gauge group). *The nominal symmetry group G_S of the LTI realization is the observer-gauge group: state-coordinate changes $x \mapsto Tx$ and output-injection reparametrisations that preserve the nominal residual law (residual identically zero under no fault). Its action on fault signatures is by the induced change of basis; two signatures are in the same G_S -orbit iff they are related by a gauge transformation, i.e. span the same residual direction.*

► **Theorem 16** (Orbit separation recovers the LTI rank test). *Under the observer-gauge group (Definition 15) with the residual-direction invariant $\iota(f_i) = [g_i]$ (the signature up to gauge):*

1. f_i is orbit-separated from the nominal iff $g_i \neq 0$ – equivalently, f_i is detectable by the standard test;
2. f_i, f_j are orbit-separated from each other iff g_i, g_j are linearly independent – equivalently, $\{f_i, f_j\}$ is isolable by the standard rank test.

Hence $\text{Sep}(M)$ for the observer field equals the set of fault pairs the rank test declares isolable, and the number of distinct orbits equals the rank of the fault signature matrix.

Proof. The residual is a G_S -invariant (gauge transforms preserve the nominal residual law), and its direction $[g_i]$ is orbit-separating by construction. (1) f_i shares the nominal orbit iff its signature is gauge-equivalent to zero, i.e. $g_i = 0$; so orbit-separation from nominal $\Leftrightarrow g_i \neq 0 \Leftrightarrow$ detectability. (2) f_i, f_j share an orbit iff g_i, g_j are gauge-related, i.e. span the same direction, i.e. linearly *dependent*; so orbit-separation $\Leftrightarrow$ independence $\Leftrightarrow$ the rank-2 isolability condition. The orbit count is the number of independent signature directions, which is the rank of $[g_1 \cdots g_k]$. ◀

► **Example 17** (Two-tank, numerically verified). Linearising the two-tank about $(h_1, h_2) = (1.2, 0.6)$ gives $A = \begin{bmatrix} -0.387 & 0.387 \\ 0.387 & -0.645 \end{bmatrix}$, $C = I$. The pump fault ($E = [1, 0]^\top$) and valve fault ($E = [-1, 1]^\top$) yield residual signatures $g_{\text{pump}} = [6.46, 3.87]^\top$ and $g_{\text{valve}} = [-2.58, 0]^\top$; a state-space sensor fault yields $g = 0$. The framework's verdicts match the standard test exactly: pump and valve detectable ($g \neq 0$), the zero-signature fault undetectable, and pump/valve orbit-separated (signatures independent, rank 2) hence isolable. The distinct-orbit count (2) equals the signature-matrix rank (2). All checks pass in the numerical verification.

4.2 Reiter MBD: orbit separation is conflict diagnosability

Consider the logical realization $S = R(SD)$ over components with health in $\{\text{ok}, \text{ab}\}$. The classical condition [12, 6]: two fault modes are *diagnosable* under an observation set O iff they produce different minimal diagnoses (minimal hitting sets of the conflicts) given O .

► **Theorem 18** (Orbit separation recovers conflict diagnosability). *Let G_S be the clause-preserving component-relabel group and let the invariant be the conflict signature $\iota_O(f) = (\text{minimal conflicts}, \text{minimal diagnoses})$ induced by fault f under observation set O . Then, relative to O , two fault modes are orbit-separated iff they yield distinct conflict signatures iff they are conflict-diagnosable under O . Restricting O (hiding an observable) acts by the evidence-restriction to the stabilizer subgroup; faults whose only distinguishing behaviour is hidden become orbit-merged – exactly the loss of diagnosability under reduced observation.*

Proof. The conflict signature is a function of SD and O , invariant under clause-preserving relabellings (they permute syntactically identical components without changing the conflict structure), so it is a G_S -invariant. Two fault modes share an orbit under O iff their conflict signatures coincide, i.e. they induce the same minimal diagnoses – the negation of conflict-diagnosability. Hence orbit separation $\Leftrightarrow$ distinct diagnoses $\Leftrightarrow$ diagnosability under O . Reducing O removes the behaviours distinguishing certain faults; by Proposition 26 the orbit space coarsens under fewer observations (here fewer separating predicates), merging faults whose distinguishing observable is removed. ◀

► **Example 19** (Two-tank, numerically verified). The two-tank Reiter model has components $\{P, V, S\}$ (pump, valve, sensor) with clauses $P=\text{ok} \Rightarrow Q_{\text{in}} > 0$, $V=\text{ok} \Rightarrow \text{transfer}$, $S=\text{ok} \Rightarrow \text{alarm-consistent}$. Under full observations each single fault produces a distinct conflict $\{P\}$, $\{V\}$, or $\{S\}$ and a distinct minimal diagnosis; all three pairs are orbit-separated and conflict-diagnosable (verified: all pairs MATCH). Under *restricted* observations that hide the alarm behaviour, the sensor fault S produces no conflict – its signature equals the nominal – so S becomes orbit-merged with nominal and undiagnosable, precisely the evidence-restriction prediction (verified: S orbit-merges with nominal under restricted O). The observation-relative statement of Theorem 18 is therefore not a technicality but the mechanism by which reduced sensing loses diagnosability.

4.3 What the recovery establishes

► **Remark 20** (Orbit separation unifies two incommensurable conditions). Theorems 16 and 18 show that a continuous-analytic condition (the fault-signature rank test) and a discrete-logical condition (conflict diagnosability) are the *same* orbit-separation criterion instantiated on two different realizations, with two different symmetry groups (observer-gauge; component-relabel). Neither was designed into the framework; both are recovered. This is the credibility the abstraction needed: it reproduces the field's established detectability and isolability tests

as special cases, and the two-tank instances confirm the equivalences numerically rather than by assertion. The discrete-event (Sampath) condition, which lives at the level of event languages rather than states, is studied in Section 4.4.

4.4 Discrete-Event Diagnosability: Orbit Separation at the Language Level

The third canonical condition is Sampath diagnosability for discrete-event systems [13]: a fault is diagnosable iff its occurrence is certain to be detected within a bounded number of events, equivalently iff the diagnoser automaton contains no *indeterminate cycle* – a cycle of fault-ambiguous states corresponding to two arbitrarily long observation-equivalent traces, one faulty and one normal. Unlike the LTI and Reiter conditions, this is a property of event *languages*, not of static states, and the naive state-level orbit mapping does not capture it. We show the correct correspondence, at the diagnoser’s language level, and are explicit about where the mapping is delicate.

Why state-level orbits are insufficient

Orbit separation as used for LTI and Reiter is a per-configuration notion: two configurations lie in different orbits. Sampath diagnosability is temporal – it concerns whether faulty and normal traces remain observation-indistinguishable *arbitrarily far into the future*. A cycle among fault-ambiguous diagnoser states is *not* by itself an indeterminate cycle: the cycle must be realizable by both a genuine faulty trace and a genuine normal trace with identical observations. Treating any ambiguous-state cycle as indeterminate over-flags (it declares diagnosable systems non-diagnosable). The correct object is therefore orbit separation on the diagnoser’s *language*, not its states.

► **Definition 21** (Diagnoser realization and its symmetry). *For a DES with observable event set Σ_o and unobservable fault event, the diagnoser realization S_{diag} is Sampath’s diagnoser automaton: states are fault-labelled observation-equivalence classes (subsets of system states with labels in $\{N, F\}$), transitions are on observable events with unobservable-reach closure. Its nominal symmetry group $G_{S_{\text{diag}}}$ is the group of relabellings of unobservable events and symmetric states that preserve the observable projection – the automorphisms of the observation structure.*

► **Theorem 22** (Orbit separation at the language level recovers Sampath diagnosability). *Let the invariant be the diagnoser fault label $\iota : S_{\text{diag}} \rightarrow \{N, F, A\}$ ($A = \text{ambiguous}$), and lift orbit separation to traces: a faulty trace is orbit-separated from the normal traces iff it reaches an F -labelled diagnoser state, i.e. is not confined to A -labelled states forever. Then the fault is orbit-separated (every faulty trace eventually separates) iff the diagnoser has no indeterminate cycle iff the fault is Sampath-diagnosable, provided the indeterminate-cycle test uses trace realizability: a cycle counts as indeterminate only if its event sequence is realizable by both a faulty and a normal underlying trace.*

Argument and scope. The diagnoser label is $G_{S_{\text{diag}}}$ -invariant (observation-mask relabellings preserve which system states are observation-equivalent, hence the label), and it separates the F -orbit from the N -orbit by construction. A faulty trace fails to separate iff it stays in A -labelled states for all time, i.e. lies on a cycle of ambiguous states whose event sequence is realized by both a faulty and a normal trace (else one branch is forced out, separating the label). That is exactly Sampath’s indeterminate cycle. Hence non-separation $\Leftrightarrow$ indeterminate

cycle $\Leftrightarrow$ non-diagnosability; contrapositively, orbit separation $\Leftrightarrow$ diagnosability. The scope caveat is essential: the equivalence holds for the *trace-realizable* indeterminate-cycle test, not the coarser ambiguous-state-cycle test, which over-flags delayed-but-diagnosable faults. $\blacktriangleleft$

► **Example 23** (Two-tank DES, numerically verified). We model the valve-stuck-closed fault as an unobservable event in a two-tank DES with observable events $\{\text{fill, drain, overflow}\}$ and a fault-only observable (tank-1 distinctive overflow). Two designs are tested. In the *diagnosable* design the faulty branch is forced to emit the fault-only observable, so no trace-realizable indeterminate cycle exists: the diagnoser reaches an F -label on every faulty trace. In the *non-diagnosable* design the faulty branch mimics the nominal observable behaviour, producing a genuine indeterminate cycle. The framework's orbit-separation verdict matches Sampath's diagnosability in both: separated $\Leftrightarrow$ diagnosable, merged $\Leftrightarrow$ non-diagnosable (both cases verified numerically). A third, *delayed-diagnosable* design (the fault-only observable reachable after a bounded delay) is the boundary case: it is Sampath-diagnosable, and the coarse ambiguous-state-cycle test wrongly flags it, whereas the trace-realizable test of Theorem 22 is required to classify it correctly – confirming that the language-level lift, not the state-level one, is the right correspondence.

► **Remark 24** (Three conditions, one criterion – with an honest boundary). With the LTI rank test (Theorem 16) and Reiter conflict diagnosability (Theorem 18), the DES result completes a recovery of three historically incommensurable diagnosability conditions – continuous-analytic, discrete-logical, and discrete-event – as instances of orbit separation. The DES case is the most delicate and the most informative: it shows the criterion must be applied at the level of structure appropriate to the paradigm (states for LTI/Reiter, traces for DES), and that applying it at the wrong level (state-cycles for DES) over-flags. Far from weakening the unification, this locates it precisely: orbit separation is the common criterion, and each paradigm supplies the realization – and the level – on which it is evaluated. We state the DES correspondence with its trace-realizability scope explicit rather than claim a state-level equivalence that does not hold.

5 From Detectability to Diagnosability

Theorem 12 characterizes *detectability* of a single fault: f is visible to M iff its signature lies in $\text{Sep}(M)$. The classical conditions we recover (Section 4) are, however, *diagnosability* conditions – about distinguishing faults from one another, not merely from nominal. This section makes the lift explicit and, in doing so, answers three foundational objections: it pins down the symmetry group canonically (removing analyst choice), it gives non-diagnosability a checkable, witness-producing characterization defined independently of diagnosability (breaking the apparent circularity), and it shows diagnosability is an invariant of the minimal sufficient realization, which lets the framework *predict* diagnostic equivalence of two methods without running either.

5.1 The symmetry group is canonical, not chosen

We now show that G_S is determined by the model, as the *maximal* group preserving nominal behaviour.

► **Definition 25** (Nominal symmetry group, maximal form). *Let $S = R(B)$ be a realization with nominal behaviour predicate $\mathcal{N}_S$ (the fault-free input-output relation, or the nominal measure, as appropriate). The nominal symmetry group is $G_S = \{g \in \text{Aut}(S) : g \text{ preserves } \mathcal{N}_S\}$, the group of all structure automorphisms fixing nominal behaviour. It is the maximal such group by construction.*

► **Proposition 26** (Uniqueness and canonicity of G_S). *G_S in Definition 25 is unique and canonical: it is determined by $(S, \mathcal{N}_S)$ alone, independent of the analyst. Any group G' used for diagnosis with G' -invariant fields satisfies $G' \leq G_S$, and using a proper subgroup $G' < G_S$ can only refine orbits (split them), never merge; using a group $G'' \not\leq G_S$ is unsound (it would identify behaviourally distinct configurations).*

Proof. The set of automorphisms preserving $\mathcal{N}_S$ is closed under composition and inverse (if g, h preserve $\mathcal{N}_S$ so do gh and g^{-1}) and contains the identity, hence is a group; it is the unique maximal such group because it is defined as the full set of behaviour-preserving automorphisms. Any G' whose invariants are used for a *sound* diagnosis must preserve nominal behaviour (else a nominal configuration maps to a non-nominal one, corrupting the invariant), so $G' \leq G_S$. A subgroup induces a finer orbit partition (fewer identifications), refining Sep; a group not contained in G_S contains an element moving a nominal configuration off $\mathcal{N}_S$, which merges behaviourally distinct states and is unsound. ◀

► **Remark 27** (What this settles). The orbit space S/G_S is therefore a function of the model, not a modelling choice: the analyst does not select G_S , they compute it as the automorphism group preserving nominal behaviour (for the paradigms of Section 4 this is a concrete, solvable object – gauge group, graph-automorphism group, observation-mask group). Proposition 26 also clarifies the safe direction of error: an under-estimated group over-splits (sound, costlier), an over-estimated group is unsound – so when uncertain one takes the largest *provably behaviour-preserving* group, which is exactly G_S .

5.2 Diagnosability is pairwise orbit separation

► **Corollary 28** (Diagnosability = pairwise orbit separation). *Let F be a fault set and $F^+ = F \cup \{\text{nominal}\}$. The fault set F is diagnosable by M iff for every pair $f_i \neq f_j \in F^+$, the signature pair $([s^{f_i}], [s^{f_j}]) \in \text{Sep}(M)$ – i.e. every pair of modes lands in distinct, M -separated orbits. Detectability (Theorem 12) is the special case $f_j = \text{nominal}$.*

Proof. Diagnosability requires every pair of modes to be distinguishable by the observed field. By Theorem 12 applied to the pair (f_i, f_j) (treating f_j as the reference in place of nominal), f_i and f_j are distinguishable iff their signature pair lies in $\text{Sep}(M)$. Quantifying over all pairs gives the claim; the $f_j = \text{nominal}$ instance is detectability. ◀

This is the statement the recovery theorems (Section 4) actually use: the rank test, conflict-diagnosability, and DES diagnosability are all *pairwise* distinguishability conditions.

5.3 When is a system NOT diagnosable? A non-circular criterion

The characterization of *non*-diagnosability is where the framework earns its keep against the circularity objection: it is a checkable structural condition, defined purely through the invariant map, that produces an explicit witness – and it never mentions diagnosability in its hypothesis.

► **Definition 29** (Invariant map and its fibres). *For a field $M = (\iota_1, \dots, \iota_r)$ let $\Phi_M : F^+ \rightarrow V$ send each mode to its invariant image $\Phi_M(f) = (\iota_1(s^f), \dots, \iota_r(s^f))$. A confusable class is a fibre $\Phi_M^{-1}(v)$ containing more than one mode; a fibre is non-trivial if it contains two distinct modes. Φ_M and its fibres are defined without any reference to diagnosability – they are computed from the invariants alone.*

► **Theorem 30** (Non-diagnosability has a fibre witness). *F is non-diagnosable by M iff Φ_M is not injective on F^+ , i.e. iff there exists a non-trivial fibre $\Phi_M^{-1}(v) = \{f_i, f_j, \dots\}$. The fibre is an explicit witness: the modes it contains are exactly those M cannot tell apart, and the shared value v identifies which invariants are blind to their difference. Equivalently, F is diagnosable iff Φ_M is injective on F^+ .*

Proof. By Corollary 28, F is diagnosable iff every pair $f_i \neq f_j$ has $([s^{f_i}], [s^{f_j}]) \in \text{Sep}(M)$, i.e. some ι_k separates them: $\iota_k(s^{f_i}) \neq \iota_k(s^{f_j})$, i.e. $\Phi_M(f_i) \neq \Phi_M(f_j)$. So diagnosability $\Leftrightarrow \Phi_M$ injective. Non-diagnosability is the negation: a coincidence $\Phi_M(f_i) = \Phi_M(f_j)$, i.e. a non-trivial fibre, whose members are by definition indistinguishable by every ι_k and hence by M . ◀

► **Remark 31** (Why this is not circular). The hypothesis of Theorem 30 – “ Φ_M has a non-trivial fibre” – is a property of the invariant map on the fault set, computable before diagnosability is even defined. It does not say “diagnosability fails because faults are not separated”; it says “faults f_i, f_j share invariant value v ,” a checkable fact about Φ_M , and *concludes* non-diagnosability. The invariants are not defined through diagnosability (they are G_S -invariant, orbit-separating functionals, Definition 6, fixed by the model via the canonical G_S of Proposition 26); diagnosability is *derived* from them. The direction of definition is thus model $\rightarrow G_S \rightarrow$ invariants $\rightarrow \Phi_M \rightarrow$ fibres $\rightarrow$ (non-)diagnosability, with no step depending on its successor. This is the precise sense in which the account is non-circular, and the fibre is the tangible object the classical conditions were implicitly detecting.

► **Remark 32** (The three classical failures are all “a fibre exists”). Non-diagnosability unifies the three paradigms’ *failure* conditions as a single statement. The rank test fails iff two fault signature columns are dependent – a fibre of Φ_M (equal residual directions). Conflict diagnosability fails iff two faults induce identical conflicts – a fibre (equal conflict signatures). Sampath diagnosability fails at an indeterminate cycle – a fibre at the *language* level (two traces with equal observation projection). Each classical non-diagnosability condition is exactly “ Φ_M has a non-trivial fibre,” with the paradigm fixing the level at which Φ_M is evaluated (states for LTI/Reiter, traces for DES; cf. the DES level-lift, Theorem 22).

5.4 Diagnosable-equivalence of realizations, and prediction

The deepest consequence is that diagnosability is an invariant of the *minimal sufficient realization* (Definition 4), which lets the framework predict when two structurally different methods are diagnostically identical – without running either.

► **Definition 33** (Diagnosable-equivalence). *Two methods with fields M, M' (on realizations S, S') are diagnosable-equivalent on F , written $M \equiv_F M'$, if they induce the same partition of F^+ into confusable classes: Φ_M^{-1} and $\Phi_{M'}^{-1}$ give the same fibres over F^+ .*

► **Theorem 34** (Diagnosable-equivalence is isomorphism of minimal sufficient realizations on F). *$M \equiv_F M'$ iff the minimal sufficient realizations $S_M, S_{M'}$ (Definition 4) are Struct-isomorphic when restricted to the fault set F^+ . Consequently diagnostic capability on F is a function of $S_M|_{F^+}$ alone: two methods with isomorphic minimal sufficient realizations over F^+ have identical diagnosability, regardless of how different their fields, inference rules, or paradigms appear.*

Proof. $M \equiv_F M'$ means Φ_M and $\Phi_{M'}$ have equal fibres on F^+ , i.e. they induce the same equivalence $\sim_M = \sim_{M'}$ restricted to F^+ . The minimal sufficient realization is the quotient by exactly this equivalence (Definition 4), so equal fibres give $S_M|_{F^+} = S_{M'}|_{F^+}$ up to relabelling,

i.e. a Struct-isomorphism restricted to F^+ (invariant-preserving by construction. Conversely a Struct-isomorphism of the restricted minimal realizations matches their invariant partitions, hence their fibres, giving $M \equiv_F M'$. Diagnosability is a function of the fibre partition (Theorem 30), so equal partitions give equal diagnosability. $\blacktriangleleft$

► **Corollary 35** (Diagnostic partial order and degeneration). *Define $M \preceq_F M'$ if $\Phi_{M'}$'s fibres refine Φ_M 's on F^+ (equivalently $S_{M'}|_{F^+}$ factors onto $S_M|_{F^+}$). Then $\preceq_F$ is a partial order on methods (modulo $\equiv_F$), $M \preceq_F M'$ means M' is at least as diagnostic as M on F , and a degeneration U_σ (forgetful functor) satisfies $U_\sigma M \preceq_F M$: each simplification moves down the diagnostic order by coarsening the confusable-class partition. The degeneration hierarchy is thus recovered from a purely diagnosability standpoint.*

Proof. Refinement of partitions is a partial order; it corresponds to factoring of the quotient realizations, so $\preceq_F$ orders methods by their minimal sufficient realizations. $M \preceq_F M'$ gives Φ_M coarser, hence fewer separated pairs, hence weaker diagnosability. A forgetful functor discards invariants, coarsening Φ , so $U_\sigma M \preceq_F M$. $\blacktriangleleft$

► **Remark 36.** Theorem 34's value is not that it predicts an otherwise-unknowable fact, but that it makes diagnosable-equivalence cheap to certify: computing $S_M|_{F^+}$ requires only evaluating the invariants on each fault mode, not running the full method (estimator, solver, or trained model) and comparing outputs. The feeder example (Sec. 6.2) substantiates this – the equivalence of the $\{3, 5, 6\}$ and full placements is established from the sensitivity-matrix signatures, not from running seven vs. three diagnosers to convergence.

6 Empirical Validation: Power Distribution Feeder

We now validate our approach on a power distribution feeder [1] and exercise *two* of its realizations: a structural (LinDistFlow) realization whose diagnosability recovers measurement-placement observability, and a discrete-event realization whose diagnosability recovers Sam-path's condition for a protection fault. Both use the orbit-separation criterion.

6.1 LinDistFlow Radial Feeder Model and Fault Diagnostics Framework

6.1.1 System Description and Mathematical Framework

The system under analysis consists of a radial electrical distribution feeder modeled via the linearized *LinDistFlow* equations, a standard simplification of the full non-linear AC power flow equations tailored for radial networks. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represent a tree-structured primary distribution graph where $\mathcal{V} = \{0, 1, \dots, N\}$ is the set of buses (nodes), and $\mathcal{E}$ is the set of distribution lines (edges). Bus $0 \in \mathcal{V}$ denotes the substation slack bus, maintained at a fixed reference voltage magnitude v_0 . Each line $e = (i, j) \in \mathcal{E}$ possesses a known series resistance r_e .

Under the active-power-dominant, flat-voltage linear approximation, the bus voltage magnitude vector $v \in \mathbb{R}^{N+1}$ is determined by active power injections $p \in \mathbb{R}^{N+1}$ via $v = v_0 \mathbf{1} - R_{\text{eff}} p$ where $\mathbf{1}$ is a vector of ones, and $R_{\text{eff}} \in \mathbb{R}^{(N+1) \times (N+1)}$ represents the effective path resistance sensitivity matrix. The elements of the sensitivity matrix $S = -R_{\text{eff}}$ govern the voltage variations $\frac{\partial v_i}{\partial p_j}$ and are defined explicitly by the shared path structure from the slack bus:

$$S_{i,j} = \frac{\partial v_i}{\partial p_j} = - \sum_{e \in \mathcal{P}_i \cap \mathcal{P}_j} r_e \quad (1)$$

where $\mathcal{P}_i \subset \mathcal{E}$ denotes the unique ordered set of lines forming the directional path from the slack bus 0 to bus i .

6.1.2 Fault Signatures and Structural Perturbations

The framework evaluates three distinct classes of operational anomalies spanning structural, parametric, and observation-side configurations:

1. **Structural Line Faults:** A localized change in topology or physical parameters (e.g., a line degradation or partial failure) manifests as an impedance increment Δz on a given line $e_k = (f, t) \in \mathcal{E}$. This alters the underlying path resistance, inducing a systematic voltage drop downstream of the faulted edge:

$$\Delta v_b^{\text{line}} = \begin{cases} -\Delta z \cdot r_{e_k} & \text{if } b \in \mathcal{D}(t) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $\mathcal{D}(t) \subseteq \mathcal{V}$ is the downstream subtree rooted at the child bus t .

2. **Parametric Injection Faults:** Deviations in nodal power consumption or unmonitored distributed generation yield a net injection shift Δp_j at bus j . The resulting full voltage signature corresponds to the j -th column of the sensitivity matrix S : $\Delta v^{\text{inj}} = S_{:,j} \Delta p_j$
3. **Observation-Side Sensor Faults:** A measurement instrument anomaly (e.g., a PMU or smart meter bias) alters only the local readout without perturbing the physical grid state, i.e., $\Delta v^{\text{sensor}} = \mathbf{e}_b \delta_{\text{bias}}$ where $\mathbf{e}_b$ is the b -th standard basis vector, and δ_{bias} is the additive measurement bias.

6.1.3 Observation Field and Confusable Fibres

$\mathcal{M} \subseteq \mathcal{V}$ denotes the observation placement (the subset of metered buses). The observation operator $\Phi_{\mathcal{M}} : \mathbb{R}^{N+1} \rightarrow \mathbb{R}^{|\mathcal{M}|}$ projects the full system state signature to the measured subspace: $\Phi_{\mathcal{M}}(\Delta v) = \mathbf{P}_{\mathcal{M}} \Delta v$ where $\mathbf{P}_{\mathcal{M}}$ is a projection matrix extracting rows indexed by $\mathcal{M}$.

Two system faults f_1, f_2 belong to the same *confusable fibre* $\mathcal{F}$ under the placement $\mathcal{M}$ if and only if their measured signatures are indistinguishable:

$$\mathcal{F}(f_1) = \{f_2 \in \mathcal{F}_{\text{all}} \mid \Phi_{\mathcal{M}}(\Delta v(f_1)) = \Phi_{\mathcal{M}}(\Delta v(f_2))\} \quad (3)$$

Complete structural diagnosability is attained if and only if all induced fibres are singletons ($\forall \mathcal{F}, |\mathcal{F}| = 1$). Placements that produce identical partitions of the fault space are certified as *diagnosable-equivalent*.

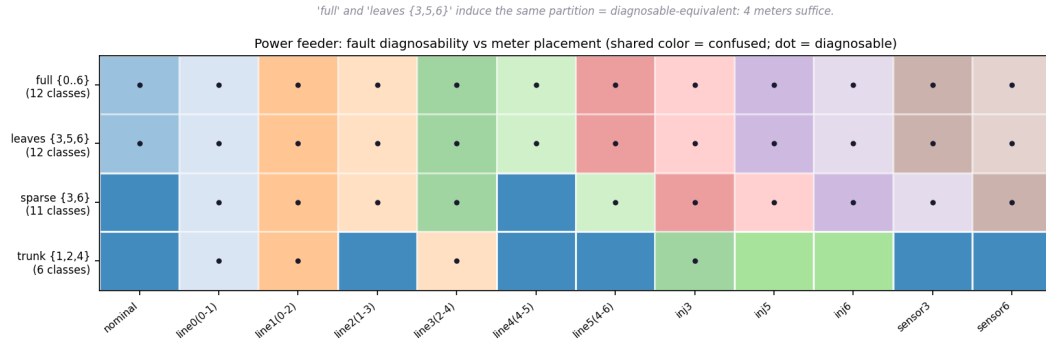
6.2 Structural realization: diagnosability vs. meter placement

We take the 7-bus radial feeder (6 lines) with the LinDistFlow linearization [1], so bus voltages depend linearly on injections through the sensitivity matrix S (algebraic, no dynamic simulation). The fault set mixes three types, giving non-trivial fibre structure: *line* faults (structural – an impedance change perturbs downstream voltages), *injection* faults (parametric – a load/generation deviation), and *sensor* faults (observation-side – a meter bias). The measurement field is the set of metered buses; its invariants are the measured voltage components of each fault's signature. Orbit separation under this field is exactly measurement-placement observability: two faults are diagnosable iff their measured voltage signatures differ.

Table 2 reports diagnosability for four meter placements, and Figure 1 shows the confusable-class partitions. The results recover the standard power-system facts and expose them as orbit separation: full metering is diagnosable; a sparse placement $\{3, 6\}$ cannot see the line-4 fault (its voltage drop is unmeasured), confusing it with nominal; a trunk

■ **Table 2** Structural diagnosability of the feeder vs. meter placement. Orbit separation under the metered-voltage field recovers measurement-placement observability; witnesses are the confusable fibres.

Meter placement	Confusable witness fibre(s)	Diag.?
full {0–6}	–	yes
leaves {3, 5, 6}	–	yes
sparse {3, 6}	{nom, line ₄ }	no
trunk {1, 2, 4}	{nom, line ₂ , line ₄ , line ₅ , sen ₃ , sen ₆ }; {inj ₅ , inj ₆ }	no



■ **Figure 1** Feeder fault diagnosability across meter placements (rows) over the fault modes (columns); shared colour = confused, dot = diagnosable. The sparse placement {3, 5, 6} induces the same partition as full metering: by Theorem 34 the two are *diagnosable-equivalent*, so four meters suffice for full diagnosability.

placement {1, 2, 4} produces two witness fibres – it misses downstream leaf and sensor faults, and it confuses the two injection faults {inj₅, inj₆} that share a trunk path. Each witness (Theorem 30) names the faults a placement cannot resolve and, by the unmetered channel, the meter whose addition would resolve them.

A non-obvious prediction: sparse metering suffices. The computation finds that metering only the three leaf buses {3, 5, 6} induces the *same* fibre partition as metering all seven buses. By Theorem 34 the two placements are diagnosable-equivalent: the framework certifies, without enumerating faults or running any estimator, that four of the seven meters can be removed with *zero* loss of diagnosability. This is the predictive content of the diagnosability theory applied to a design question (sensor placement) it was not tuned for, and it is exactly the kind of cross-configuration equivalence the per-paradigm observability tests do not express directly.

6.3 Discrete-event realization: protection diagnosability

The same feeder also has a discrete-event realization: protection relays and breakers generate observable switching events, and a protection fault – a breaker that fails to open (*stuck-closed*) – is an unobservable event whose diagnosability is a Sampath condition. We build the protection diagnoser and apply orbit separation at the language level (Theorem 22).

Two designs are tested. When backup protection observably fires on a stuck breaker (an observable `backup_trip` event), the faulty trace reaches a fault-certain diagnoser state: no indeterminate cycle, Sampath- diagnosable, and orbit separation agrees. When the backup is

silent, the stuck-breaker trace produces the same observations as a normal clear, creating an indeterminate cycle: non-diagnosable, and orbit separation agrees (both cases verified). The protection fault's diagnosability is thus the same orbit-separation criterion that governs the structural layer and the two-tank, now on the feeder's event language.

6.4 What the Power System Example Establishes

► **Remark 37** (One criterion, two realizations, a new plant). The feeder demonstrates the framework on a system structurally unlike systems like the two-tank, and does so through two different realizations of the *same* plant: a structural realization recovering measurement-placement observability, and a discrete-event realization recovering Sampath protection-diagnosability. Both are orbit separation; only the realization and its symmetry group differ. The structural layer also yields a concrete, non-obvious engineering prediction – a sparse meter placement diagnosable-equivalent to full metering – illustrating that the diagnosability theory answers design questions, not only classification ones.

7 Computational Advantages of Cross-Realization Reasoning

A framework that unifies diagnosability could be merely descriptive. We therefore address directly what *computational* advantage orbit separation provides beyond a common language. The honest answer has a sharp boundary, which we state first, and three genuine advantages, which lie on the far side of it.

7.1 What is not claimed: no single-realization complexity win

Within a single fixed realization, computing the orbit partition is not cheaper than computing the classical condition – because it is, up to interpretation, the *same* computation. For the LTI realization the fibres of Φ_M are obtained from the fault-signature directions, i.e. the rank computation itself; for the discrete-event realization the language-level fibres are the diagnoser states, i.e. the Sampath construction itself; for the Reiter realization the fibres are the conflict signatures. Orbit separation reinterprets these objects; it does not produce them faster. Any claim of a raw complexity advantage at the single-realization level would therefore be spurious, and we make none. The advantages below are all at the level where the classical theories are *silent* – across realizations, across a method library, and in guiding acquisition – because each classical test is confined to one representation and provides no cross-representation comparison.

7.2 A1 – Certified cheapest realization within an equivalence class

The diagnosable-equivalence theorem (Theorem 34) partitions candidate realizations into classes of equal diagnostic capability. Within a class one may run diagnosis on the *cheapest* member – fewest sensors, smallest state, lowest per-query cost – with a *certificate* that no diagnostic capability is lost, because the minimal sufficient realizations coincide. The advantage is not that the framework sorts realizations by cost (any scheme can do that) but that it certifies *which realizations are interchangeable*, so the cheaper one is provably sufficient.

► **Example 38** (Sparse metering, from Section 6). On the power feeder, metering the three leaf buses $\{3, 5, 6\}$ is diagnosable-equivalent to metering all seven (Theorem 34, verified in Section 6.2). Diagnosis may therefore run on a 3-dimensional measurement vector instead

of 7, with a structural certificate of identical diagnosability. The saving is concrete – fewer sensors to install and read, a smaller estimation problem – and the certificate is exactly what the per-placement observability test cannot supply: that test verifies each placement in isolation but does not establish the two are *equivalent*, which would otherwise require a fault-by-fault comparison of their diagnoses.

The general statement: because diagnosability is an invariant of the minimal sufficient realization (Theorem 34), the cheapest realization with the same $S_M|_{F+}$ is a provably-lossless substitute, and identifying it is a comparison of partitions – cheap relative to constructing or running the realizations themselves.

7.3 A2 – Partial-order pruning of a method library

When several methods or realizations are available, the diagnostic partial order (Corollary 35) lets dominated candidates be discarded *without evaluating them*. If $M \prec_F M'$ then M' diagnoses everything M does; if in addition M' is no more costly, M is dominated and can be dropped before any risk or accuracy computation. Domination is detected structurally – by partition refinement – rather than by running each method, so the expensive per-method evaluation is confined to the Pareto-undominated survivors.

This is a genuine combinatorial saving in the library setting: the cost of selection scales with the number of *surviving* (undominated) methods, not the library size, and the survivors are identified by comparing partitions. The refinement test is a containment check on fibres, whose cost is governed by the fault set size, independent of the internal complexity of the methods being ordered. A library with many dominated variants collapses to its Pareto frontier before the costly step, which is where the per-realization diagnosis or risk estimate is actually incurred.

7.4 A3 – Fibre-targeted incremental acquisition

When a fault set is non-diagnosable, the confusable fibre (Theorem 30) localizes the deficiency: it names the confused modes and, through the invariant they share, the specific distinction the current field cannot make. This makes improvement *incremental and targeted* rather than global. One need not rebuild the diagnoser or re-solve the full diagnosis problem; one computes only the *additional* invariant that splits the offending fibre.

► **Example 39** (Targeted meter addition, from Section 6). The trunk placement $\{1, 2, 4\}$ yields the witness fibre $\{\text{inj}_5, \text{inj}_6\}$: the two injection faults share the trunk-path voltage signature. The fibre identifies precisely the missing distinction – a measurement downstream of the branch separating buses 5 and 6 – so the remedy is a single targeted meter, computed against the one confusable class, not a re-instrumentation and re-solve of the whole network. Acquisition is thus guided by the fibre to the minimal resolving addition.

The general pattern: the confusable fibres form a to-do list of exactly the distinctions the current realization lacks, each pointing at the invariant whose addition resolves it. Refinement of diagnosability proceeds fibre-by-fibre, touching only the affected confusable class, rather than by global recomputation – the diagnostic analogue of splitting a coarse equivalence only where it is too coarse.

7.5 Accuracy: error avoidance, not raw improvement

We are equally precise about accuracy. The framework does not make any single method more accurate – an observer’s residual is what it is. Its contribution to accuracy is *error avoidance*: by certifying the confusable set (Theorem 30), it prevents the characteristic

failure of trusting a method on a fault it is structurally blind to. A confident diagnosis of a mode that lies in a non-trivial fibre is a confident *misdiagnosis*; the fibre flags this in advance, so the method is not over-trusted beyond its resolving power. The accuracy contribution is a guarantee about *when not to trust a verdict* – valuable precisely because classical per-paradigm confidence scores do not express structural blindness.

7.6 Summary of the advantage

► **Remark 40** (Where the advantage lives). The computational advantage of orbit separation is not a faster diagnosability test for a fixed method – there it coincides with the classical computation. It is the ability to reason *across* realizations and methods with a common currency, the fibre partition, which the per-paradigm tests cannot provide: to substitute a cheaper equivalent realization with a certificate (A1), to prune a method library to its diagnostic Pareto frontier before the costly step (A2), and to target acquisition at the exact confusable class that limits diagnosability (A3), while avoiding confident misdiagnosis by certifying structural blindness. These are advantages of *comparison and selection*, cheap relative to running the methods, and they are available only because diagnosability has been reduced to a single computable invariant – the minimal sufficient realization’s partition – shared across paradigms.

8 Related Work

Fault diagnosis has traditionally been studied through paradigm-specific theories, including observer-based FDI, consistency-based diagnosis, probabilistic diagnosis, discrete-event diagnosis, geometric diagnosis, and more recently learning-based approaches [9, 8, 12, 13]. Each paradigm introduces its own notion of distinguishability – e.g., residual rank conditions, conflict sets, indeterminate cycles, or observability codistributions – and these notions are rarely analyzed within a common framework. Our contribution is to show that they can all be interpreted as instances of a single realization-theoretic criterion: orbit separation.

Behavioural systems and realizations. The behavioural approach of Willems treats behaviour rather than state-space representation as the primary object of analysis [15]. Our realization map follows this perspective by associating a behavioural specification with a structure on which diagnosis is performed. The key difference is that behavioural theory studies equivalence of behaviours, whereas we study equivalence of diagnostic capability. Two realizations may differ behaviourally yet be diagnosable-equivalent if they induce the same partition of a fault set.

Bisimulation and quotient constructions. Bisimulation theory characterizes behavioural equivalence by quotienting states that are observationally indistinguishable [14]. Our minimal sufficient realization plays a similar role: it is the quotient induced by diagnostic invariants. Diagnosable-equivalence can therefore be viewed as a fault-relative analogue of bisimulation. The distinction is that our observables are generalized diagnostic invariants rather than transition labels, allowing continuous, logical, probabilistic, and geometric realizations to be handled within a common framework.

Discrete-event diagnosability. Sampath et al. characterize diagnosability through the diagnoser automaton and the absence of indeterminate cycles [13]. In our framework the diagnoser itself serves as the realization, and diagnosability becomes orbit separation at the language level. The resulting correspondence shows that DES diagnosability is another instance of the same structural criterion used in continuous and logical settings.

Geometric and observer-based diagnosis. Geometric observability theory characterizes state distinguishability through observability codistributions and associated indistinguishability manifolds [10]. Fault detection and isolation methods based on geometric control theory extend this perspective to fault signatures and isolability conditions [7]. In our terminology, the indistinguishability manifolds correspond to orbits and observability conditions correspond to orbit-separating invariants. The classical fault-signature rank test appears as the linear specialization of this broader picture.

Structural diagnosis and analytical redundancy. Structural FDI approaches characterize detectability and isolability through graph structure, analytical redundancy relations, and matching conditions [2, 9]. These methods already reason through a realization, namely a structural graph representation. Our framework generalizes this idea by allowing the realization to take many forms, including logical theories, Bayesian networks, diagnoser automata, geometric models, or learned representations. Structural diagnosis thus becomes one realization class among several rather than the foundational abstraction itself.

Consistency-based diagnosis. Consistency-based diagnosis and model-based reasoning identify faults through conflicts and minimal diagnoses derived from a logical system description [12]. Our recovery result shows that conflict diagnosability corresponds to orbit separation in a logical realization under component-relabelling symmetries. This places logical diagnosis within the same structural framework as observer-based and DES methods.

Symmetry and learned representations. The emphasis on symmetries and invariants connects naturally with work on equivariant systems, geometric deep learning, and invariant representation learning [4, 3]. These approaches study what information is preserved under group actions, whereas our focus is diagnostic distinguishability. The connection is that diagnostic capability is determined by the invariants exposed by a realization and the orbit structure induced by its symmetry group.

Summary. Across these literatures a common pattern emerges: each defines a notion of indistinguishability and a corresponding quotient structure. The realization-theoretic framework identifies these as manifestations of the same construction. Orbit separation provides a common criterion for fault distinguishability, while diagnosable-equivalence allows diagnostic methods from different paradigms to be compared through their minimal sufficient realizations. This enables cross-paradigm comparisons that individual diagnosis theories, considered in isolation, do not provide.

9 Summary and Conclusions

This paper introduced a realization-theoretic framework for fault diagnosis based on the Diagnostic System Specification (DSS), which separates behavioural specifications, realizations, measurement fields, and inference procedures. The framework identifies diagnosis as a property of the realization through which a system is viewed rather than of any particular diagnostic algorithm.

The central result is that diagnosability is an orbit-separation property. Relative to a realization and its nominal symmetry group, faults are distinguishable iff they occupy different invariant-separated orbits. This perspective yields a minimal sufficient realization that completely characterizes diagnostic capability and leads to a structural notion of diagnosable-equivalence among diagnostic methods.

We showed that several classical diagnosability conditions are recovered as special cases of the same criterion, including observer-based rank conditions, Reiter conflict diagnosability, and Sampath diagnosability. The framework therefore provides a common language for comparing diagnostic paradigms that are traditionally studied separately.

The empirical studies on the two-tank system and a power distribution feeder demonstrated both the recovery of established results and the ability to identify diagnostically equivalent realizations, including equivalent sensor placements.

Future work includes extending the framework to richer probabilistic and learning-based systems, developing scalable algorithms for constructing minimal sufficient realizations, and applying diagnosable-equivalence to sensor placement, observer design, and model-selection problems.

References

- 1 Mesut E Baran and Felix F Wu. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transactions on Power delivery*, 4(2):1401–1407, 1989.
- 2 Mogens Blanke, Michel Kinnaert, Jan Lunze, and Marcel Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer, 3rd edition, 2016.
- 3 Benjamin Bloem-Reddy and Yee Whye Teh. Probabilistic symmetries and invariant neural networks. *Journal of Machine Learning Research*, 21(90):1–61, 2020. URL: <https://jmlr.org/papers/v21/19-322.html>.
- 4 Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint*, 2021. [arXiv:2104.13478](https://arxiv.org/abs/2104.13478).
- 5 Baoping Cai, Lei Huang, and Min Xie. Bayesian networks in fault diagnosis. *IEEE Transactions on industrial informatics*, 13(5):2227–2240, 2017. doi:10.1109/TII.2017.2695583.
- 6 Johan De Kleer and Brian C Williams. Diagnosing multiple faults. *Artificial intelligence*, 32(1):97–130, 1987. doi:10.1016/0004-3702(87)90063-4.
- 7 Claudio De Persis and Alberto Isidori. A geometric approach to nonlinear fault detection and isolation. *IEEE Transactions on Automatic Control*, 46(6):853–865, 2001. doi:10.1109/9.928586.
- 8 Steven X Ding. *Data-driven design of fault diagnosis and fault-tolerant control systems*. Springer, 2014.
- 9 Janos Gertler. *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, 1998.
- 10 Robert Hermann and Arthur J. Krener. Nonlinear controllability and observability. *IEEE Transactions on Automatic Control*, 22(5):728–740, 1977.
- 11 Gregory Provan. Statistical foundations of local geometric fault detection: the curvature channel. In *Principles of Diagnosis and Resilient Systems*, 2026.
- 12 Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987. doi:10.1016/0004-3702(87)90062-2.
- 13 Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995. doi:10.1109/9.412626.
- 14 Rob J. van Glabbeek and W. Peter Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996. doi:10.1145/233551.233556.
- 15 Jan C. Willems. The behavioral approach to open and interconnected systems. *IEEE Control Systems Magazine*, 27(6):46–99, 2007.

An Explainable GNN Framework for Component-Level Anomaly Diagnosis

Sena Ozgunay ✉ 

LAAS-CNRS, Université de Toulouse, France

Louise Travé-Massuyès ✉ 

LAAS-CNRS, Université de Toulouse, France

Jean-Michel Loubes ✉ 

INRIA, Toulouse, France

Raul Sena Ferreira ✉ 

Airbus Protect, Toulouse, France

Abstract

Industrial processes are complex systems composed of multiple interacting sensors that generate multivariate time series (MTS). Detecting anomalies in such systems is critical for reliability and safety, yet understanding their origin is equally important. Existing Graph Neural Network (GNN)-based methods for anomaly detection primarily focus on sensor-level deviations and either attribute anomalies directly to the deviating sensors. When diagnosis is attempted, generally, the most deviated sensor is identified as a root cause of a system fault. However, in many industrial systems, anomalies do not arise from faulty sensors but from disruptions in the influences governing the system dynamics. We propose an explainable GNN-based anomaly detection framework that shifts the perspective from sensor-level anomalies to component-level diagnosis, hypothesizing that anomalous measurements are symptoms of altered inter-sensor influences. Experiments show that the method effectively identifies and prioritizes the true faulty components, providing interpretable insights into system failures.

2012 ACM Subject Classification Computing methodologies

Keywords and phrases Anomaly Detection, Fault Diagnosis, Graph Neural Networks, Time Series

Digital Object Identifier 10.4230/OASICS.DX.2026.2

Supplementary Material *Software (Source Code)*: <https://github.com/sozgny/XS2C>
archived at `swb:1:dir:b553c65cd6c307b50127cc9d7fd5e4cfd4b27962`

Funding This work is supported by the AI Interdisciplinary Institute ANITI, funded by the France 2030 program under the Grant agreements n° ANR-19-PI3A-0004 and n° ANR-23-IACL-0002.

Acknowledgements The authors would also like to thank Ferhat Tamssaouet for valuable support and insightful discussions for this work.

1 Introduction

Detecting anomalies in industrial systems is insufficient for understanding their origins, especially in complex processes with strongly interconnected sensor measurements. Graph Neural Networks (GNNs) capture such dependencies through a graph where nodes are sensors and edges are their interactions [23], and their message-passing mechanism has made them an effective tool widely used for anomaly detection in multivariate time series [6, 29].

However, detection alone leaves the cause of the failure unresolved. In GNN-based frameworks, diagnosis is occasionally addressed by attributing anomalies to the most deviated sensors [6, 19, 14, 18] or by ranking their contributions [32, 33, 30, 10]. Yet, in many industrial settings, anomalies do not originate from sensors but from physical components



© Sena Ozgunay, Louise Travé-Massuyès, Jean-Michel Loubes, and Raul Sena Ferreira;
licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 2; pp. 2:1–2:17

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

that disrupt the influences governing the system dynamics. Functional sensors may then produce anomalous measurements due to perturbations: for instance, a pipe leak in a water distribution network alters flow dynamics and yields abnormal readings from multiple healthy sensors [8]. This highlights the need to move beyond sensor-level analysis and consider the underlying sensor influences when performing anomaly diagnosis. We therefore argue that anomalous measurements should be interpreted as *symptoms* rather than faults. This perspective is naturally aligned with model-based diagnosis (MBD) [22, 17], where a consistency-based diagnosis identifies a subset of components that, assumed faulty, makes the system description consistent with the observations. We adopt this principle by treating the graph structure as the system description and the inter-sensor influences as the components to be diagnosed, with anomaly-detection results as observations.

In this paper, we propose the XS2C (eXplainable Sensor-to-Component) framework, which integrates an attention-based GNN with consistency-based diagnosis to identify diagnoses for system anomalies interpreted as symptoms of faults. To the best of our knowledge, no prior method has jointly leveraged these two approaches in a unified manner. In summary, the contributions of this work are as follows:

1. An adaptation of the Graph Deviation Network (GDN) [6] for anomaly detection, including a graph construction strategy based on Neural Granger Causality (NGC) and a sensor-wise anomaly thresholding.
2. A fine-tuning of the model with test data containing anomalies to learn the abnormal behavior of the system and computing attention changes of influences based on the model’s pre- and post-finetuning states.
3. The identification of diagnoses for the system anomalies using path search in the graph structure and consistency-based diagnosis reasoning.
4. The generation of a global explanation consisting of ranked diagnoses based on the attention changes.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 presents our framework. In Section 4, we demonstrate the effectiveness of the proposed method. Section 5 shows our conclusions and future works.

2 Related work

In the real world, MTS data often lack an explicit graph describing the dependencies among variables, yet such a graph is a crucial input for GNNs. This has motivated a growing body of work on graph construction/learning, where the graph is either learned jointly with the predictive model [6, 33] or inferred beforehand, computing similarity metrics [15], using separate algorithms [19, 28], or introducing domain knowledge [30, 8]. For instance, GDN [6] learns a graph structure end-to-end using cosine similarity between learnable node embedding vectors, which might be called global node representations. The graph structure is particularly important because anomaly detection and diagnosis are not only about accurate prediction but also about generating meaningful explanations by identifying which graph interactions are responsible for system anomalies. Among these strategies, Neural Granger Causality (NGC) [24] is particularly attractive, as it can capture directed, data-driven influences between sensors. These causal influences are expected to provide more information and greater interpretability for explaining anomalies than other graph construction/learning strategies.

Numerous GNN-based anomaly detection methods have been proposed for MTS. They generally aim to learn the system’s normal behavior and then detect anomalies by analyzing deviations in sensor measurements. This analysis can be achieved using a reconstruction-

based approach or a forecasting-based approach. Reconstruction-based approaches aim to rebuild the given input MTS values by accurately reproducing normal inputs but fail to reconstruct abnormal observations [31, 18]. Forecasting-based methods focus on predicting future MTS values based on past values and detecting anomalies by the deviations from observations [6, 11, 33]. Indeed, GDN [6] employs an adaptive version of GAT that integrates global and local node representations for attention-based forecasting. For each forecast time tick, GDN computes an anomaly score for each sensor, then combines them to obtain a global anomaly score for that time tick to determine whether it is an anomaly. In the literature, reconstruction and forecasting strategies are combined to detect anomalies based on predictions from reconstruction and forecasting probabilities [32, 8]. While these models can effectively detect anomalous patterns, their output is usually limited to anomaly scores or flagged time ticks, leaving the origin/root cause of the anomalies unresolved.

The use of attention mechanisms within GNNs, particularly in a Graph Attention Network (GAT) [25], has gained significant importance in recent years, as attention coefficients can catch most relevant sensor influences in the system by assigning importance scores to edges for a target node. However, the interpretability of attention as explanation still remains a debated topic [13, 26, 1]. The role of attention in anomaly explanation remains controversial, as it may not correspond to a faithful explanation of model predictions. In MTS, however, attention can still provide a useful signal about changes in sensor influences, as an indicator of how an influence evolves between normal and faulty system conditions.

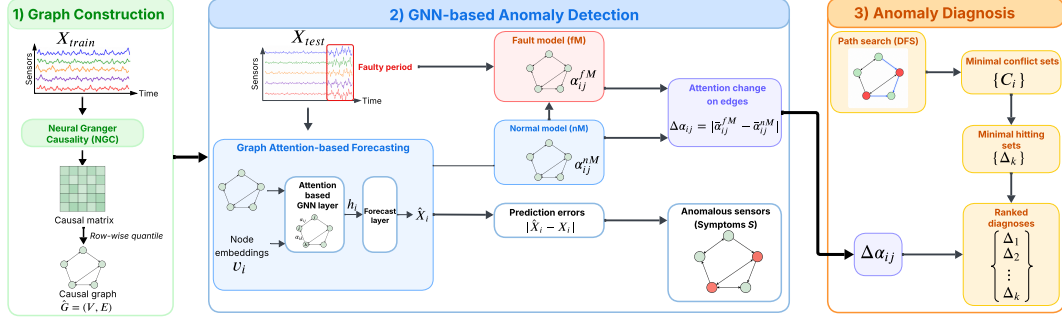
3 Proposed Framework

3.1 Overview

The overall architecture of XS2C is given in Figure 3. We consider a multi-sensor system with n sensors represented as an MTS and assume the following hypotheses:

- **Hypothesis 1.** *In normal conditions, sensor measurements are consistent with the underlying system relationships.*
- **Hypothesis 2.** *Under faulty conditions, sensor functionality remains intact, with abnormal sensor measurements resulting from system faults.*
 - *An anomaly in a sensor measurement is a symptom.*
 - *Anomalies result from alterations in the relationships between measured quantities. In other words, when dependencies between quantities are altered, some measurements exhibit unexpected behavior.*

In the context of semi-supervised learning, our training data over T_{train} time ticks, $X_{train} \in \mathbb{R}^{n \times T_{train}}$, consists solely of normal observations to learn the system's normal behavior. We denote the learned causal graph from X_{train} using NGC as $\hat{G} = (V, E)$ where $V = \{1, \dots, n\}$ is the set of nodes/sensors, and $E \subseteq V \times V$ is the set of edges/influences. The first objective is to detect anomalies in test data $X_{test} \in \mathbb{R}^{n \times T_{test}}$. At time $t \in T_{test}$, we detect which sensor measurement(s) is/are anomalous, i.e., $\hat{y}_i(t) \in \{0, 1\}$, $i \in V$, with 0 indicating the normal class and 1 the anomaly class. If ground truth time tick labels are only available at the system level, the sensor-level status is aggregated as $\hat{y}(t) = 1$ if $\exists i \in V$ such that $\hat{y}_i(t) = 1$. The GNN anomaly detection module is based on an attention mechanism, outputting attention coefficients $\alpha_{ij}(t)$ on the edges from a neighbor node $j \in V$ to a target node $i \in V$, and detecting abnormal nodes. The second objective is to diagnose anomalies. Here, we fine-tune our base model on X_{test} to learn the system's abnormal behavior. We then



■ **Figure 1** Our framework is decomposed into three components: *Graph construction*, *GNN-based anomaly detection*, and *anomaly diagnosis*. In the *graph construction*, a causal graph is extracted from X_{train} using NGC. For the *GNN-based anomaly detection*, an attention-based forecasting is applied on X_{test} based on the graph structure. Symptoms are identified for the next step. During the fault period, we fine-tune the model on X_{test} to capture the system behavior under faulty conditions. Attention changes on edges are computed based on the model's pre- and post-finetuning states. In the *anomaly diagnosis* step, the symptoms and attention changes are then used to generate ranked diagnoses to identify faulty system components.

compute changes in edge attention to capture the evolution between normal and faulty system states. We compute diagnoses from nodes detected as abnormal via path search in the graph, leveraging the consistency-based diagnosis theory [22]. The output is a global explanation of the system fault and ranked diagnoses based on attention changes. In summary, XS2C has three components: *graph construction*, *GNN-based anomaly detection*, and *anomaly diagnosis*, as detailed in the next subsections.

3.2 Graph Construction

In case of no prior graph, our objective is to construct a directed causal graph from X_{train} using NGC [24]. A variable $j \in \{1, \dots, n\}$ Granger-causes a variable $i \in \{1, \dots, n\}$ if knowledge of the past of j improves the prediction of the future of i . In NGC, we quantify the temporal influence of j on i by modeling each sensor as a neural network and extracting the Granger causal structure by forcing some sets of weights to zero via group penalties. For a sensor i at time t , we define:

$$\hat{x}_i(t) = G_i(x_1(< t), \dots, x_n(< t)) + \epsilon_i(t) \quad (1)$$

where $x_j(< t) = (x_j(t-K), \dots, x_j(t-2), x_j(t-1))$ are the inputs of j for past K lags, and $\epsilon_i(t)$ is a zero-mean noise. As a neural model G_i , we use a component-wise MLP (cMLP) [24]. Only the first layer is used to infer causal relationships, since it directly maps the lagged input variables to the target variable. Hence, the weights of the first layer are divided into blocks by time lags $W^1 = \{W^{11}, \dots, W^{1k}, \dots, W^{1K}\} \in \mathbb{R}^{H \times (nK)}$, where H is the hidden layer dimension and $W^{1k} \in \mathbb{R}^{H \times n}$ captures the contribution of inputs at lag k . For a source variable j , its overall influence on the target i is determined by the collection $W_j^1 = \{W_{:,j}^{11}, \dots, W_{:,j}^{1K}\}$, where $W_{:,j}^{1k} \in \mathbb{R}^H$ denotes all connections from j to the hidden layer at lag k .

A group-lasso penalty is then applied to these weights, such that if all corresponding weights are shrunk to zero, variable j is considered not to Granger-cause variable i . We consider this penalty as a causal score $s_{ij} = \|W_j^1\|_F$, where $\|\cdot\|_F$ is the Frobenius matrix norm, and s_{ij} is the causal influence from the source sensor j to the target sensor i . We then convert these causal scores into a directed graph by retaining the source sensors whose causal

scores exceed a row-wise quantile threshold q as neighbors of i : $\mathcal{N}_i = \{j \mid s_{ij} \geq Q_i(q), j \neq i\}$, where $Q_i(q)$ is the q -th quantile of the causal scores in row i . Thus, a directed causal graph $\hat{G} = (V, E)$ is obtained from X_{train} with $E = \{e_{ji} \mid j \in \mathcal{N}_i\}$ and used as a fixed structure in the rest of the methodology, ensuring causally meaningful edges and a stable, interpretable system.

3.3 GNN-based Anomaly Detection

We deploy the attention-based forecasting architecture of the GDN [6] to predict sensor measurements at time t based on a historical sliding window of size w :

$$[X(t-w), X(t-w+1), \dots, X(t-1)] \in \mathbb{R}^{n \times w} \quad (2)$$

3.3.1 Attention-based Forecasting

First, each sensor $i \in \{1, \dots, n\}$ is represented with a global embedding vector that captures its characteristics: $v_i \in \mathbb{R}^d$, where d is the embedding dimension. These vectors are initialized randomly and trained with the model to capture the underlying factors that shape nodes' different behaviors. Given a historical window as in 2, the message-passing mechanism computes node i 's aggregated local representation via an attention mechanism enhanced with global sensor embeddings, unlike GAT [25]. At time t , attention coefficients $\alpha_{ij}(t)$ are computed as follows:

$$g_i(t) = v_i \oplus W X_i(t) \quad (3)$$

$$\phi(i, j) = \text{LeakyReLU}(a^\top (g_i(t) \oplus g_j(t))) \quad (4)$$

$$\alpha_{ij}(t) = \frac{\exp(\phi(i, j))}{\sum_{k \in \mathcal{N}(i) \cup i} \exp(\phi(i, k))} \quad (5)$$

where $W \in \mathbb{R}^{d \times w}$ is a trainable weight matrix, $X_i(t) \in \mathbb{R}^w$ is the node i 's input measurements, $a^\top \in \mathbb{R}^{4d}$ is a learned attention weight vector. The node representation is then:

$$h_i(t) = \text{ReLU} \left(\alpha_{ii}(t) W X_i(t) + \sum_{j \in \mathcal{N}_i} \alpha_{ij}(t) W X_j(t) \right) \quad (6)$$

After obtaining all final local node representations $h_i(t), i = \{1, \dots, n\}$, we predict the sensor measurements at time t as:

$$\hat{X}(t) = f_\theta([v_1 \circ h_1(t), \dots, v_n \circ h_n(t)]) \in \mathbb{R}^n \quad (7)$$

where $\circ$ is the element-wise multiplication and f_θ is an MLP. The training process is achieved by minimizing the Mean Squared Error (MSE) loss, which is computed as follows:

$$\mathcal{L}_{MSE} = \frac{1}{T_{train} - w} \sum_{t=w+1}^{T_{train}} \|\hat{X}(t) - X(t)\|_2^2 \quad (8)$$

3.3.2 Computing thresholds and detecting anomalies

Traditional MTS anomaly detection methods use a single threshold based on aggregated anomaly scores to detect abnormal time ticks at the system level [6, 32, 33]. GDN [6], for instance, computes per-sensor scores but applies a max-based thresholding at the system

level, which often ignores gradual or sensor-specific anomalous behaviors in favor of the largest immediate deviations. However, in real-world systems, sensors might measure different quantities, leading to heterogeneity in abnormality. That is why we adopt a sensor-wise thresholding strategy to detect anomalous sensor measurements and subsequently identify them as symptoms. For each sensor, we compute an anomaly score based on its prediction errors at time t : $\mathcal{A}_i(t) = |X_i(t) - \hat{X}_i(t)|$. On the validation set T_{val} (containing only normal measurements), we collect $\mathcal{A}_i^{val} = \{\mathcal{A}_i(t) \mid t \in T_{val}\}$ and adaptively select the quantile level matching a target positive prediction rate (PPR) p_{target} , which controls the expected false alarm behavior $q^* = \operatorname{argmin}_q |p_{val}(q) - p_{target}|$, where $p_{val}(q)$ is the empirical proportion of time ticks flagged as anomalous in validation data when using quantile q . The sensor-wise threshold is then $\tau_i = Q_{\mathcal{A}_i^{val}}(q^*)$. At $t \in T_{test}$, if $\mathcal{A}_i(t) > \tau_i$, we detect t as anomalous, i.e., $\hat{y}_i(t) = 1$.

3.3.3 Identifying symptoms

Given the binary anomaly predictions for each sensor over X_{test} , we compute the *anomaly occurrence* for each sensor, particularly in the faulty period $T_f \subseteq T_{test}$. For a sensor i , its anomaly occurrence in T_f is computed as follows:

$$O_i = \sum_{t \in T_f} \hat{y}_i(t) \quad (9)$$

Symptoms are identified on the sensors with the highest anomaly occurrence. Sensors are sorted as $O_{i_1} \geq O_{i_2} \geq \dots \geq O_{i_n}$, and the largest gap between consecutive values $d_k = O_{(k)} - O_{(k+1)}$ defines a cut point:

$$k^* = \operatorname{argmax}_k d_k \quad (10)$$

This criterion identifies the point of maximum separation between highly anomalous sensors and the remaining ones, allowing for a data-driven selection of symptoms:

$$S = \{i_1, i_2, \dots, i_{k^*}\} \quad (11)$$

3.4 Anomaly Diagnosis

We aim to generate a global explanation of the set of detected symptoms in the test data X_{test} . As detailed below, our approach first quantifies the changes of the attention coefficients given by (3) associated with the graph's edges from the GNN normal model to the faulty one and then uses these changes to rank the diagnoses obtained via the theory of model-based diagnosis along the consistency-based approach [22], which takes the graph of the GNN as a model.

3.4.1 Fine-tuning of the GNN model

We consider two models trained under normal and faulty operating conditions:

- The *normal model* (nM) is the base GNN model trained on the normal train set X_{train} and tested on X_{test} containing anomalies.
- The *fault model* (fM) is a fine-tuned, semi-frozen version of the normal model. The semi-frozen GNN is retrained and tested only on the data collected after a faulty occurrence in X_{test} . We assume faults to be strong and persistent, i.e., once a fault occurs, its effects remain present throughout the considered faulty period T_f and continuously affect

the system behavior. Consequently, the subsequent occurrence data are assumed to predominantly reflect the faulty operating mode, allowing the fine-tuned model to learn the altered sensor influences associated with the fault.

From the normal model, the semi-frozen GNN retains the graph topology ($\hat{G}$) and the global sensor embeddings (v_i). The remaining learnable weights are initialized using the weights from the normal model. This strategy allows us to highlight differences between the two models that reflect true behavioral changes – rather than structural variations – particularly in the attention coefficients on the edges. These coefficients capture changes in the interactions between quantities measured by the sensors, providing signals to identify root-cause faulty components in the system.

3.4.2 Computing attention changes

The attention coefficients on all forecasted time ticks $t \in T_f$ are computed for both models, yielding $\alpha_{ij}^{\text{nM}}(t)$ and $\alpha_{ij}^{\text{fM}}(t)$. Since we aim at a global explanation, we average them over T_f :

$$\bar{\alpha}_{ij}^{\text{nM}} = \frac{1}{T_f} \sum_{t \in T_f} \alpha_{ij}^{\text{nM}}(t) \quad (12)$$

$$\bar{\alpha}_{ij}^{\text{fM}} = \frac{1}{T_f} \sum_{t \in T_f} \alpha_{ij}^{\text{fM}}(t) \quad (13)$$

Then, we compute absolute differences of average attentions, $\Delta\alpha_{ij} = |\bar{\alpha}_{ij}^{\text{fM}} - \bar{\alpha}_{ij}^{\text{nM}}|$, which will be considered as attention changes of influences.

3.4.3 Bridging with consistency-based diagnosis

The consistency-based diagnosis approach, originally developed using logical models [22, 5], has been explored in recent works integrating model-based reasoning with data-driven learning, often through residual classifiers or learned representations [27, 16]. In this work, we propose using this approach to explain detected anomalies by identifying the sets of faulty components that may have caused them.

Formally, a *system* is defined by the triplet (SD, COMPS, OBS), where SD is the *system description*, COMPS is the *set of components*, and OBS is the *set of observations*. A system description describes how the components normally behave by appealing to the distinguished predicate AB whose negation $\neg AB$ has the intended meaning “normal” [22]. In our setting, this triplet is defined as:

- SD is the graph $\hat{G}$ composed of sensors and influences.
- COMPS are the influences, since they can be associated with the physical components that create them, i.e., $\text{COMPS} = E$.
- OBS are defined by symptomatic labels $l_i \in \{0, 1\}$, where $l_i = 1$ indicates that sensor i is a symptom.

► **Definition 3 (Diagnosis).** A diagnosis for the system (SD, COMPS, OBS) is a set $\mathcal{D} \subseteq \text{COMPS}$ such that $\text{SD} \cup \text{OBS} \cup \{AB(e) \mid e \in \mathcal{D}\} \cup \{\neg AB(e) \mid e \in \text{COMPS} - \mathcal{D}\}$ is satisfiable.

This implies that the hypothesis – where the influences in $\mathcal{D}$ are considered faulty, and all others are normal – is consistent with both OBS and SD. A diagnosis thus assigns a mode (normal or faulty) to each graph influence in a way that aligns with the model and the observations.

► **Definition 4** (Minimal diagnosis). *A minimal diagnosis is a diagnosis $\mathcal{D}$ such that $\forall \mathcal{D}' \subset \mathcal{D}$, $\mathcal{D}'$ is not a diagnosis.*

Typically, the set of diagnoses is obtained through a two-step process, with the first step grounded in the concept of conflict – originally introduced in [22] and expanded in [4].

► **Definition 5** (Conflict and minimal conflict). *A conflict is a set $\mathcal{C} \subseteq COMPS$ in which all the components of $\mathcal{C}$ are normal is not consistent with SD and OBS . A minimal conflict is a conflict that does not contain other conflicts.*

As shown by [22], minimal diagnoses correspond to the hitting sets ¹ of the set of minimal conflicts, and the parsimony principle suggests favoring minimal diagnoses. In our setting, the first step for identifying them is to search $\hat{G}$ for the paths π that end in a symptom node, start in a non-symptom node, and have no cycles:

$$\begin{cases} \pi = (i_k, i_{k-1}, \dots, i_1, i_0), \\ \forall j \text{ s.t. } 0 \leq j \leq k-1, i_j \in S, \\ i_k \notin S, \\ i_j \neq i_l, j \neq l. \end{cases} \quad (14)$$

In this search, starting from the symptoms, we explore the graph by iteratively visiting its neighboring nodes, stopping once a non-symptom node is reached. This stopping criterion is motivated by the assumption that a faulty influence should manifest through observable deviations, i.e., symptoms: an influence between two non-symptom nodes is unlikely to be the root cause of the system fault, as it would not produce any detectable effect. This reasoning aligns with an exoneration principle [3]: components that do not contribute to any abnormal behavior are excluded from the set of root causes.

For path search, we use a depth-first search (DFS) algorithm. At each step, the state is the triplet $(i_l, V_{\text{vis}}, E_{\text{path}})$, where $i_l \in V$ is the current node, $V_{\text{vis}} \subseteq V$ the set of nodes already visited on the current path, and $E_{\text{path}} \subseteq E$ the set of edges traversed up to i_l .

Transition Rules: For any incoming edge $e_{l,l-1} = (i_l, i_{l-1}) \in E$, we apply the following rules:

- If $i_l \notin S$, the path is closed and a path is generated $\pi = E_{\text{path}} \cup \{e_{l,l-1}\}$;
- If $i_l \in S$ and $i_l \notin V_{\text{vis}}$, the procedure continues recursively from i_l ;
- Otherwise, the transition is ignored to avoid any repetition of the node or cycle.

For each $i \in S$, we denote the *set of edge paths*: $\pi_i = \{\pi \mid \pi \text{ is an admissible path}\}$. The conflict associated with symptom i is defined as the union of the influences appearing in these paths:

$$\mathcal{C}_i = \bigcup_{\pi \in \pi_i} E(\pi), \quad (15)$$

where $E(\pi)$ denotes the set of edges belonging to path π . Since anomalies may propagate through reciprocal interactions between process variables, a faulty bidirectional influence relation is considered capable of covering the conflicts in which it appears, regardless of edge direction. Consequently, reciprocal edges are treated as a single faulty interaction during diagnosis generation; only the direction exhibiting the largest attention change is reported for representation purposes.

¹ A set that intersects every set in the collection.

After obtaining all conflict sets $\{C_i \mid i \in S\}$, we retain the minimal ones. We then find the minimal hitting set(s) using the *Hitman* solver based on a MaxSAT formulation [21] via the PySAT toolkit [12]. Each conflict set C_i is translated into a hard clause $(e_{k,k-1} \vee \dots \vee e_{l,l-1} \vee \dots \vee e_{1,0})$ stating that at least one edge must be chosen in each C_i , and each unique edge induces a soft clause $\neg e_{l,l-1}$ penalizing its inclusion. These minimal hitting sets correspond to *minimal diagnoses* [3].

As a global explanation of system anomalies, we aim to rank these diagnoses using attention changes. The attention changes of influences are computed as $\Delta\alpha_{ij}$ in Section 3.4.2. If a diagnosis consists of a single influence, its ranking score is directly given by this value. For a diagnosis $\mathcal{D}$ containing multiple influences, we compute the ranking score as the average of the attention changes of its constituent influences:

$$\text{score}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{e_{ji} \in \mathcal{D}} \Delta\alpha_{ij} \quad (16)$$

Diagnoses are then ranked by ranking scores in descending order. The intuition is that fault-related propagation should induce larger changes in the corresponding sensor influences, so diagnoses with higher average attention changes are considered more plausible explanations of the observed anomalous behavior.

4 Experiments

In this section, we conduct experiments to evaluate the XS2C’s anomaly detection performance and the capacity of its anomaly diagnosis component.

4.1 Experimental Setup

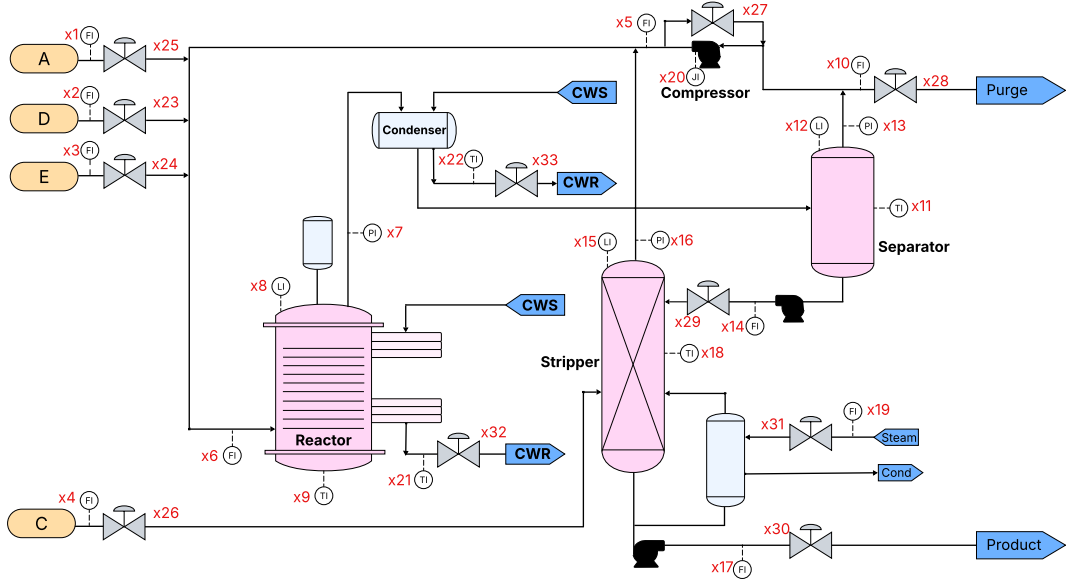
4.1.1 Datasets

We evaluate XS2C on two widely used benchmarks for MTS anomaly detection: TEP and SWaT. More details about the TEP and SWaT can be found in [7, 9].

- **Tennessee Eastman Process (TEP)** is an industrial chemical process for producing two liquid components (G and H) from four gaseous reactants (A, C, D, and E). It consists of five main unit operations (reactor, condenser, liquid-vapor separator, recycling compressor, and product stripper). TEP contains 41 process measurements and 12 manipulated variables. We consider the first 22 processes and 11 manipulated variables, as the remaining measurements are not continuously monitored, and the last variable is constant [28]. The TEP variables used in this study are shown in Table 1, and the process is illustrated in Figure 2.
- **The Secure Water Treatment (SWaT)** dataset is a scaled-down industrial water treatment testbed developed by the Singapore Public Utilities Board [20]. It includes 7 days of normal operation for training and 4 days of attack scenarios for testing, where anomaly labels correspond to injected attacks at different time intervals. .

We use a Python-based simulator² for TEP data generation [2], operating in *closed-loop* control mode. We simulate 7 days of normal operation for training. For evaluation, 20 different fault scenarios IDV(1-20) are simulated over 48 hours, where the first 8 hours correspond to normal operation followed by the fault period, except for IDV6 and IDV18,

² <https://github.com/jkitchin/tennessee-eastman-profbraatz>



■ **Figure 2 TEP flowchart.** The considered process variables are in red text.

whose simulations were terminated after ~ 7 and ~ 12 hours of the faulty period, respectively, due to the violation of the shutdown limits. The simulation yields 20 test datasets with measurements recorded every 3 minutes.

Following prior works [6, 33], the first 21,600 samples are removed from the SWaT dataset, and the data is down-sampled to one measurement every 10 seconds using median values. Table 2 reports the statistics of datasets.

■ **Table 1** TEP variables used in this study.

Variable	Description	Variable	Description
x1	A feed	x18	Stripper temperature
x2	D feed	x19	Stripper steam flow
x3	E feed	x20	Compressor work
x4	A and C feed	x21	Reactor cooling water outlet temperature
x5	Recycle flow	x22	Separator cooling water outlet temperature
x6	Reactor feed rate	x23	D feed flow
x7	Reactor pressure	x24	E feed flow
x8	Reactor level	x25	A feed flow
x9	Reactor temperature	x26	A and C feed flow
x10	Purge rate	x27	Compressor recycle valve
x11	Product separator temperature	x28	Purge valve
x12	Product separator level	x29	Separator pot liquid flow
x13	Product separator pressure	x30	Stripper liquid product flow
x14	Product separator underflow	x31	Stripper steam valve
x15	Stripper level	x32	Reactor cooling water flow
x16	Stripper pressure	x33	Condenser cooling water flow
x17	Stripper underflow		

4.1.2 Baselines

We compare our anomaly detection component with three representative GNN-based frameworks for MTS anomaly detection: GDN [6], MTAD-GAT [32], and CST-GL [33].

■ **Table 2** Statistics of datasets.

Datasets	#Variables	#Train	#Test	Anomalies
TEP (IDV1-5,7-17,19-20)	33	3330	930	86%
TEP (IDV6)	33	3330	254	51.18%
TEP (IDV18)	33	3330	563	23.09%
SWaT	51	47515	44986	11.97%

- **GDN** learns a graph structure end-to-end using cosine similarity between learnable node embedding vectors and applies a GAT-based forecasting as explained in Section 3.3. For anomaly detection, GDN computes a per-sensor anomaly score at each time tick and then aggregates them into a system-level score through max-based thresholding.
- **MTAD-GAT** is a hybrid forecasting-reconstruction anomaly detection method that uses two GAT layers; one for the feature dimension and one for the temporal dimension to capture both feature-wise and temporal dependencies. It detects anomalous time ticks using a joint score derived from predictions and reconstruction probabilities.
- **CST-GL** is a forecasting-based anomaly detection method that explicitly captures multivariate correlations through graph learning. It combines graph-convolution and temporal-convolution layers to model spatial-temporal dependencies and identifies time tick anomalies based on forecasting deviations.

4.1.3 Parameter settings

For the NGC graph learning module, hyperparameters are optimized via random search on X_{train} : a cMLP model is trained with the Adam optimizer, combining group-lasso regularization for sparse causal structure learning and ridge regularization for weight stabilization (learning rate and group-lasso coefficient (λ) are 0.001) [24]. Configurations are evaluated based on the average MSE loss across all variables.

The GNN module is trained for up to 100 epochs using the Adam optimizer ($lr=0.001$), with early stopping patience of 15 and a validation ratio of 0.1. A self-loop is added to each node to enhance the attention-based message passing mechanism. Hyperparameters are selected by empirical search minimizing the validation MSE; the final settings for both stages are given in Table 3 (NGC: lag l , hidden layer dimension H , ridge coefficient λ_{ridge} , quantile q used for causal graph extraction; GNN: batch size b , sliding window w , node embedding dimension d , forecasting layer hidden dimension h , and target positive prediction rate p_{target}).

Baselines reproduced from corresponding github repositories are used ³ with sliding window size $w=5$ (30) for SWaT (TEP). On SWaT, the default settings reported by GDN and CST-GL are kept; on TEP, an empirical search is performed around their default values. The main settings are reported as SWaT (TEP): **GDN** – 50 epochs (early-stopping patience of 10), batch size 16, embedding dimension 64, forecasting layer hidden dimension 128 (256), neighbor size 15 (3). **CST-GL** – 20 epochs, batch size 64, embedding dimension 256, retain ratio 0.1 (0.3), saturation rate 20 (10), neighbor size 15 (30), 2 (3) temporal-convolution layers with 16 (64) channels, 2 graph-convolution layers with 16 channels, 32 skip channels and 64 (128) end channels. **MTAD-GAT** – 50 epochs (with dropout 0.3), batch size 64, initial convolution kernel size 7, one GRU layer (150 units), three forecasting fully-connected layers (150 units), and one reconstruction layer (128 units).

³ <https://github.com/d-ailin/GDN>, <https://github.com/huankoh/CST-GL>, <https://github.com/>

■ **Table 3** Parameter settings for the NGC and GNN modules in XS2C.

Dataset	NGC-Graph Learning				GNN Module in XS2C				
	l	H	λ_{ridge}	q	b	w	d	h	p_{target}
TEP	16	64	0.001	0.90	16	30	64	256	0.09
SWaT	4	128	0.0001	0.95	16	5	64	64	0.06

4.1.4 Evaluation metrics

To evaluate the performance of the GNN anomaly detection model, since only time-tick-level ground truth labels are available, we aggregate sensor-level predictions to obtain a system-level decision: t is classified as anomalous if at least $N \leq n$ sensors are simultaneously flagged as anomalous: $\hat{y}(t) = 1$ if $\sum_{i=1}^n \hat{y}_i(t) \geq N$. This rule reduces spurious detections caused by sensor noise and reflects that a single underlying fault can manifest across multiple sensors. We use $N = 2$ for TEP, since faults are reflected by localized deviations affecting a limited subset of process variables, and $N = 15$ for SWaT, since attacks typically cause process-wide consequences (tank overflows, underflows, shutdown sequences, flow disruptions affecting multiple stages), so a larger aggregation reduces false alarms from isolated deviations.

Detection performance is evaluated with the macro-F1 score (F_1) and the Matthews Correlation Coefficient (MCC). Macro-F1 ensures a balanced evaluation of normal and anomalous classes; MCC provides a robust global assessment by simultaneously considering true and false positives and negatives, making it more reliable than accuracy across datasets with varying class distributions.

4.2 Anomaly detection performance

Table 4 reports anomaly detection results on TEP (across 20 fault scenarios) and SWaT. On TEP, the anomaly detection component of our proposed framework XS2C has the best overall performance compared to the baseline. Particularly, we achieve substantially improved performance across most fault scenarios compared with the GDN. Moreover, we are competitive with MTAD-GAT and CST-GL by achieving the best performance on several scenarios, particularly those associated with persistent or structured abnormal dynamics such as IDV(4, 10, 12, 13, 16, 17, 18, 19, 20). Lower performance is consistently observed across all methods on disturbances such as IDV (3, 5, 9, 15), known to involve weaker deviations without abnormal dynamics [19, 14, 10].

On SWaT, we remain competitive with the baseline with a slightly better performance in both F_1 and MCC, demonstrating that our proposed framework can handle large industrial cyber-physical systems while maintaining explainability. Despite anomaly explanation and diagnosis being our primary objective of the proposed framework, we demonstrate an effective anomaly detection performance on two well-known benchmarks. Scenarios with high F_1 and MCC should be retained for the next step in anomaly diagnosis, since reliable detection is necessary to provide sufficient symptom information.

■ **Table 4** Anomaly detection results. The best performing method in each experiment is in bold.

Datasets	GDN		MTAD-GAT		CST-GL		Our AD component	
	F_1	MCC	F_1	MCC	F_1	MCC	F_1	MCC
IDV1	0.28	0.16	0.99	0.98	0.99	0.98	0.98	0.96
IDV2	0.27	0.12	0.96	0.92	0.93	0.87	0.95	0.90
IDV3	0.17	0.05	0.12	-0.03	0.17	0.05	0.15	0.02
IDV4	0.17	0.06	0.26	0.14	0.23	0.11	0.98	0.97
IDV5	0.34	0.21	0.34	0.19	0.37	0.21	0.35	0.19
IDV6	0.75	0.60	1	0.99	0.99	0.98	0.99	0.98
IDV7	0.35	0.21	0.87	0.77	0.62	0.43	0.48	0.30
IDV8	0.47	0.30	0.93	0.86	0.88	0.79	0.92	0.85
IDV9	0.16	0.05	0.12	-0.03	0.16	0.04	0.16	0.02
IDV10	0.20	0.10	0.18	0.08	0.46	0.28	0.74	0.58
IDV11	0.19	0.08	0.40	0.24	0.43	0.26	0.57	0.38
IDV12	0.46	0.29	0.90	0.82	0.93	0.86	0.97	0.94
IDV13	0.36	0.22	0.80	0.65	0.76	0.61	0.84	0.72
IDV14	0.16	0.03	0.99	0.98	1	0.99	0.96	0.92
IDV15	0.17	0.07	0.13	-0.02	0.18	0.07	0.15	0.02
IDV16	0.20	0.09	0.14	0.02	0.48	0.30	0.71	0.54
IDV17	0.41	0.24	0.77	0.62	0.87	0.76	0.90	0.82
IDV18	0.60	0.42	0.91	0.83	0.90	0.81	0.94	0.88
IDV19	0.17	0.06	0.16	0.06	0.46	0.28	0.54	0.35
IDV20	0.27	0.14	0.49	0.31	0.64	0.45	0.71	0.54
Average	<i>0.31</i>	<i>0.18</i>	<i>0.57</i>	<i>0.47</i>	<i>0.62</i>	<i>0.51</i>	0.70	0.59
SWaT	0.87	0.76	0.39	0.17	0.87	0.76	0.88	0.78

4.3 Anomaly diagnosis results and analysis

In TEP, IDV1 and IDV14 are chosen as case studies for anomaly diagnosis. For each symptom, a local subgraph is extracted from its neighborhood; these subgraphs serve as the basis for searching paths that yield minimal conflicts and, subsequently, minimal diagnoses. Figure 3 illustrates local subgraphs for IDV1 and IDV14, respectively.

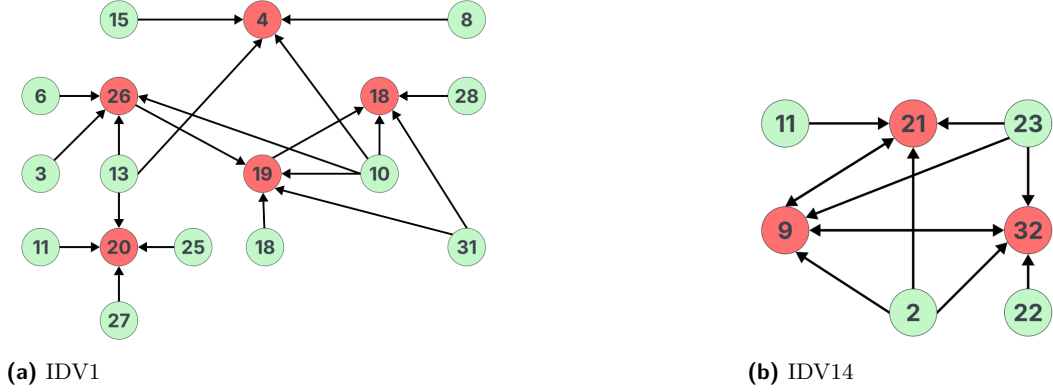
IDV1 is a step change disturbance in the A/C feed ratio in stream 4. The identified symptoms are x_4 (A and C feed in stream-4), x_{18} (stripper temperature), x_{19} (stripper steam flow), x_{20} (compressor work), and x_{26} (manipulated variable related to A and C feed flow in stream-4). The top-3 ranked diagnoses (with attention changes of edges in them) are:

$$\mathcal{D}_1 = \{e_{13,4} (0.82), e_{25,20} (0.04), e_{13,26} (0.14)\} \quad (17)$$

$$\mathcal{D}_2 = \{e_{13,4}, e_{13,20} (0.02), e_{13,26}\} \quad (18)$$

$$\mathcal{D}_3 = \{e_{13,4}, e_{27,20} (0.01), e_{13,26}\} \quad (19)$$

The nodes involved in the edges composing the diagnoses also include x_{13} , the product separator pressure, x_{25} , the A feed flow in stream-1, and x_{27} , the compressor recycle valve. The diagnoses highlight influences associated with the stream-4 feed mechanism and its downstream propagation, which is coherent IDV1. In the literature, this disturbance is often interpreted through the control system response, where the A feed flow (x_{25}) is adjusted to compensate for changes in the feed ratio and is identified as a root-cause variable [19].



■ **Figure 3 Causal subgraphs obtained for IDV1 and IDV14:** A local subgraph is used for identifying diagnoses regarding neighbors. Red nodes correspond to symptoms, and green nodes are non-symptoms.

Although x_{25} is not a symptom in our framework, it still appears in the top-ranked diagnoses through its perturbed influence, illustrating our hypothesis: the objective is not to identify the symptoms themselves as root causes but the altered influences that explain them. The result should thus not be read as identifying x_{25} itself as an isolated root cause, but as indicating that the dependency involving this variable is significantly perturbed during the fault, consistent with the expected control response to a disturbance in stream-4. Moreover, the repeated involvement of relations connected to x_{13} further suggests that the disturbance propagates downstream through the recycle stream until a new steady state is reached, which agrees with the reported behavior of IDV1. Thus, the proposed diagnosis provides a global explanation: it localizes the disturbance around stream-4 feed interactions while capturing its downstream propagation.

Regarding IDV14, the identified symptoms are the reactors x_9 (temperature), x_{21} (cooling water temperature), and x_{32} (cooling water flow). The top-3 diagnoses are:

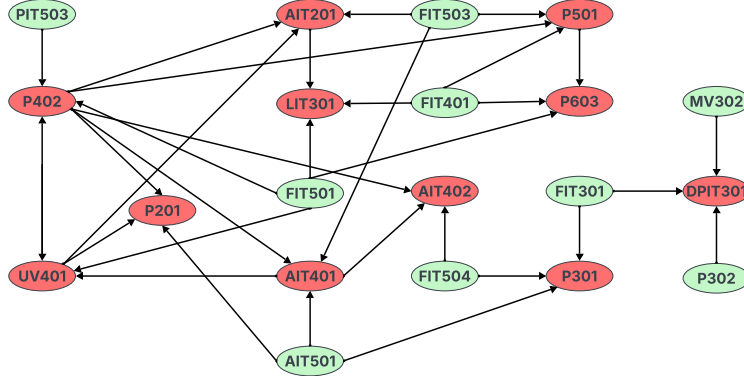
$$\mathcal{D}_1 = \{e_{32,9} \text{ (0.34)}\} \quad (20)$$

$$\mathcal{D}_2 = \{e_{2,9} \text{ (0.15)}\}, \{e_{11,21} \text{ (0.15)}\} \quad (21)$$

$$\mathcal{D}_3 = \{e_{23,9} \text{ (0.14)}\} \quad (22)$$

The nodes involved in the edges composing the diagnoses include x_{11} , the product separator temperature, and x_{23} , the D feed flow in stream-2. The highest-ranked diagnosis $e_{32,9}$ directly links the reactor cooling water flow x_{32} to the reactor temperature x_9 . Since the cooling water mechanism directly regulates the reactor temperature, the perturbation of this influence strongly suggests that the fault originates from the reactor cooling water valve system, consistent with previous studies [19, 18]. Compared to approaches that focus only on abnormal variables, our framework provides a more relational interpretation by explicitly identifying the perturbed dependency underlying the cooling process. The remaining diagnoses remain physically meaningful: a sticking valve is expected to alter several temperature-related influences around the reactor [19], so while $e_{32,9}$ provides the most direct localization, the additional diagnoses support the propagation of the disturbance through the reactor cooling subsystem.

For the SWaT case study, we selected the attack scenario corresponding to the continuous closure of valve $MV303$ between 29/12/2015 14:38:12-14:50:08, which disrupts the backwash process in stage-3 (filtering undesirable materials from the water). The identified symptoms



■ **Figure 4** Causal subgraph obtained for SWaT dataset.

mainly involve variables associated with stage-3 and its downstream propagation. Figure 4 illustrates local subgraph for SWaT. The top-3 diagnoses are:

$$\begin{aligned}
 \mathcal{D}_1 &= \{e_{P302, DPIT301} (0.32), e_{AIT501, P301} (0.11), \\
 &\quad e_{FIT501, UV401} (0.10)\}, \\
 \mathcal{D}_2 &= \{e_{P302, DPIT301} (0.32), e_{AIT501, P301} (0.11), \\
 &\quad e_{AIT501, AIT401} (0.09)\}, \\
 \mathcal{D}_3 &= \{e_{P302, DPIT301} (0.32), e_{AIT501, P301} (0.11), \\
 &\quad e_{PIT503, P402} (0.06)\}.
 \end{aligned} \tag{23}$$

In these diagnoses, the influence $e_{P302, DPIT301}$ between the ultra filtration feed pump $P302$ and the differential pressure transmitter $DPIT301$ of stage-3 systematically appears with the largest attention variation, indicating that the hydraulic and backwash dynamics in this region are considerably perturbed. This is physically coherent with a valve-induced disturbance propagation. The other influences appearing in $\mathcal{D}_1$ may be interpreted as secondary propagation effects: $e_{AIT501, P301}$ (water quality analyzer $AIT501$ to pump $P301$) suggests propagation toward the downstream circulation mechanisms, while $e_{FIT501, UV401}$ (flow transmitter $FIT501$ to dechlorinator $UV401$) indicates that abnormal flow conditions induced by the disrupted backwash further impact the UV treatment stage. The diagnoses thus not only localize the disturbance around the backwash drain valve but also reveal plausible downstream propagation pathways.

5 Conclusion

This work proposes an explainable anomaly detection framework that shifts diagnosis from anomalous sensors toward disrupted sensor influences. By combining an attention-based GNN with consistency-based diagnosis, our approach, XS2C, produces physically coherent and interpretable diagnoses that localize the disturbed process components/regions underlying system faults. Experimental results demonstrate that XS2C can achieve effective anomaly detection and meaningful explanation of fault propagation dynamics in complex systems. Future work will focus on improving the performance of the anomaly detection component, validating XS2C on real-world industrial systems. Additionally, weakening the Hypothesis 2 by including the possibility that the sensor might be faulty via self-attention coefficients will be an interesting avenue for future work. A sensitivity analysis of the framework with respect to numerical non-determinism in training remains an another perspective.

References

- 1 Adrien Bibal, Rémi Cardon, David Alfter, Rodrigo Wilkens, Xiaou Wang, Thomas François, and Patrick Watrin. Is attention explanation? an introduction to the debate. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (volume 1: long papers)*, pages 3889–3900, 2022. doi:10.18653/V1/2022.ACL-LONG.269.
- 2 Leo Chiang, Evan Russell, and Richard Braatz. *Fault Detection and Diagnosis in Industrial Systems*. Springer, 2001. doi:10.1007/978-1-4471-0347-9.
- 3 M-O Cordier, Philippe Dague, François Lévy, Jacky Montmain, Marcel Staroswiecki, and Louise Travé-Massuyès. Conflicts versus analytical redundancy relations: A comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2163–2177, 2004. doi:10.1109/TSMCB.2004.835010.
- 4 Johan De Kleer, Alan K Mackworth, and Raymond Reiter. Characterizing diagnoses and systems. *Artificial intelligence*, 56(2-3):197–222, 1992. doi:10.1016/0004-3702(92)90027-U.
- 5 Johan De Kleer and Brian C Williams. Diagnosing multiple faults. *Artificial intelligence*, 32(1):97–130, 1987. doi:10.1016/0004-3702(87)90063-4.
- 6 Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4027–4035, 2021. doi:10.1609/AAAI.V35I5.16523.
- 7 J Downs and E Vogel. A plant-wide industrial process problem control. *Comput Chem Eng*, 17(3):245–255, 1993.
- 8 Garðar Örn Garðarsson, Francesca Boem, and Laura Toni. Graph-based learning for leak detection and localisation in water distribution networks. *IFAC-PapersOnLine*, 55(6):661–666, 2022.
- 9 Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. A dataset to support research in the design of secure water treatment systems. In *International conference on critical information infrastructures security*, pages 88–99. Springer, 2016. doi:10.1007/978-3-319-71368-7_8.
- 10 Lei Guo, Hongbo Shi, Shuai Tan, Bing Song, and Yang Tao. A knowledge-driven spatial-temporal graph neural network for quality-related fault detection. *Process Safety and Environmental Protection*, 184:1512–1524, 2024.
- 11 Xiangheng Huang, Ningjiang Chen, Ziyue Deng, and Suqun Huang. Multivariate time series anomaly detection via dynamic graph attention network and informer. *Applied Intelligence*, 54(17–18):7636–7658, 2024. doi:10.1007/s10489-024-05575-y.
- 12 Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018. doi:10.1007/978-3-319-94144-8_26.
- 13 Sarthak Jain and Byron C Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, volume 1, pages 3543–3556, 2019. doi:10.18653/V1/N19-1357.
- 14 Guoqian Jiang, Kaili Shen, Xiufeng Liu, Xu Cheng, and Ping Xie. Hierarchical spatiotemporal graph network for fault diagnosis of industrial processes. *IEEE Internet of Things Journal*, 12(3):3043–3054, 2024. doi:10.1109/JIOT.2024.3476287.
- 15 Guangyin Jin, Yuxuan Liang, Yuchen Fang, Zezhi Shao, Jincai Huang, Junbo Zhang, and Yu Zheng. Spatio-temporal graph neural networks for predictive learning in urban computing: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2023.
- 16 Daniel Jung. Isolation and localization of unknown faults using neural network-based residuals. *arXiv preprint arXiv:1910.05626*, 2019. arXiv:1910.05626.
- 17 Johan de Kleer and Brian C Williams. Reasoning about multiple faults. In *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, pages 132–139, 1986. URL: <http://www.aaai.org/Library/AAAI/1986/aaai86-021.php>.

- 18 Shijin Li, Jun Hao, and Jianbo Yu. Graph autoencoder with causal relationship inference for fault detection and root cause identification in complex industrial process. *Advanced Engineering Informatics*, 72:104408, 2026. doi:10.1016/J.AEI.2026.104408.
- 19 Yingxiang Liu and Behnam Jafarpour. Graph attention network with granger causality map for fault detection and root cause diagnosis. *Computers & Chemical Engineering*, 180:108453, 2024. doi:10.1016/J.COMPHEMENG.2023.108453.
- 20 Aditya P Mathur and Nils Ole Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pages 31–36. IEEE, 2016. doi:10.1109/CYSWATER.2016.7469060.
- 21 Erick Moreno-Centeno and Richard M Karp. The implicit hitting set approach to solve combinatorial optimization problems with an application to multigenome alignment. *Operations Research*, 61(2):453–468, 2013. doi:10.1287/OPRE.1120.1139.
- 22 Raymond Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987. doi:10.1016/0004-3702(87)90062-2.
- 23 Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008. doi:10.1109/TNN.2008.2005605.
- 24 Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily B Fox. Neural granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4267–4279, 2021. doi:10.1109/TPAMI.2021.3065601.
- 25 Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL: <https://openreview.net/forum?id=rJXMpikCZ>.
- 26 Sarah Wiegrefe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 11–20, 2019. doi:10.18653/V1/D19-1002.
- 27 Yannick Wilhelm, Peter Reimann, Wolfgang Gauchel, and Bernhard Mitschang. Overview on hybrid approaches to fault detection and diagnosis: Combining data-driven, physics-based and knowledge-based models. *Procedia Cirp*, 99:278–283, 2021.
- 28 Peng Yan and Qiang Liu. Dynamic reconstruction feature-based graph attention networks for semi-supervised process fault diagnosis. *IFAC-PapersOnLine*, 58(4):198–203, 2024.
- 29 Ziqi Yuan, Qingyun Sun, Haoyi Zhou, Minglai Shao, and Xingcheng Fu. A comprehensive survey on GNN-based anomaly detection: taxonomy, methods, and the role of large language models. *International Journal of Machine Learning and Cybernetics*, pages 1–26, 2025.
- 30 Fengzhen Zhang, Qibing Jin, Dazi Li, Yang Zhang, and Qian Zhu. Physical graph-based spatiotemporal fusion approach for process fault diagnosis. *ACS omega*, 9(8):9486–9502, 2024.
- 31 Weiqi Zhang, Chen Zhang, and Fugee Tsung. GRELEN: Multivariate time series anomaly detection from the perspective of graph relational learning. In *IJCAI*, pages 2390–2397, 2022. doi:10.24963/IJCAI.2022/332.
- 32 Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Multivariate time-series anomaly detection via graph attention network. In *2020 IEEE international conference on data mining (ICDM)*, pages 841–850. IEEE, 2020. doi:10.1109/ICDM50108.2020.00093.
- 33 Yu Zheng, Huan Yee Koh, Ming Jin, Lianhua Chi, Khoa T Phan, Shirui Pan, Yi-Ping Phoebe Chen, and Wei Xiang. Correlation-aware spatial-temporal graph learning for multivariate time-series anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Dissecting NetDoktor's Online Questionnaire with Entropy, Tree Simplification & Logical Abduction

Alexander Perko ✉ 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Emir Mujić ✉ 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Franz Wotawa ✉ 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Abstract

Medical diagnosis and its accuracy are dependent on the available information. Hence, it is in the interest of both the patient and the doctor to provide the necessary information and ask the right questions in an efficient manner. The same can be said about medical questionnaires, which resemble the conversation during a doctor's appointment. NetDoktor's SymptomChecker is a freely accessible, expert-curated questionnaire in the German language. It can be traversed like a large decision tree, linking symptoms to diseases. However, does it find an optimal conversation path to acquire the necessary information from its users? Here, our study surfaces structural deficiencies and ways to optimise the questionnaire paths. In particular, we model the process of acquiring information and refining prognoses as an abductive reasoning problem and provide a corresponding algorithm. In our experiments, we apply simple tree transformations and optimisations, building upon Shannon's information entropy to expose redundant subtrees and inefficient question order. Our results contribute to a better understanding of medical questionnaires and open the door for future applications of the data provided by NetDoktor. Because SymptomChecker is validated by medical professionals, it is a prime candidate to serve as a reference for benchmarking other systems. For instance, a dataset derived from SymptomChecker can be used for testing systems such as large language models in the task of medical diagnosis.

2012 ACM Subject Classification Information systems → Information extraction; Computing methodologies → Causal reasoning and diagnostics; Computing methodologies → Logic programming and answer set programming; Applied computing → Health care information systems

Keywords and phrases Medical Questionnaire, Diagnosis, Abductive Reasoning, Information Entropy

Digital Object Identifier 10.4230/OASICS.DX.2026.3

Supplementary Material *Software (Replication Package)*: <https://zenodo.org/records/20425820>

Funding The work presented in this paper was partially funded by the European Union under Grant 101159214 – ChatMED. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

1 Introduction

Medical questionnaires (i.e., expert systems with a questionnaire user interface; distinct from questionnaires used in scientific studies) are modelled to resemble conversations a patient might have with a doctor. The goal is an accurate diagnosis, where the accuracy is dependent on the provided information. In turn, the information flow is guided by the questions raised by the questionnaire or doctor. This means that the questions and their order matter to arrive at accurate diagnoses early. This can be described using information theory, and Shannon's information entropy [20] specifically. Entropy is used to describe information gain,



© Alexander Perko, Emir Mujić, and Franz Wotawa;
licensed under Creative Commons License CC-BY 4.0

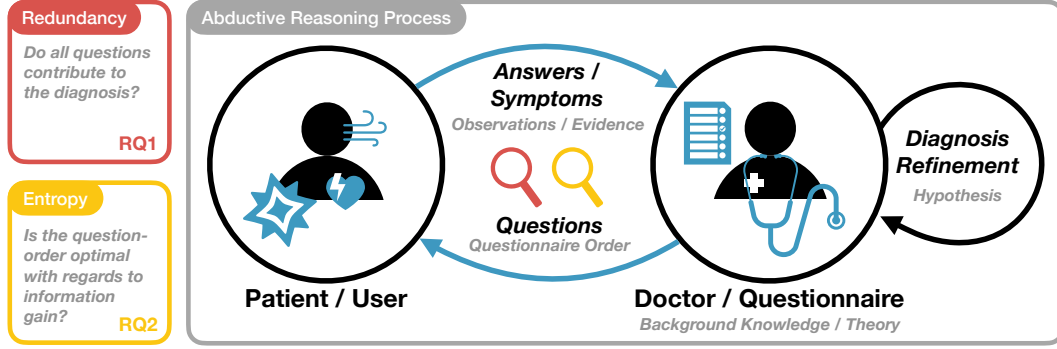
37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 3; pp. 3:1–3:20

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** Doctor's Appointment $\approx$ Medical Questionnaire $\approx$ Abductive Reasoning Process.

measured in bits and given through the formula:

$$H(P) = - \sum_{\Delta} p(\Delta) \log_2 p(\Delta),$$

where P are all paths a conversation with a doctor may take, Δ is the diagnostic outcome a conversation path ends with, and $p(\Delta)$ is the fraction of conversation paths leading to the same outcome. This reading of Entropy yields the possibility to measure how much information is gained when asking a particular question on the conversation path, if paths and possible outcomes are known. Moreover, we can model the retrieval of a diagnosis in the context of a medical questionnaire as an abductive reasoning problem, where abduction was originally coined by Peirce [13] and is nowadays understood as inference to the best explanation.

Figure 1 shows this framing of a medical questionnaire as an abduction problem with the goal of finding a diagnosis (i.e., a hypothesis), which explains symptoms (i.e., observations / evidence) under the consideration of background knowledge (i.e., a theory). If now an observation does not change anything with regards to the hypothesis, we say it is redundant. Again, this ties-in with entropy, as a redundant observation also does not yield any information gain with respect to the hypothesis. Having now prepared our operating room, we can turn to the subject we want to dissect. NetDoktor [19] provides a freely accessible questionnaire, which is validated by medical professionals. It is called SymptomChecker, as it asks users consecutive questions about symptoms and provides possible diagnoses based on the given answers. In our previous work [15, 11], we demonstrated how to traverse SymptomChecker like a large decision tree and used the connection between tree and logical representations to transform the tree into logical formulae [11]. More specifically, we represent all questionnaire paths - from start to diagnosis - as conjunctions of the question-answer pairs, which are interpreted as propositions. The whole tree (i.e., questionnaire) can be represented as a disjunction of all questionnaire paths. In this work the main focus is a concrete subtree of SymptomChecker, where we are interested in its structure, information content and usability for logical abduction, in particular. Our work is guided by the main research question of whether NetDoktor's SymptomChecker is structured in an efficient manner, which we can further separate into the two sub-criteria:

- **RQ1:** Do all questions in SymptomChecker contribute to the outcome (i.e., the diagnoses) or is there redundancy?
- **RQ2:** Is SymptomChecker's question order optimal with respect to the information gained at each step?

Here, we first structurally analyse a dataset aggregated from over 90000 individual SymptomChecker sessions, which is first compiled as part of this work, to find redundant subtrees (RQ1) and sub-optimal question order (RQ2) after which we use the resulting structural variants for a comparative analysis in an abductive reasoning setting. The latter aims to unveil differences when performing reasoning. Both questions are answered based on the subset of SymptomChecker seen in the recorded sessions. That said, this paper contributes by combining and applying a framework of methodologies - all of which are rooted in well-known theories - to investigate and transform a tree-like questionnaire. Furthermore, this work can be seen as a cornerstone for using NetDoktor's data for benchmarking other systems in the future. This paper is organised as follows: We start by tapping into related research. Then, we introduce the subject of our investigation: NetDoktor's SymptomChecker, where we highlight the concrete subtree (i.e., dataset) aggregated through web scraping. After that, we introduce the used methodologies and experimental setup comprising a structural analysis and an abductive reasoning experiment. Figure 5 shows this and contextualises the other figures, tables, and algorithms. Finally, after presenting and discussing the results, we conclude this paper.

2 Related Work

Medical diagnosis can be viewed as a reasoning problem in which observed symptoms, answers, or test results have to be explained by one or more possible diagnoses. This is closely related to abductive reasoning [13], where observations are explained by selecting suitable abducible hypotheses subject to integrity constraints [8]. This can be easily translated into medical questionnaires where patients' answers can be treated as observations and candidate diagnoses can be treated as hypotheses. The issue is that unrestricted abductive reasoning is computationally inefficient [3]. The complexity motivates us to use more structured representations such as decision trees to restrict the search space.

Another way we can view medical diagnosis is using model-based diagnosis [18]. An approach to hypothesis classification using model-based abduction was done by Friedrich *et al.* [4]; they distinguish between different roles that a hypothesis can play as necessary, possible, or irrelevant. This is important because diagnostic reasoning is not only concerned with finding any compatible diagnosis, but also with identifying which hypotheses are actually needed to explain the observations.

Abduction has been applied directly in medicine; Lucas [10] used abduction for checking the quality of medical guidelines. The work showed how abductive reasoning can support the analysis of whether guideline recommendations satisfy broader medical requirements. Pukancová and Homola [16] studied medical diagnosis using description-logic abduction, focusing on diabetes as a use case and arguing that medical diagnosis often requires abductive reasoning as it eliminates diagnostic hypotheses that do not explain all the symptoms. More recently, Xu *et al.* [24] proposed a medical dialogue system that explicitly separates abductive and deductive components, using abduction to generate candidate diagnostic hypotheses and deduction to verify them against the dialogue context. Unlike classical logical abduction/deduction, they leverage black-box models to achieve the results. These works show that abduction remains relevant for modern medical decision-support systems.

Decision trees have been widely used as transparent classifiers that recursively select informative attributes and produce human-readable decision paths [17]. In a medical questionnaire, such paths naturally correspond to sequences of questions and answers, which, in turn, resemble a conversation with a doctor. From a more formal perspective, tree

■ **Table 1** Exemplary SCQ Questions. German Original (DE). English Translation (EN).

ID	Question (DE)	Question (EN)
1	Um wen geht es?	Who is this about?
2	Bist du männlich oder weiblich?	Are you male or female?
3	Zu welcher Altersgruppe gehörst du?	Which age group do you belong to?
4	In welcher Körperregion hast du Beschwerden?	In which body region do you have complaints?
5	Wo treten die Beschwerden auf?	Where do the complaints occur?
6	Welches Symptom belastet dich am stärksten?	Which symptom troubles you the most?
7	Ist deine Haut stellenweise oder überall gerötet?	Is your skin reddened in places or all over?
12	Hast du eine nässende, wund Stelle an der Haut?	Do you have a weeping, sore spot on the skin?
13	Hattest du schon einmal eine Allergie?	Have you ever had an allergy?
14	Spürst du irgendwo ein Brennen?	Do you feel a burning sensation anywhere?
17	Fühlt sich ein Körperteil besonders warm an?	Does a body part feel particularly warm?
32	Hast du offene Verletzungen an der Schleimhaut?	Do you have open lesions on a mucous membrane?
33	Hattest du ungeschützten Sex oder häufig wechselnde Partner?	Have you had unprotected sex or frequently changing partners?

automata provide a mathematical framework for reasoning about tree-shaped structures and their accepted paths; Comon *et al.* [2] describe the correspondence between regular tree languages, finite tree automata, and logical characterisations of trees. Tree-based methods have also been used in clinical decision support and questionnaire reduction.

Hannöver *et al.* [7] applied classification trees to data from the German multicenter study on eating disorders, showing how tree models can support clinical decision-making in an interpretable way. Strobl *et al.* [21] provide an especially relevant medical example: using data from the German DizzyReg patient registry, they identified a small set of eight key questions that helped classify common vestibular disorders, to be used as triage before conducting a clinical examination. This work is close in spirit to questionnaire-based diagnosis, since it demonstrates how a larger clinical history can be reduced to a compact and interpretable set of discriminative questions.

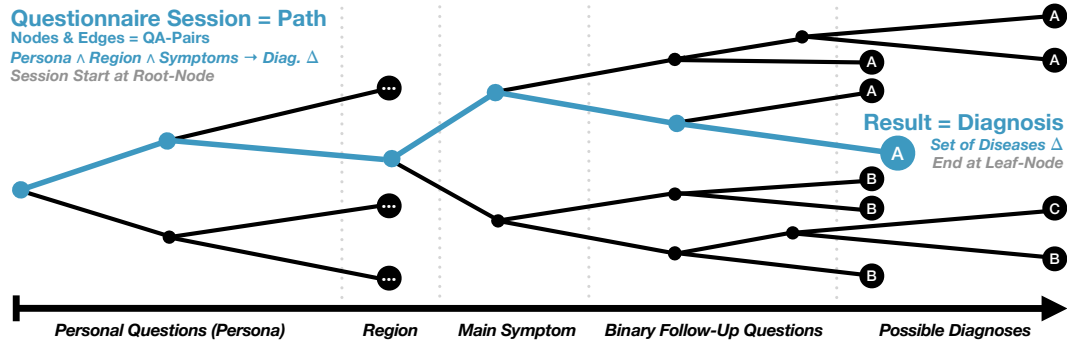
3 NetDoktor’s SymptomChecker Questionnaire

NetDoktor SymptomChecker [19] is a freely accessible¹, medical expert system with a questionnaire UI that retrieves likely diagnoses from a person’s reported symptoms. SymptomChecker itself is based on a proprietary, expert-curated database called AMBOSS [1]. The questionnaire is available in German and combines static questions about a person such as sex, age (referred to as “persona” in our work) and the affected body region and main symptom, with dynamically selected follow-up questions. The dynamic part consists of closed-ended questions with possible answers “Yes”, “No”, and “Skip”. We do not use “Skip” in our experiments and only consider positive and negative answers, which is why we refer to these questions as binary. We treat SymptomChecker as a deterministic reference model and traverse the questionnaire systematically, storing the resulting interactions as a tree, which we explained in our previous work [15, 11] and is illustrated in Figure 2.

¹ SymptomChecker Questionnaire: www.netdoktor.at/symptom-checker

3.1 Questionnaire Traversal & Tree Representation

The extraction procedure consists of two main steps: ID extraction and the questionnaire traversal. Since the questionnaire is meant for humans, the questions are written in natural language. It would be inefficient to save all question-answer (QA) pairs in such a format, so we set an ID integer value for each unique question and each unique answer. Then each question and answer is saved as a QA-pair. For example, a QA-pair [13, 7] represents a decision node in a tree where a question “Hattest du schon einmal eine Allergie?” (Have you ever had an allergy?) is answered by “Yes”. A QA-pair [13, 8] represents a negative answer to the same question. We follow a depth-first search to extract the tree structure. This way, the questionnaire is unfolded into a finite decision tree T where each internal node corresponds to a question, and each edge corresponds to an answer. A complete root-to-leaf path p therefore represents one possible questionnaire session and can be interpreted as a conjunction of extracted predicates, each corresponding to a QA-pair. The sets of diseases at the leaf are the diagnostic conclusions associated with this conjunction. We denote a set of potential diseases as diagnosis Δ . In turn, all distinct diseases in the whole tree are referred to by $\mathcal{D}$. Furthermore, we denote a collection of paths (i.e., a disjunction of conjunctions) as $p \in P$ and assume that a tree constructed in the way we described (or a subtree thereof) can be represented as such a collection.



■ **Figure 2** Tree Representation of NetDoktor's SymptomChecker Questionnaire.

3.2 SCQ Dataset

In this subsection, we describe the dataset, which we refer to as SCQ, for SymptomChecker Questionnaire, which we derived by traversing SymptomChecker. It forms the basis for all of our experiments. In particular, the dataset we extract is a subset of the complete questionnaire tree. For our extraction, we balanced the search between depth and breadth of the constructed trees. This was done since there is no public information on the true size of SCQ, or its algorithm, and the search doesn't spend too much time only looking into a single set of symptoms, but explores other branches as well. Furthermore, for simplicity, we focus on a single persona description, an adult male, and on the sub-region skin (German: “Haut” and “Kopfhaut”). We selected skin, as this sub-region appears in all but two of the main regions of the full tree. The first row of Table 2 shows the statistics for the SCQ dataset used in our experiments. From the table, we can see that the total number of complete paths extracted from SymptomChecker is 95,318. A path is a single possible sequence of answered questions from the beginning of the questionnaire to the terminal nodes (diagnosis sets Δ). The total number of diseases $|\mathcal{D}|$ is 129. There are 635 unique diagnoses (i.e., sets of

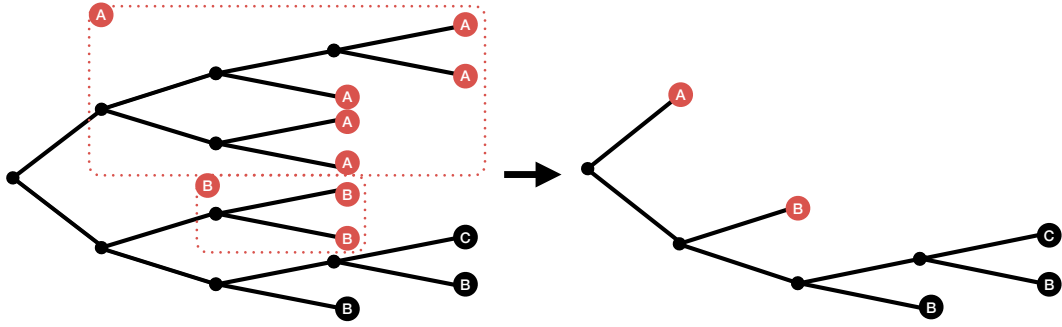
diseases). The average depth per questionnaire path is 20.4, meaning that it takes, on average, 20.4 questions to obtain a set of diseases. The shortest path only has 6 question-answer pairs while the longest has 24.

4 Methodology

In this section, we introduce the theoretical basis of our experiments, which comprises naïve tree simplification, entropy-based tree reconstruction, and Horn Clause Abduction Problems.

4.1 Naïve Tree Simplification

In this section, we describe a naïve tree simplification method, which we will use to identify and remove redundant subtrees. In our context, this translates to finding QA pairs which do not change the outcome and can be removed from the questionnaire. The basic idea of the algorithm is shown in Figure 3: On the left-hand side is the binary input tree, and on the right-hand side the simplified output tree. In both trees, the same diagnoses (leaves) are reachable. The output tree, however, has the same or a smaller amount of paths leading there. We define the following recursive algorithm: For each binary, follow-up subtree, preceded by

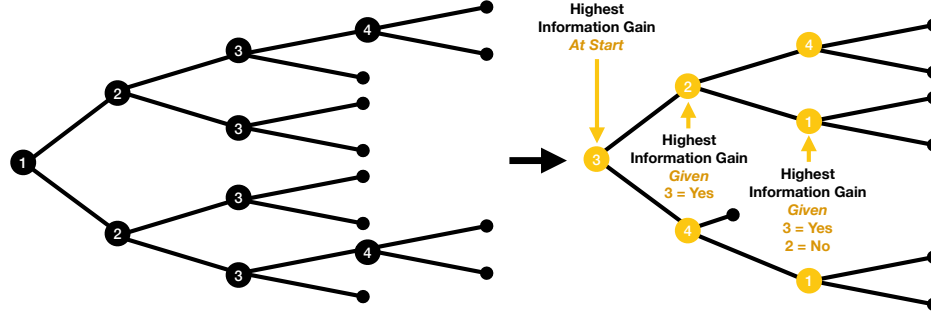


■ **Figure 3** Tree Simplification: Identification & Removal of Redundant Questions.

a main symptom node, we walk the paths from leaf upwards. At each QA-node, we compare the two children and collapse them, i.e., replace the current QA-node with the child-nodes' diagnoses, iff both child-nodes are leaves and carry the same diagnoses. We repeat this collapsing procedure until there are no QA-nodes satisfying the criterion.

4.2 Entropy-Based Tree (Re-)Construction

Following our previous work, where we introduced an entropy-based algorithm and demonstrated it within a chatbot-powered questionnaire [12, 22], we employ a similar strategy here. The core idea of the algorithm follows information-gain-based decision tree induction [17]: at each step, the algorithm selects the eligible question that produces the greatest reduction in entropy over the possible answers. However, the previously introduced algorithm was built with user interaction in mind and modelled as a constraint satisfaction problem. While the core principle remains the same for this implementation, we apply the algorithm without user interaction and to reorder an existing diagnostic tree. Figure 4 shows an exemplary input tree on the left-hand side and a reconstructed output tree on the right-hand side. The questions in the output tree are always ordered to maximise the information gain by selecting the question to come next with the highest discriminative power given the previous



■ **Figure 4** Entropy-Based Tree (Re-)Construction / Question Reordering.

QA-nodes. We define a recursive algorithm: For each binary, follow-up subtree, preceded by a main symptom node, per path we collect the set of diseases it ends in and the QA-pairs (i.e., questions, which were answered in this path and the answer), which we write as a pair (QA, Δ) . Then we start recursively constructing a new tree from the subtree-root by calling the function `EntropyTree`, which is introduced in Algorithm 1, and first called with the set of paths of a binary follow-up subtree P and an empty set of used questions U . We start by determining question eligibility in Line 2, where a question q is considered if it is on every path in P , i.e., every $(QA, \Delta) \in P$ contains a pair (q, a) , and is not part of the set of questions, which have been used already U . Eligible questions are collected in E , where we abort and return one terminal node $\text{Leaf}(\Delta)$ per distinct diagnosis in P , i.e., one per $\Delta \in \text{Diagnoses}(P) = \{\Delta \mid (QA, \Delta) \in P\}$, when E stays empty (i.e., when there are no eligible questions) in Lines 3-4. Then, H is used to compute the information gain IG for all eligible questions, which can be seen in Lines 5-6, where $P_a^q = \{(QA, \Delta) \in P \mid (q, a) \in QA\}$ refers to all paths in P whose answer to q is a , and where the sum runs over $\text{Answers}(q, P)$, the answers to q observed in P . H in this context refers to the information entropy after Shannon [20]. Specifically, we use it as $H(P) = -\sum_{\Delta \in \text{Diagnoses}(P)} p(\Delta) \log_2 p(\Delta)$, where $p(\Delta)$ is the fraction of paths in P ending in diagnosis Δ . This information is used to select the question with the highest information gain q^* , in Line 7. We stop and return one leaf per distinct diagnosis in P , if there is no information to be gained, i.e., even the highest information gain is (close to) zero $IG(q^*) \leq \varepsilon \approx 0$, in Lines 8-9. Finally, in Lines 10-13, we reconstruct the tree by attaching the result of a recursive call of `EntropyTree` below a branch node $\text{Branch}(q^*, a, S)$, which carries the question-answer-pair (q^*, a) and the recursively constructed subtree S , for every answer a (i.e., branch) to question q^* that is observed in P . The recursive call per branch is invoked using the subset of paths $P_a^{q^*}$ to which the answer to q^* is the branching answer a , and with U updated by adding q^* . Ultimately, the length of the new, reconstructed paths is guided by the availability of eligible questions and their information gain, while it is ensured that all diagnoses / diseases are preserved.

4.3 Horn Clause Abduction Problems

In this paper, and this Subsection in particular, the definitions are based on those of Propositional Horn Clause Abduction Problems (PHCAP) by Friedrich et al. [4], which we adapt for our application domain and modelling using the Answer Set Programming (ASP) paradigm [6, 9].

► **Definition 1.** $HCAP = \langle \text{Var}, \text{Hyp}, \text{SD}, \text{Obs} \rangle$. *Horn Clause Abduction Problem.*

Algorithm 1 Entropy-Based Tree Re-Construction.

Require: Paths P of one binary follow-up subtree; used questions U . Initial call per subtree: $\text{ENTROPYTREE}(P, \emptyset)$

Ensure: The reconstructed subtree below the current QA-pair (below the main symptom node on the initial call)

```

1: function  $\text{ENTROPYTREE}(P, U)$ 
  // Eligible Questions: Answered by Every Path in  $P$ , Not Yet Used
2:    $E \leftarrow \{ q \mid \text{every } (QA, \Delta) \in P \text{ contains a pair } (q, a) \} \setminus U$ 
3:   if  $E = \emptyset$  then
4:     return  $\{ \text{Leaf}(\Delta) \mid \Delta \in \text{Diagnoses}(P) \}$ 
  // Score Every Eligible Question by Information Gain
5:   for all  $q \in E$  do
6:      $\text{IG}(q) \leftarrow H(P) - \sum_{a \in \text{Answers}(q, P)} \frac{|P_a^q|}{|P|} H(P_a^q)$ 
  // Select the Most Informative Question; Stop If None Discriminates
7:    $q^* \leftarrow \arg \max_{q \in E} \text{IG}(q)$ 
8:   if  $\text{IG}(q^*) \leq \varepsilon$  then
9:     return  $\{ \text{Leaf}(\Delta) \mid \Delta \in \text{Diagnoses}(P) \}$ 
  // Construct Subtree by Recursing Once per Answer of  $q^*$  Observed in  $P$ 
10:   $\text{nodes} \leftarrow \emptyset$ 
11:  for all  $a \in \text{Answers}(q^*, P)$  do
12:     $\text{nodes} \leftarrow \text{nodes} \cup \{ \text{Branch}(q^*, a, \text{ENTROPYTREE}(P_a^{q^*}, U \cup \{q^*\})) \}$ 
13:  return  $\text{nodes}$ 

```

► **Definition 2.** *Var.* All possible questions (1...244), answers (1...25) and diseases (1...129) on the paths of SCQ.

► **Definition 3.** *Hyp* $\subset$ *Var.* The abducible hypotheses. In SCQ, *Hyp* = $\mathcal{D}$.

► **Definition 4.** *SD.* The System Description encoded via Horn Clauses (ASP rules).

► **Definition 5.** *Obs* $\subset$ *Var.* Observations, i.e., QA-pairs (ASP facts).

4.3.1 From Questionnaire Tree to Horn Clauses

In this section, we introduce Algorithm 2 to obtain a system description SD from a questionnaire tree T . In particular, we use this algorithm to derive ASP [6, 9] models to be used with Clingo [5] as a solver. It has to be stated that this work does not aim at nor claim to derive a perfect SD, but to use abductive reasoning as a mere vehicle to investigate SCQ. Hence, the main goal of this algorithm is the straightforward, automatic extraction of an SD given a tree in the format of SCQ. That said, Algorithm 2 produces rules (i.e., Horn Clauses) of two types, which are based on the polarity of the QA pair and form subsets of SD: SD^+ and SD^- . Together with a predefined set of base rules SD_{base} (see Subsection 5.1.2), the two subsets of polarity-based rule types are unified to form the full SD:

$$\text{SD} = \text{SD}_{\text{base}} \cup \text{SD}^+ \cup \text{SD}^-$$

In order to have some control over the added rules, we introduce confidence thresholds τ^+ and τ^- for the two polarity-specific subsets, respectively. In tandem with SD_{base} as used in our experiments, rules in SD^+ enable diagnoses to cover observations and those in SD^-

eliminate a diagnosis, if an observation is “Ja” for the same symptom. Looking now at Algorithm 2, we see in Lines 1-6 the initialisation of the co-occurrence matrices CO_{cat} , for categorical questions, i.e., questions 1-6 (see Table 1), and for all binary, follow-up questions, i.e., CO_{bin} , CO_{bin}^+ , CO_{bin}^- as well as the total counter. Next, in Lines 7-18, the co-occurrence matrices are filled by iterating over the questionnaire paths and increasing the corresponding co-occurrence counter for every pair of diagnosis and question. Finally, the SD subsets are aggregated in Lines 19-29 based on the confidence thresholds and at the end unified with SD_{base} .

■ **Algorithm 2** Extract System Description SD from Questionnaire Tree T .

Require: Questionnaire Tree T ; Base Rules SD_{base} ; Confidence Thresholds τ^+ , τ^-

Ensure: $SD = SD_{base} \cup SD^+ \cup SD^-$

```

1: function EXTRACTSYSTEMDESCRIPTION( $T, SD_{base}, \tau^+, \tau^-$ )
  // Init Co-Occurrence Matrices for Diagnoses and Questions
2:   for all  $\Delta \in \text{Diagnoses}(T)$  do
3:      $total[\Delta] \leftarrow 0$ 
4:     for all  $s \in \text{Atoms}_{cat}(T)$  do
5:        $CO_{cat}[\Delta][s] \leftarrow 0$ 
6:     for all  $q \in \text{Questions}_{bin}(T)$  do
7:        $CO_{bin}[\Delta][q] \leftarrow 0$ ;  $CO_{bin}^+[\Delta][q] \leftarrow 0$ ;  $CO_{bin}^-[\Delta][q] \leftarrow 0$ 
  // Fill Co-Occurrence Matrices by Iterating over Questionnaire Paths
8:   for all  $p \in \text{Paths}(T)$  do
9:      $S_{cat} \leftarrow \{ (q, a) \mid (q, a) \in \text{Region}(p) \cup \text{MainSymptom}(p) \}$ 
10:    for all  $\Delta \in \text{Leaves}(p)$  do
11:       $total[\Delta] \leftarrow total[\Delta] + 1$ 
12:      for all  $s \in S_{cat}$  do
13:         $CO_{cat}[\Delta][s] \leftarrow CO_{cat}[\Delta][s] + 1$ 
14:      for all  $(q, \text{polarity}) \in \text{FollowUp}(p)$  do
15:         $CO_{bin}[\Delta][q] \leftarrow CO_{bin}[\Delta][q] + 1$ 
16:        if polarity is positive ("Ja") then
17:           $CO_{bin}^+[\Delta][q] \leftarrow CO_{bin}^+[\Delta][q] + 1$ 
18:        if polarity is negative ("Nein") then
19:           $CO_{bin}^-[\Delta][q] \leftarrow CO_{bin}^-[\Delta][q] + 1$ 
  // Construct Horn Clauses using Confidence Thresholds
20:  for all  $\Delta, s$  do
21:    if  $CO_{cat}[\Delta][s] / total[\Delta] \geq \tau^+$  then
22:       $SD^+ \leftarrow SD^+ \cup \{ s \leftarrow \Delta \}$ 
23:  for all  $\Delta, q$  with  $CO_{bin}[\Delta][q] > 0$  do
24:     $c^+ \leftarrow CO_{bin}^+[\Delta][q] / CO_{bin}[\Delta][q]$ ;  $c^- \leftarrow CO_{bin}^-[\Delta][q] / CO_{bin}[\Delta][q]$ 
25:    if  $c^+ \geq \tau^+$  then
26:       $SD^+ \leftarrow SD^+ \cup \{ s^+(q) \leftarrow \Delta \}$ 
27:    if  $c^- \geq \tau^-$  then
28:       $SD^- \leftarrow SD^- \cup \{ s^-(q) \leftarrow \Delta \}$ 
29:  return  $SD_{base} \cup SD^+ \cup SD^-$ 

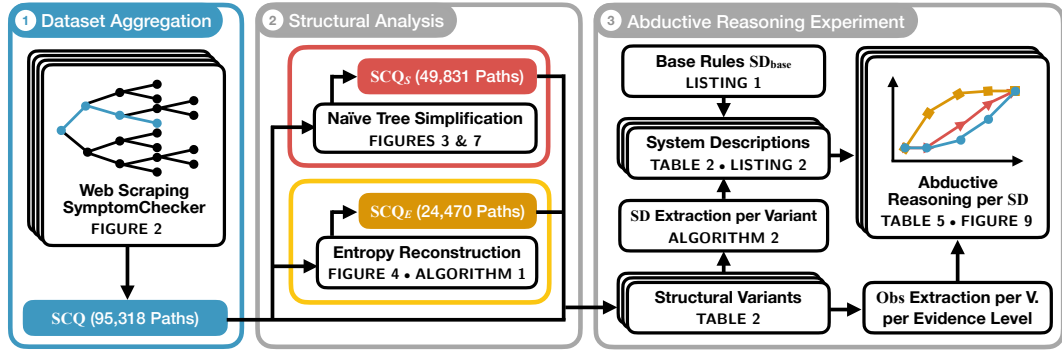
```


5 Experimental Evaluation

In this section, we describe our experimental setup, which is modelled around the information gained during a doctor’s appointment, or rather a questionnaire session. Furthermore, we introduce structural variations of SCQ and methods to produce diagnoses from the questionnaire paths. Finally, we present the experimental results and discuss them.

5.1 Experimental Setup

The core idea of our experimental setup is to simulate an SCQ session via traversals of the SCQ tree and structural variants thereof as an abductive reasoning problem. We obtain observations and system descriptions from SCQ and its variants. Then, we evaluate the reasoning performance at evidence levels, which symbolise the acquisition of information throughout the questionnaire session. This experimental setup is coupled with a statistical analysis of the SCQ dataset with a focus on prominence of diagnoses, information gain (as measured by entropy), and redundancy (as detected by tree simplification).



■ **Figure 5** Experimental Setup & Overview of Figures, Tables, Algorithms, and Listings.

5.1.1 Structural Variants of SCQ

In our abductive reasoning experiments, the original SCQ tree serves as a reference. In addition, we use two structural variants of SCQ, which are in-line with our research questions. First, we apply naïve tree simplification, as described in Subsection 4.1, to explore potential redundancies. Here we aim to answer **RQ1**, as a tree with redundant question nodes would demonstrate that these questions are not contributing to the final diagnosis. Second, we use an entropy-based tree construction, as described in Subsection 4.2. The aim of this is to answer **RQ2**, as a comparison between the original and the entropy-optimised tree should have equal (or similar) results with regards to the information gained throughout the steps of the abductive reasoning process. In summary, we use the following three structural variants in our experiments: **SCQ**, **SCQ_S**, and **SCQ_E**, which are listed and explained further in Table 2. Furthermore, we use a balanced subsample of SCQ used for hyperparameter search and to ground the comparison. We call this variation **SCQ'** and balance it by limiting the number of paths per set of diseases to 8 and randomly select paths when there are more than 8 for a diagnosis. This way, dominant diagnoses do not skew the dataset as much.

■ **Table 2** Statistics of Structural Variants of SCQ and Their Derived System Descriptions. Number of Individual Diseases (129) and Number of Distinct Sets of Diagnoses (635) are Equal Among All Variants. We use the following shorthands: Number of Paths (#P), Average (Avg).

Structural Variant		#P	Path Depth		Derived SD Rules		
			Range	Avg	SD ⁺	SD ⁻	Σ
SCQ	Original questionnaire tree.	95 318	6/24	20.4	6 322	3 132	9 454
SCQ _S	Simplified/pruned version of SCQ.	49 831	6/24	19.7	6 083	3 044	9 127
SCQ _E	Entropy-reconstruction of SCQ.	24 474	6/23	19.1	5 955	3 085	9 040
SCQ'	Balanced subset of SCQ.	4 165	6/23	19.7	5 552	2 870	8 422

5.1.2 System Description

SD is obtained using Algorithm 2, which we introduced in Subsection 4.3. In this subsection, we list the input parameters used for our abductive reasoning experiment. Listing 1 shows the base rules SD_{base} as used for our experiments. In this case, the base rules comprise a rule introducing *Hyp*, a cardinality-based minimisation criterion, and the following integrity constraints:

- C1: Every positive observation must be explained by some hypothesised diagnosis.
- C2: A positive observation (“Ja”) contradicts a negative.
- C3: A negative observation (“Nein”) penalises a positive-predicting diagnosis.

Guided by the results of a hyperparameter search (see Subsection 5.1.4) to find a good balance between answers in SCQ and rules in the corresponding SD, we assume the following confidence thresholds for the polarity-based rules: $\tau^+ = 0$ and $\tau^- = 0.8$. Using consistent SD_{base} and confidence thresholds, we use $SD(SCQ)$, $SD(SCQ_S)$, and $SD(SCQ_E)$ when referring to a complete SD derived from SCQ, SCQ_S , and SCQ_E , respectively. Listing 2 shows an excerpt of the polarity-based rules derived from SCQ, i.e., $SD^+ \cup SD^- \subset SD(SCQ)$.

■ **Listing 1** Predefined Subset of Base Rules SD_{base} of SD.

```
possible_diagnosis(1..129).           % Hyp: Abducible Diagnoses
{ diagnosis(D) } :- possible_diagnosis(D).
:- obs_ja(Q, A), not sym_ja(Q, A).     % Integrity Constraint C1
:- obs_ja(Q, 7), sym_nein(Q).          % Integrity Constraint C2
:- obs_nein(Q), sym_ja(Q, 7). [1@2, Q] % Integrity Constraint C3
:- diagnosis(D). [1@1, D]              % Minimise Cardinality of Hyp
```

■ **Listing 2** United Subsets $SD^+ \cup SD^-$ of SD Extracted from SCQ Using Algorithm 2.

```
sym_ja(5, 5) :- diagnosis(1). % SD+ Rule 1: conf_ja=1.00 [Fusspilz]
sym_ja(6, 6) :- diagnosis(1). % SD+ Rule 2: conf_ja=0.54 [Fusspilz]
sym_ja(7, 7) :- diagnosis(1). % SD+ Rule 3: conf_ja=0.71 [Fusspilz]
...                          % All SD+ Rules: 6322 (conf_ja >= 0)
sym_nein(17) :- diagnosis(1). % SD- Rule 1: conf_nein=0.98 [Fusspilz]
sym_nein(32) :- diagnosis(1). % SD- Rule 2: conf_nein=1.00 [Fusspilz]
sym_nein(33) :- diagnosis(1). % SD- Rule 3: conf_nein=1.00 [Fusspilz]
...                          % All SD- Rules: 3132 (conf_nein >= 0.8)
```

5.1.3 Observations & Evidence Levels

As pointed out above, SCQ structurally resembles a doctor’s appointment. At first, only the patient’s information according to the intake form is known, which translates to the persona in SCQ. Then, at the beginning of the examination, the general region is located, which is followed by the description of the main symptom. Afterwards, further information is acquired through follow-up questions by the doctor, which is concluded by giving a final (hopefully accurate) diagnosis. We translated this into the 5 distinct evidence levels L0-L4, at which observations are added consecutively and abduction is evaluated. L0-L4 are tightly bound to the structure of the questionnaire sessions, which is why we introduce additional percentage-based evidence levels P0-P100 for reference. Both are further described in Table 3.

■ **Table 3** Discrete evidence levels at which observations are drawn and abduction is evaluated.

Based on Doctor’s App. / Questionnaire Session		Percentage-Based	
L0	Persona information only.	P0	No information.
L1	Persona and region.	P25	25% of information.
L2	Persona, region and main symptom.	P50	50% of information.
L3	Persona, region, main symptom and 50% of follow-up.	P75	75% of information.
L4	All the information.	P100	All the information.

5.1.4 Hyperparameter Search

We performed a hyperparameter search to find values for the confidence thresholds τ^+ and τ^- controlling the polarity based SD^+ and SD^- . The whole setup is shown in Table 4. Each cell in the table corresponds to one combination of τ^+ and τ^- , where τ^+ controls positive rules, and τ^- negative ones. For every combination, we generated an SD from the balanced SCQ’ and computed the F1-score at evidence level P100. Notably, we included thresholds excluding all negative rules ($\tau^- = \infty$), as well as keeping all ($\tau^- = 0$). Conversely, for positive rules we did not include a setting excluding all of them, as this would yield a degenerate SD. Overall, keeping all positive rules ($\tau^+ = 0$) and a τ^- of 0.8 performed best.

■ **Table 4** Hyperparameter Search for Confidence Thresholds τ^- (Columns) and τ^+ (Rows): Grid search using SCQ’ and maximising F_1 at evidence level P100. Best value in **bold**.

τ^+	τ^-												
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.95	0.99	∞
0	0.002	0.008	0.012	0.018	0.031	0.060	0.134	0.220	0.294	0.246	0.190	0.145	0.052
0.05	0.000	0.006	0.010	0.018	0.031	0.060	0.129	0.212	0.274	0.245	0.215	0.195	0.120
0.1	0.000	0.006	0.010	0.018	0.032	0.060	0.133	0.219	0.293	0.274	0.257	0.235	0.139
0.2	0.000	0.005	0.007	0.014	0.024	0.046	0.095	0.160	0.213	0.220	0.216	0.202	0.127
0.3	0.000	0.004	0.007	0.013	0.020	0.036	0.080	0.133	0.158	0.181	0.185	0.186	0.127
0.5	0.001	0.005	0.006	0.012	0.019	0.034	0.051	0.065	0.082	0.095	0.099	0.102	0.140

5.1.5 Evaluation

Following a structural analysis of SCQ, we will evaluate the questionnaire sessions as an abductive reasoning process using the standard metrics Precision (P), Recall (R), and F1.

5.2 Results & Discussion

In this section, we show our experimental results and provide our interpretation. We provide a replication package including SCQ, its structural variants, as well as the corresponding SDs and code to simultaneously draw Obs from the SCQ variants and execute Clingo using our settings. We use Clingo 5.8.0 in brave mode.

5.2.1 Disease Prominence & Prior Baseline

We start discussing our results by highlighting the statistical imbalance found in the SCQ dataset. In particular, we find that there are some diseases more prominent than others to a high degree. Figure 6 shows this by comparing the most prominent with the least prominent

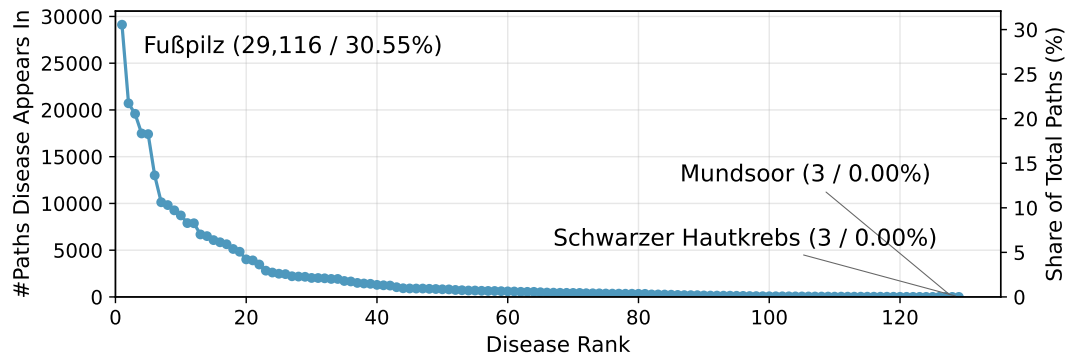
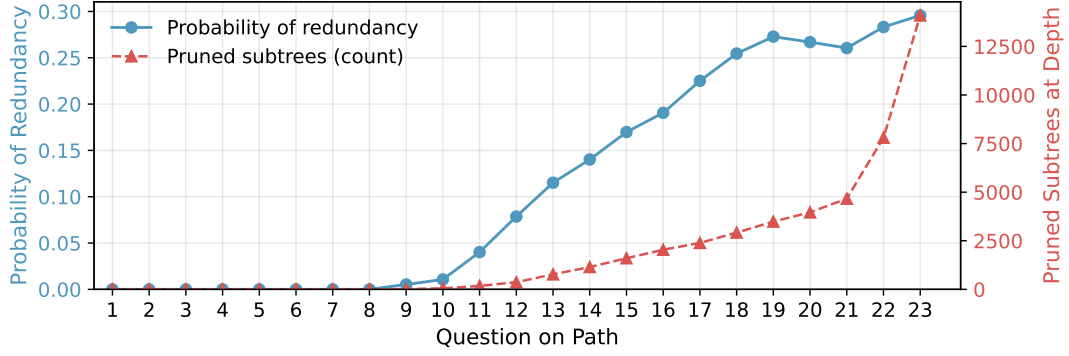


Figure 6 Disease Prominence in SCQ from Most Prominent (i.e., Rank 1 = Fußpilz/Athlete’s Foot) to Least Prominent (i.e., Rank 129 = Schwarzer Hautkrebs/Melanoma & Mundsoor/Oral Thrush).

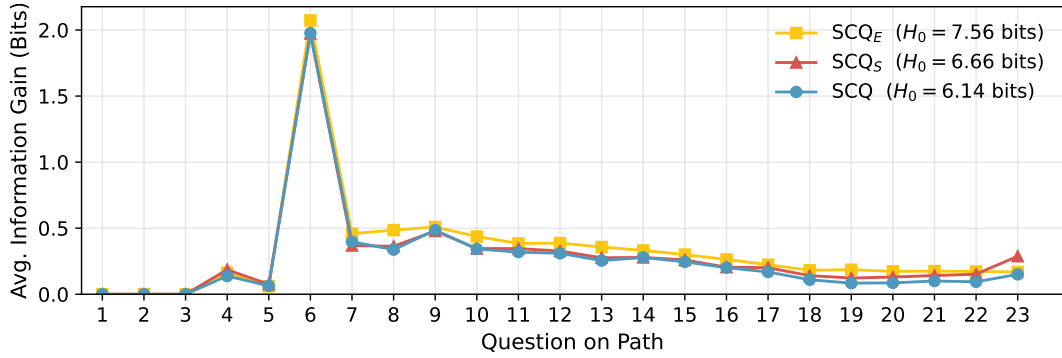
disease based on the number of paths they appear in. For instance, the most prominent disease “Athlete’s foot” appears in 29116/95318 paths, which is over 30% of SCQ. As a stark contrast, the least prominent diseases, “Melanoma” and “Oral thrush”, each appear in only 3/95318 paths. Overall one can easily spot the sharp peak of a few, very prominent diseases, followed by the “long tail” of the many diseases appearing in few paths in Figure 6. This, of course, leads to a skewed prior if we consider it as the relative frequency of any individual disease. The other approach for selecting a prior is using $\frac{1}{|\mathcal{D}|}$ for each disease. We include both the skewed prior and $\frac{1}{|\mathcal{D}|}$ as baselines in the experiments below, where we refer to them as B1 and B2, respectively.

5.2.2 Redundancy & Information Gain

Now, we want to shift our attention to the main research questions by graphically demonstrating 1) the redundancy present in the original SCQ and 2) the differences in information gain per question between the structural variants of SCQ. The former is visualised in Figure 7 and the latter, in Figure 8. Starting with 1), we can observe in Figure 7 that the probability for a question being redundant in SCQ rises (blue line) after question 8, which is only two questions into the binary follow-up questions. The red line, showing the pruned subtrees, also rises, but more slowly, because it counts the subtrees, not the individual questions. This shows that SCQ is not free of redundancy to begin with and that there exist indeed entire redundant subtrees (as the probability of individual questions being redundant rises faster than the count of pruned subtrees). Turning to 2), we can read from Figure 8 that there is



■ **Figure 7** Probability of Redundancy by Question as Encountered in SCQ (Left Y-Axis; Blue). Number of Pruned subtrees during Simplification of SCQ to Generate SCQ_S (Right Y-Axis; Red).



■ **Figure 8** Average Information Gain by Question per SCQ Variant.

no information gain at the beginning of the questionnaire. This is not surprising, as we are looking at a subtree with fixed persona, which translates to the first three questions having no discriminative power at all within SCQ. Compare to Table 1, where the first 6 (among others) questions are listed. Question 4, which asks the user for the main body region the symptom occurs in, shows a first local maximum. This is explainable, as we did not fix the main region when acquiring SCQ. Question 5, on the other hand, was fixed to the two skin-related sub-regions, which shows as a drop in information gain in the plot. After this question, we encounter the global maximum at question 6, which asks for the main symptom. After this point, the binary questions related to other symptoms follow. Here, we can observe two main differences across the structural variants of SCQ: On the one hand, we see that SCQ_S gets ahead of the original SCQ after question 16, which is a sign of pruning redundant subtrees actually contributing to information gain. On the other hand, we find that SCQ_E is continuously ahead of the other two (except for the last one, where SCQ_S overtakes it). This, of course, comes as no surprise given the construction of SCQ_E . It must be noted, however, that neither of these differences is major.

5.2.3 Abductive Reasoning

Having analysed the statistics and gained insight into each of the structural variants of SCQ, we can finally discuss the derived SD and Obs, which serve as simulated questionnaire sessions. The main questions to ask are: 1) Do the derived SDs differ? And: 2) How do they cope with Obs drawn from any of the structural variants? To answer 1), we read from Table 2 that the three SDs are similarly sized, even though their origins are vastly different as measured by their number of paths. For instance, SCQ has 95318 paths and the corresponding SD,

■ **Table 5** Precision (P), Recall (R), and F1 per System Description (SD), Observation (Obs), and Evidence Level (L0-L4 & P0-P100). Baselines are constant across levels and shown next to each SD header where: B1 = Skewed Prior, B2 = $1/|\mathcal{D}|$.

	Obs	L0	L1	L2	L3	L4	P0	P25	P50	P75	P100
SD(SCQ) · B1 (P/R/F1) = 0.224/0.221/0.222 · B2 (P/R/F1) = 0.024/0.023/0.023											
P	SCQ	0.000	0.031	0.103	0.385	0.493	0.000	0.105	0.317	0.497	0.493
	SCQ'	0.000	0.034	0.108	0.135	0.313	0.000	0.095	0.117	0.176	0.313
	SCQ _S	0.000	0.032	0.106	0.316	0.427	0.000	0.103	0.254	0.405	0.427
	SCQ _E	0.000	0.029	0.092	0.229	0.395	0.000	0.082	0.181	0.291	0.395
R	SCQ	0.000	1.000	1.000	0.636	0.519	0.000	0.770	0.643	0.700	0.519
	SCQ'	0.000	1.000	1.000	0.297	0.267	0.000	0.731	0.344	0.299	0.267
	SCQ _S	0.000	1.000	1.000	0.614	0.455	0.000	0.812	0.627	0.650	0.455
	SCQ _E	0.000	1.000	1.000	0.454	0.407	0.000	0.745	0.473	0.478	0.407
F1	SCQ	0.000	0.058	0.184	0.455	0.475	0.000	0.178	0.396	0.561	0.475
	SCQ'	0.000	0.065	0.190	0.171	0.265	0.000	0.160	0.159	0.204	0.265
	SCQ _S	0.000	0.060	0.189	0.396	0.408	0.000	0.176	0.341	0.479	0.408
	SCQ _E	0.000	0.055	0.167	0.287	0.370	0.000	0.144	0.242	0.343	0.370
SD(SCQ_S) · B1 (P/R/F1) = 0.224/0.221/0.222 · B2 (P/R/F1) = 0.024/0.023/0.023											
P	SCQ	0.000	0.031	0.103	0.362	0.586	0.000	0.103	0.296	0.472	0.586
	SCQ'	0.000	0.034	0.108	0.131	0.349	0.000	0.095	0.114	0.175	0.349
	SCQ _S	0.000	0.032	0.106	0.306	0.544	0.000	0.103	0.245	0.394	0.544
	SCQ _E	0.000	0.029	0.092	0.214	0.468	0.000	0.082	0.169	0.287	0.468
R	SCQ	0.000	1.000	1.000	0.625	0.534	0.000	0.763	0.630	0.682	0.534
	SCQ'	0.000	1.000	1.000	0.285	0.274	0.000	0.730	0.337	0.281	0.274
	SCQ _S	0.000	1.000	1.000	0.609	0.507	0.000	0.810	0.620	0.653	0.507
	SCQ _E	0.000	1.000	1.000	0.443	0.441	0.000	0.743	0.463	0.480	0.441
F1	SCQ	0.000	0.058	0.184	0.437	0.514	0.000	0.176	0.379	0.536	0.514
	SCQ'	0.000	0.065	0.190	0.164	0.285	0.000	0.160	0.154	0.196	0.285
	SCQ _S	0.000	0.060	0.189	0.387	0.478	0.000	0.176	0.332	0.472	0.478
	SCQ _E	0.000	0.055	0.167	0.273	0.416	0.000	0.144	0.230	0.341	0.416
SD(SCQ_E) · B1 (P/R/F1) = 0.224/0.221/0.222 · B2 (P/R/F1) = 0.024/0.023/0.023											
P	SCQ	0.000	0.031	0.103	0.398	0.614	0.000	0.106	0.314	0.519	0.614
	SCQ'	0.000	0.034	0.108	0.141	0.365	0.000	0.095	0.121	0.193	0.365
	SCQ _S	0.000	0.032	0.106	0.325	0.569	0.000	0.103	0.250	0.427	0.569
	SCQ _E	0.000	0.029	0.092	0.241	0.506	0.000	0.083	0.189	0.323	0.506
R	SCQ	0.000	1.000	1.000	0.612	0.511	0.000	0.771	0.577	0.680	0.511
	SCQ'	0.000	1.000	1.000	0.282	0.269	0.000	0.729	0.333	0.287	0.269
	SCQ _S	0.000	1.000	1.000	0.581	0.490	0.000	0.812	0.564	0.651	0.490
	SCQ _E	0.000	1.000	1.000	0.476	0.448	0.000	0.748	0.495	0.514	0.448
F1	SCQ	0.000	0.058	0.184	0.455	0.509	0.000	0.180	0.379	0.559	0.509
	SCQ'	0.000	0.065	0.190	0.168	0.286	0.000	0.161	0.159	0.206	0.286
	SCQ _S	0.000	0.060	0.189	0.394	0.478	0.000	0.177	0.323	0.490	0.478
	SCQ _E	0.000	0.055	0.167	0.301	0.434	0.000	0.146	0.253	0.375	0.434

SD(SCQ), comprises 9454 rules in its polarity-based subset (we will always refer to this subset from now on). SCQ_S has 49831 paths, which corresponds to roughly 52% as compared to SCQ. SD(SCQ_S) has 9127 rules, which is about 97% of SD(SCQ). Correspondingly, SCQ_E has 24474 paths (26%) and SD(SCQ_E) comprises 9040 rules (96%). This tells us that way less paths were necessary to extract SDs (i.e., ASP encodings) of similar size in the case of the structural variants. Of course, as Algorithm 2 was used for this extraction, this is tied to the selected confidence thresholds (which remained unchanged across all three SDs). Nevertheless, we interpret this as a sign of much bloat in the original SCQ. This, however, is just one part of the story, as we also have to look at the evaluated, simulated questionnaire sessions, which answer question 2):

Table 5 shows the results, where each SD, namely $SD(SCQ)$, $SD(SCQ_S)$, and $SD(SCQ_E)$, is evaluated on observations Obs drawn from each structural variant of SCQ, as well as a balanced version of SCQ, which we name SCQ' . Overall, there are four sets of observations: $Obs(SCQ)$, $Obs(SCQ')$, $Obs(SCQ_S)$, and $Obs(SCQ_E)$. These are revealed throughout the evidence levels. These results are visualised in Figure 9. We can look at the plot in multiple ways: A) comparing the scores, B) comparing the SDs, and C) comparing the Obs.

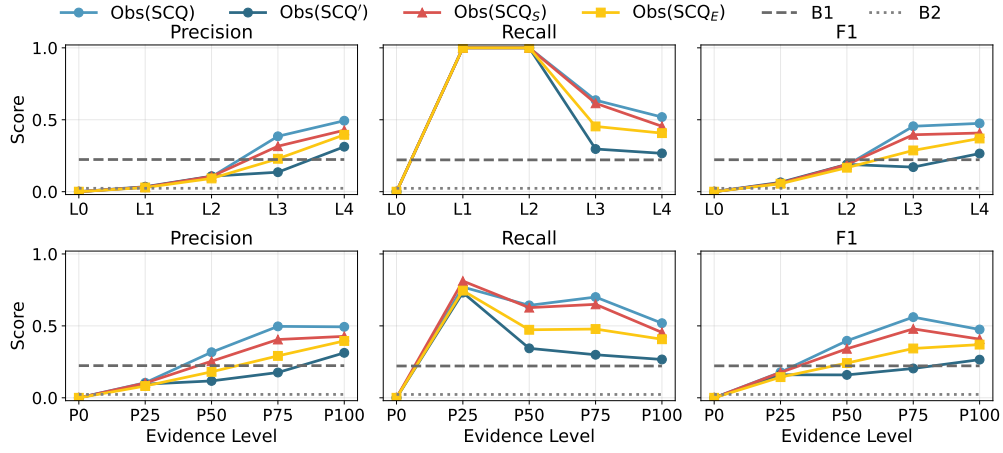
Starting with A), we can observe that Precision and F1 score generally increase throughout the evidence levels, which is expected. The increase is not monotonous in every case, however, and there are cases where both scores peak before the final evidence is presented. To be more precise, we see such an “early peak” in P75 of $SD(SCQ)$ for the observations drawn from the original SCQ, $Obs(SCQ)$, depicted in light blue. This peak of $F1 = 0.561$ is also the highest F1 score amongst all. Although not as prominent, the same “early peak” can be seen in P75 of $SD(SCQ_S)$ and $SD(SCQ_E)$. Another observation with respect to the Precision is the “long plateau”, from L0-L2, which is also expected as the discriminating evidence is not there yet. This fits well into the picture drawn in Figure 8, where information gain is low in questions 1-5 (except 4, defining the main region). Regarding Recall, it can be easily seen that it goes up immediately after L0 and peaks at L1 / P25. This can be explained by the fact that the search space with regards to possible diagnoses is wide open and not yet narrowed down in any meaningful way.

Switching now to B) the comparison of SDs, we see similar curves across all three SDs and scores but with minor differences in spreads across Obs. In fact, F1-differences of $SD(SCQ)$ and $SD(SCQ_S)$ are smaller or equal to 0.07 across evidence levels and Obs. This is another strong indicator of the SDs being very similar, although their basis is very different in the amount of paths.

Finally, we arrived at C) comparing the observations. The aforementioned spread across varies depending on the SD. The largest spread can be seen in $SD(SCQ_S)$ -Recall (0.205 on average across all evidence levels) and the least in $SD(SCQ)$ -Precision (0.147). Overall, for the average spread, it can be observed that $Recall > F1 > Precision$ (as averaged over all non-trivial evidence levels, i.e., without L0 and P0). This tells us that the Obs differ most in the simplified SD. We want to conclude this section by pointing out two additional observations: 1) Although the skewed prior baseline is strong, all SDs beat it at full evidence, across all Obs, and 2) the balanced set of observations $Obs(SCQ')$ consistently performs worse than the regular $Obs(SCQ)$, which further highlights the skewed distribution of diagnoses in the original SCQ.

5.2.4 Threats to Validity & Use of Generative AI

One of the base threats to validity of this study is the fact that we web-scrape the dataset we refer to as SCQ. While we check our data by hand, scraping might introduce artifacts. Furthermore, we only acquired a subtree of the full SymptomChecker questionnaire, and all assumptions are being made based on this. We are aware that our SCQ dataset is imbalanced, which we documented. Some diseases appear disproportionately more than others, causing a skewed prior if we consider it as the relative frequency of any individual disease. The other approach is using the prior as $\frac{1}{|D|}$ for each disease. Neither of these numbers is representative of actual, real-world frequencies of any disease following from skin as the main affected region, and the resulting methods will be biased. Some other datasets, such as DDXPlus [23], do take this into account, while we do not. Our data is based solely on the subtree extracted from SymptomChecker, as the goal of this study is to investigate what is there before acting on it. Although we cannot make an informed



(a) SD(SCQ).

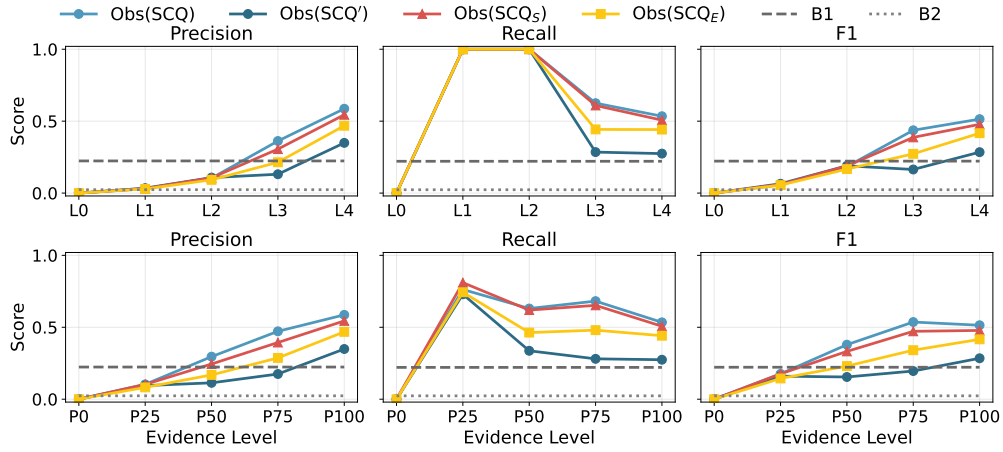
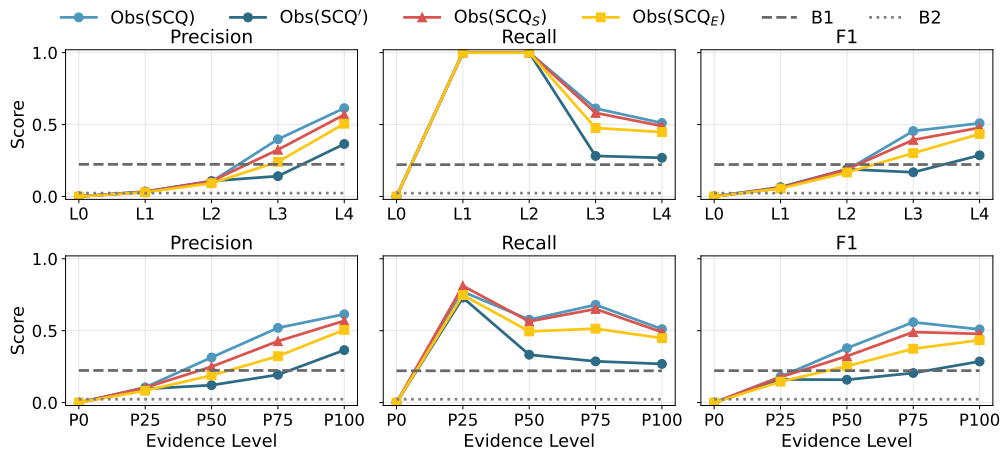
(b) SD(SCQ_S).(c) SD(SCQ_E).

Figure 9 Abductive Reasoning Results: Comparing three system descriptions, each derived from a structural variant of SCQ using Algorithm 2. Corresponds to Table 5.

assessment by just looking at a subtree, we assume that the full SymptomChecker tree is also imbalanced. While we use the balanced subsample SCQ' of SCQ during hyperparameter search, we do not exclude the SCQ' samples from SCQ and the other structural variants when producing SDs and drawing Obs. Nor yet do we use separate sets of paths for SD and Obs in general. This decision was made as the subject of our investigation, SymptomChecker, does not operate on learned knowledge but rather complete traversals on predefined paths and because there are vanishingly few samples for some of the diagnoses. Another threat is our entropy-based optimisation strategy re-orders questions and might lead to unwanted side effects in their semantics, which is partially circumvented by only re-ordering the binary follow-up questions after persona, region, and main symptom have been covered in the original order. Similarly, simplifying the tree might remove wanted redundancies. However, here we must make a clear distinction between meticulously planned medical questionnaires as found in scientific studies, where redundancy is often a wanted feature, and the case of SymptomChecker, which is a web application for information retrieval and most likely retrieves questions and their order programmatically, based on rules and a database. Neither the simplified nor entropy-optimised variants should be understood as better replacements to the original questionnaire but rather give a contrasting view on matters which might not have been considered when originally arranging the questions. Lastly, it must be emphasised that neither the scraped SCQ dataset nor its structural variants are validated by medical experts.

Parts of this work were done with the help of AI. Namely, we used Grammarly for spellchecking, grammar correction and partial rephrasing as well as Claude Code (Opus 4.7 & 4.8) also for parts of the implementation, including the visualisations and realising the algorithms and for preparing the replication package. We thoroughly checked all the generated code.

6 Conclusion

In this work, we analyse NetDoktor's SymptomChecker [19] questionnaire through the lens of abductive reasoning and answer questions about redundancy and optimality of question order. For this, we acquired a dataset called SCQ through the methodology described in Section 3.1, which links symptoms to diagnoses. SCQ represents a subset of the full SymptomChecker questionnaire and comprises over 90000 individual sessions with SymptomChecker. As we can represent SCQ as a decision tree, we refer to the sessions as paths and are able to apply algorithms on the basis of tree transformation. Thus, we produced structural variants of the SCQ tree, which we used in a comparative experimental evaluation. Our experiments revealed structural deficiencies within SCQ, ranging from a disproportionate representation of some diagnoses to redundant subtrees comprising questions without any contribution to the final diagnosis. Furthermore, we could demonstrate by comparison that the information gain throughout the question paths is sub-optimal and showed how to optimise the questionnaire using pruning and entropy-based reconstruction. What is more, our abductive reasoning experiments indicate, by very close F1-scores across SDs extracted from SCQ, SCQ_S, and SCQ_E (the gap being 0.070 at most / 0.019 on average), that the optimised variants of SCQ carry most of the original's meaning (97% / 96%, as compared by the size of a derived formal system description), while being much smaller (52% / 26%, as compared by the number of paths). Ultimately, we can answer our research questions:

- **RQ1:** Do all questions in SymptomChecker contribute to the outcome (i.e., the diagnoses) or is there redundancy?

Yes, there is redundancy. There are whole subtrees of QA pairs that are not contributing to the diagnoses. This means answering yes or no leads to the same outcome.

- **RQ2:** Is SymptomChecker’s question order optimal with respect to the information gained at each step?

No, the comparison between various question orders revealed that there are better ways to order questions and thus it is not optimal with regard to information gain for the observed subtree. However, this must be taken with a grain of salt considering the unknown size (amongst other properties) of the whole questionnaire tree.

Despite the mentioned caveats, NetDoktor’s data has potential for being used as a reference for other systems as it originates from an expert-curated database and is evaluated through medical professionals. In the future, we want to use our methodologies for testing large language models on the task of medical diagnosis, as demonstrated in our previous work [15, 14]. Moreover, we would like to compare the data derived from NetDoktor with other datasets such as DDXPlus [23], which also incorporates real-world disease distributions and census information.

References

- 1 AMBOSS GmbH. AMBOSS, 2026. URL: www.amboss.com.
- 2 Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. HAL, 2008. URL: inria.hal.science/hal-03367725.
- 3 Thomas Eiter and Georg Gottlob. The Complexity of Logic-Based Abduction. *Journal of the ACM*, 42(1):3–42, January 1995. doi:10.1145/200836.200838.
- 4 Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Hypothesis Classification, Abductive Diagnosis and Therapy. In J. Siekmann, G. Goos, J. Hartmanis, Georg Gottlob, and Wolfgang Nejdl, editors, *Expert Systems in Engineering Principles and Applications*, volume 462, pages 69–78. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990. Series Title: Lecture Notes in Computer Science. doi:10.1007/3-540-53104-1_32.
- 5 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming*, 19(1):27–82, January 2019. doi:10.1017/S1471068418000054.
- 6 Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In *In Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080, Cambridge, MA, USA, 1988. MIT Press.
- 7 W. Hannöver, M. Richard, N.B. Hansen, Z. Martinovich, and H. Kordy. A Classification Tree Model for Decision-Making in Clinical Practice: An Application Based on the Data of the German Multicenter Study on Eating Disorders, Project TR-EAT. *Psychotherapy Research*, 12(4):445–461, December 2002. doi:10.1080/713664470.
- 8 Antonis C Kakas, Robert A. Kowalski, and Francesca Toni. Abductive Logic Programming. *Journal of Logic and Computation*, 2(6):719–770, 1992. doi:10.1093/logcom/2.6.719.
- 9 Vladimir Lifschitz. Action Languages, Answer Sets, and Planning. In *The Logic Programming Paradigm: A 25-Year Perspective*, pages 357–373. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. doi:10.1007/978-3-642-60085-2_16.
- 10 Peter Lucas. Quality Checking of Medical Guidelines through Logical Abduction. In *International conference on innovative techniques and applications of artificial intelligence*, pages 309–321. Springer, 2003. doi:10.1007/978-0-85729-412-8_23.
- 11 Emir Mujić, Alexander Perko, and Franz Wotawa. Extraction of knowledge representations for reasoning from medical questionnaires. In *Proceedings of the 28th International Multiconference Information Society*, pages 781–784, Ljubljana, Slovenia, 2025. Jožef Stefan Institute. doi:10.70314/is.2025.gptzdravje.7.

- 12 Iulia-Dana Nica, Franz Wotawa, and Oliver Tazl. Chatbot-Based Tourist Recommendations Using Model-Based Reasoning. In *Proceedings of the 20th International Configuration Workshop*, volume 2220 of *CEUR*, 2018. URL: ceur-ws.org/Vol-2220/05_CONFWS18_paper_31.pdf.
- 13 Charles Sanders Peirce. *Collected Papers of Charles Sanders Peirce*. Harvard University Press, 1931.
- 14 Alexander Perko, Iulia Nica, and Franz Wotawa. Using Combinatorial Testing for Prompt Engineering of Large Language Models in Medicine. In *Proceedings of the 27th International Multiconference Information Society*, pages 930–935, Ljubljana, Slovenia, 2024. Jožef Stefan Institute. doi:10.70314/is.2024.ichtm.10.
- 15 Alexander Perko and Franz Wotawa. Testing ChatGPT’s Performance on Medical Diagnostic Tasks. In *Proceedings of the 27th International Multiconference Information Society*, pages 914–919, Ljubljana, Slovenia, 2024. Jožef Stefan Institute. doi:10.70314/is.2024.ichtm.7.
- 16 Júlia Pukancová and Martin Homola. Abductive Reasoning with Description Logics: Use Case in Medical Diagnosis. In *Proceedings of the 28th International Workshop on Description Logics*, volume 1350 of *CEUR*, 2015. URL: ceur-ws.org/Vol-1350/paper-60.pdf.
- 17 J. Ross Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986. doi:10.1023/A:1022643204877.
- 18 Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, April 1987. doi:10.1016/0004-3702(87)90062-2.
- 19 Jens Richter, Hans-Richard Demel, Florian Tiefenböck, Luise Heine, and Martina Feichter. Symptom-Checker, 2026. URL: www.netdoktor.at/symptom-checker/.
- 20 Claude Elwood Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, July 1948. doi:10.1002/j.1538-7305.1948.tb01338.x.
- 21 Ralf Strobl, Michael Grözing, Andreas Zwergal, Doreen Huppert, Philipp Filippopoulos, and Eva Grill. A Set of Eight Key Questions Helps to Classify Common Vestibular Disorders—Results From the DizzyReg Patient Registry. *Frontiers in Neurology*, 12:670944, April 2021. doi:10.3389/fneur.2021.670944.
- 22 Oliver A. Tazl, Alexander Perko, and Franz Wotawa. Conversational Recommendations Using Model-Based Reasoning. In *Proceedings of the 21th International Configuration Workshop*, volume 2467 of *CEUR*, 2019. URL: ceur-ws.org/Vol-2467/paper-03.pdf.
- 23 Arsene Fansi Tchango, Rishab Goel, Zhi Wen, Julien Martel, and Joumana Ghosn. DDXPlus: A New Dataset for Automatic Medical Diagnosis. In *Advances in Neural Information Processing Systems 35*, pages 31306–31318, New Orleans, Louisiana, USA, 2022. Neural Information Processing Systems Foundation, Inc. (NeurIPS). doi:10.52202/068431-2270.
- 24 Kaishuai Xu, Yi Cheng, Wenjun Hou, Qiaoyu Tan, and Wenjie Li. Reasoning Like a Doctor: Improving Medical Dialogue Systems via Diagnostic Reasoning Process Alignment. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6796–6814, Bangkok, Thailand, 2024. ACL. doi:10.18653/v1/2024.findings-acl.406.

Early Failure Prediction from Near-Anomaly Detection: A Proactive Approach

Léa Billet ✉ 

Schaeffler, Toulouse, France

LAAS-CNRS, University of Toulouse, INSA, France

Louise Travé-Massuyès ✉ 

LAAS-CNRS, University of Toulouse, CNRS, France

Elodie Chanthery ✉ 

LAAS-CNRS, University of Toulouse, INSA, France

Alexandre Gaffet ✉ 

Schaeffler, Toulouse, France

Abstract

Anomaly detection methods often have uncertain behavior with respect to samples near the distribution boundary, limiting their ability to anticipate future anomalies. This work introduces the concept of near-anomalies that, while not yet anomalous, lie close to the boundary and are likely to transition into anomalies in the near future. To address this, we propose an unsupervised method, named Christoffel-based ANomaly Anticipation for eaRly dIScovery (CANARI), which leverages the strong theoretical foundations of the Christoffel function to detect near-anomalies. The method is validated on industrial in-circuit testing data from printed circuit boards, with synthetically generated near-anomaly samples due to the lack of real-world data labeling. Experimental results show that CANARI outperforms the compared baselines that generally use a dual-threshold mechanism (one for anomalies and one for near-anomalies). It therefore provides a proactive solution for anticipating anomalies before they occur, offering a promising approach for resilience, predictive maintenance, and quality control.

2012 ACM Subject Classification Computing methodologies → Anomaly detection

Keywords and phrases Unsupervised anomaly detection, Christoffel function, Near-anomalies

Digital Object Identifier 10.4230/OASICS.DX.2026.4

Funding Our work has benefited from the AI Interdisciplinary Institute ANITI. ANITI is funded by the France 2030 program under the Grant agreements n°ANR-19-P3IA-0004 and n°ANR-23-IACL-0002.

Acknowledgements The authors would like to thank Christophe Merle, Head of the Manufacturing Intelligent 5 team at Schaeffler and ANITI Industrial Coordinator, for sharing his expertise in robotized production lines.

1 Introduction

Outliers, or anomalies, are typically defined as samples that are rare or out-of-distribution compared to inliers [11]. However, in many real-world applications, potentially problematic samples may not yet qualify as true anomalies at the time of observation. Instead, they may lie near the boundary of the normal distribution, remaining technically within its limits. We refer to such samples as near-anomalies. While not classical outliers, they are of critical interest because their detection enables the early anticipation of future anomalies, thereby enhancing system resilience. In this work, we focus on identifying these near-anomaly samples – those that, though still part of the distribution, are precariously close to its edge. The method is applied to data from industrial manufacturing plant product test stations.



© Léa Billet, Louise Travé-Massuyès, Elodie Chanthery, and Alexandre Gaffet; licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 4; pp. 4:1–4:18

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Near-anomaly detection has been explored in image anomaly detection, where only a few pixels deviate from the normal distribution [13]. The generation of near-anomaly images has also been proposed for method evaluation [9], notably using diffusion-based models. However, near-anomaly detection for high-dimensional tabular data remains relatively unexplored, despite its strong potential for predicting later problems.

Tabular anomaly detection methods typically score samples within a distribution and apply a threshold to classify them as inliers or outliers [12]. They are usually trained in a semi-supervised manner, using only inlier data. In principle, these methods can be adapted to detect near-anomalies by introducing a second threshold closer to the anomaly boundary. This near-anomaly threshold may be set in a supervised manner, for example, using the dataset’s known contamination rate or defining it as a percentage of the primary anomaly threshold. However, in an unsupervised setting, such a hyperparameter cannot be automatically determined for all datasets.

In this paper, we propose a method, named Christoffel-based ANomaly Anticipation for eaRly dIScovery (CANARI), based on the Christoffel Function (CF), where a second threshold to detect near-anomaly samples is derived by leveraging the theoretical properties of the CF. While we use the CF-based method CLOE (CLOE) [2] for its scalability with respect to the feature space dimension, any other CF-based method, like DyCF [4], could equally apply even if it is more limited with respect to the dimension. CLOE relies on an autoencoder trained with a CF regularized loss to construct a well-structured, low-dimensional representation of the data, which can then be advantageously used for anomaly detection. We extend the CLOE framework to identify samples that, while not yet outliers, lie close enough to the decision boundary to be classified as near-anomalies – potential precursors to future anomalies.

Experiments are conducted on printed circuit boards (PCBs) using in-circuit testing (ICT) tabular data acquired on industrial manufacturing plant product test stations. In manufacturing plants, products undergo extensive testing during production to ensure quality, with results typically recorded as high-dimensional tabular data. Quality engineers define knowledge-based thresholds for each test, and products whose values remain near these thresholds may still pass inspection and proceed through the manufacturing process. Nevertheless, these borderline products can later exhibit issues in subsequent production steps or, more seriously, after delivery to the customer. Because thresholds are defined before all production features and distribution characteristics are fully known, such products are not necessarily labeled as anomalies at the time of testing. A possible approach would be to lower the test station thresholds in order to classify more products as anomalies. However, the appropriate reduction for each threshold is unknown and cannot be reliably determined in advance. Moreover, dependencies may exist between different testing station thresholds, which makes manual adjustment even more difficult. As a result, a more flexible method is needed to identify products that are not yet anomalous but are sufficiently close to abnormal conditions to warrant attention. Detecting them as near-anomalies would therefore provide an opportunity to raise early warnings of potential future quality problems.

The paper is organized as follows: background about the CF is presented in Section 2. Then the method CANARI is described in Section 3 introducing the concept of near-anomaly and its detection. In Section 4, the set of conducted experiments and the associated results are presented with a description of the industrial use case. Finally, the conclusion and future work are given in Section 5.

2 Background - The Christoffel function for anomaly detection

This section defines the CF and establishes the background for our method [6], [7].

2.1 The Christoffel Function

Let μ be a finite Borel measure¹ supported on $\Omega \subset \mathbb{R}^d$, a compact set with a non-empty interior. Let $\mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$ and let $\alpha = (\alpha_i)_{i=1\dots d} \in \mathbb{N}^d$ be the vector of degrees associated to each variable for the monomial $\mathbf{x}^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} \dots x_d^{\alpha_d}$ of total degree $\deg(\alpha) = \sum_{i=1}^d \alpha_i$. Let $\mathbb{N}_n^d = \{\mathbb{N}^d; \deg(\alpha) \leq n\}$ and $\mathbf{v}_n(\mathbf{x}) := (\mathbf{x}^\alpha)_{\alpha \in \mathbb{N}_n^d}$ be the vector of all monomials of degree less or equal to n graded in the lexicographic order². The size of the vector $\mathbf{v}_n(\mathbf{x})$ is equal to $s_d(n) = \binom{d+n}{n}$.

Let $\mathbf{M}_n(\mu)$ be the moment matrix of μ . This matrix encodes information that quantifies the location, the shape, and the scale of Ω . $\mathbf{M}_n(\mu)$ is a real symmetric, positive definite, and non-singular matrix for all n , and is given by

$$\mathbf{M}_n(\mu) = \int_{\mathbb{R}^d} \mathbf{v}_n(\mathbf{x}) \mathbf{v}_n(\mathbf{x})^T d\mu(\mathbf{x}) \quad (1)$$

Let us introduce the Christoffel-Darboux kernel K_n^μ associated with μ . Given any orthonormal basis $(p_i)_{i=1}^{s_d(n)}$ of $\mathbb{R}_n[\mathbf{x}]$, orthonormal with respect to the inner product induced by $\mathbf{M}_n(\mu)$, K_n^μ is defined as:

$$(\mathbf{x}, \mathbf{y}) \mapsto K_n^\mu(\mathbf{x}, \mathbf{y}) := \sum_{i=1}^{s_d(n)} p_i(\mathbf{x}) p_i(\mathbf{y}). \quad (2)$$

This kernel can also be expressed in terms of the moment matrix:

$$K_n^\mu(\mathbf{x}, \mathbf{y}) := \mathbf{v}_n(\mathbf{x})^T \mathbf{M}_n(\mu)^{-1} \mathbf{v}_n(\mathbf{y}). \quad (3)$$

► **Definition 1** (The Christoffel Function). *The Christoffel Function (CF) of degree $n \in \mathbb{N}$ associated with the measure μ , denoted by Λ_n^μ , is given for any $\mathbf{x} \in \mathbb{R}^d$ as:*

$$\Lambda_n^\mu(\mathbf{x}) = K_n^\mu(\mathbf{x}, \mathbf{x})^{-1}, \quad (4)$$

which can be shown to be the solution of this minimization problem:

$$\Lambda_n^\mu(\mathbf{x}) = \min_{P \in \mathbb{R}_n[\mathbf{x}]} \left\{ \int_{\Omega} P(\mathbf{z})^2 d\mu(\mathbf{z}), \quad P(\mathbf{x}) = 1 \right\}, \quad \forall \mathbf{x} \in \mathbb{R}^d. \quad (5)$$

The intuition behind the Christoffel function can be understood from the minimization problem (5). The solution polynomial P of (5) is constrained to satisfy $P(\mathbf{x}) = 1$ at a given point $\mathbf{x}$ while minimizing the L^2 norm $\int_{\Omega} P^2 d\mu$ over the support Ω . If $\mathbf{x} \notin \Omega$ (outside the support), P can be nearly zero on Ω while still satisfying $P(\mathbf{x}) = 1$, resulting in a small minimal value. Conversely, if $\mathbf{x} \in \Omega$ (inside the support), the constraint $P(\mathbf{x}) = 1$ forces P to have significant magnitude on Ω , preventing it from being close to zero and thus yielding a large minimal value.

One of the main and salient features of the CF is its ability to encode the support Ω . The geometric shape of Ω can indeed be captured by a specific level set $\Omega_\gamma := \{\mathbf{x} : \Lambda_n^\mu(\mathbf{x})^{-1} \leq \gamma\}$, defined for some $\gamma \in \mathbb{R}_+$, even for low degrees n [4].

¹ A *Borel measure* is a way to assign a “size” (like length, area, or probability) to subsets of a topological space (here Ω), where the subsets we can measure are all those that can be built from open sets using countable unions and complements. On the real line, the usual *length* is a Borel measure which tells how long any interval is, and by extension, how long more complex sets (like unions of intervals) are.

² Lexicographic order means that monomials are first sorted by degree; then, within each degree, they are ordered lexicographically by the variables, with $X_1 = a$, $X_2 = b$, etc.

2.2 The empirical Christoffel Function

In practical applications, the theoretical measure μ is unknown. Instead, one has access to a cloud of data points $\mathbb{X} = \{\mathbf{x}_i, i = 1, \dots, N\}$ sampled from μ and associated with a discrete measure $\mu_N := \frac{1}{N} \sum_{\mathbf{x} \in \mathbb{X}} \delta_{\mathbf{x}}$ supported on $\mathbb{X}$ (where $\delta_{\mathbf{x}}$ stands for the Dirac measure at $\mathbf{x}$).

The empirical version of the moment matrix writes as

$$\mathbf{M}_n(\mu_N) = \frac{1}{N} \sum_{\mathbf{x} \in \mathbb{X}_e} \mathbf{v}_n(\mathbf{x}) \mathbf{v}_n(\mathbf{x})^T. \quad (6)$$

To guarantee the invertibility of the matrix $\mathbf{M}_n(\mu_N)$, the size of $\mathbb{X}$ must be greater than $s_d(n)$ according to [7] (Corollary 6.3.5). Under the condition $|\mathbb{X}| = N > s_d(n)$, the inverse of the empirical CF (eCF) is defined as:

$$\Lambda_n^{\mu_N}(\mathbf{x})^{-1} := \mathbf{v}_n(\mathbf{x})^T \mathbf{M}_n(\mu_N)^{-1} \mathbf{v}_n(\mathbf{x}), \mathbf{x} \in \mathbb{X}. \quad (7)$$

It has been proven that the eCF converges to the CF as N tends to infinity [7] (Theorem 6.2.3). This convergence ensures that the properties of the CF also hold (under some conditions) for the eCF.

Consequently, one can expect the inverse eCF $\Lambda_n^{\mu_N}(\mathbf{x})^{-1}$ to capture the support $\mathbb{X}$ thanks to a specific level set. Defining *inliers* as data points in the support and *outliers* or *anomalies* as data points out of the support, $\Lambda_n^{\mu_N}(\mathbf{x})^{-1}$ hence comes as a good scoring function for outlier detection.

3 Near-anomaly detection

The goal of this work is to automatically detect potentially problematic samples that, while not yet true anomalies, may lie near the boundary of the normal distribution although remaining technically within its limits. To achieve this, we propose to use the dichotomy property of the growth of the CF with the degree n presented in the following section.

3.1 The Christoffel function theoretical growth

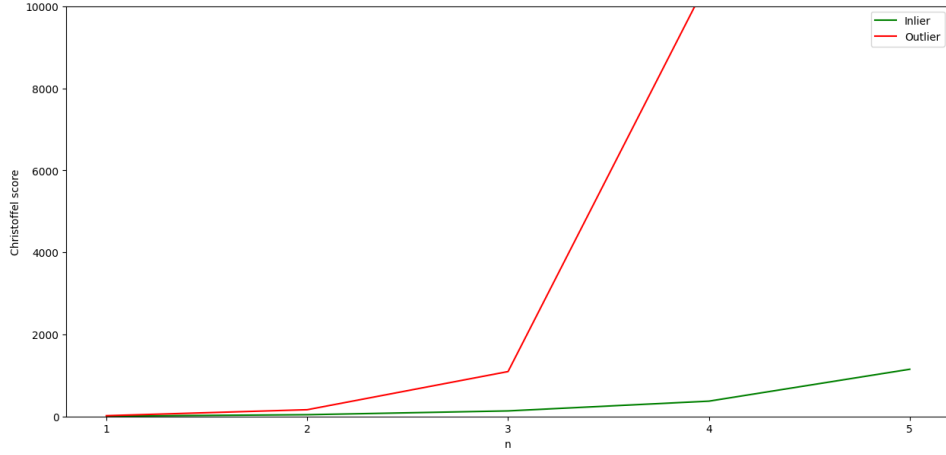
In addition to being able to capture the support of a discrete measure defined by a cloud of data points, the CF has been proven to have drastically different theoretical growths with the degree n for data points in the support of the measure, considered as inliers, and for data points out of the support of the measure, considered as outliers.

Two theorems from [7] characterize the growth of the CF as the degree n increases. The growth with n for data points in the support (inliers) is shown to be polynomial in Theorem 2 whereas it is shown to be exponential for data points out of the support (outliers) in Theorem 3. Both cases are illustrated in Figure 1 (for an outlier sample with the red graph and for an inlier sample with the green graph).

► **Theorem 2** (Lemma 4.3.2, p.51 [7]). *Let μ be a positive Borel measure supported on the compact set Ω , the closure of a bounded domain U with a nice boundary. Assume $\mu|_U$ has a density with respect to the Lebesgue measure λ restricted to U , bounded from below by $c > 0$. Let $\mathbf{x} \in \Omega$ and $\delta > 0$ such that $\text{dist}(\mathbf{x}, \partial U) \geq \delta$. Then, for any $n \in \mathbb{N}, n > 0$, and considering the equality (4), we have:*

$$\Lambda_n^{\mu}(\mathbf{x})^{-1} \leq s_d(n) \frac{2\lambda(\Omega)}{c\delta^d \omega_d} (1+d)^3 \quad (8)$$

where ω_d depends only on d and not on n .



■ **Figure 1** The figure represents the growth of an outlier in the red graph and of an inlier in the green graph as n grows from 1 to 5. The red graph increases much faster than the green graph, showing the dichotomy property stated by Theorems 2 and 3.

In equation 8 of Theorem 2, for fixed d , $s_d(n)$ is a polynomial in n of degree d :

$$s_d(n) = \binom{n+d}{n} = \frac{(n+d)(n+d-1)\cdots(n+1)}{d!} = \frac{n^d}{d!} + (\text{lower-order terms in } n).$$

So, as $n \rightarrow \infty$, it behaves like:

$$\binom{n+d}{n} \sim \frac{n^d}{d!}. \quad (9)$$

Theorem 2 hence shows that $\Lambda_n^\mu(\mathbf{x})^{-1}$ has polynomial growth with n inside the support.

► **Theorem 3** (Lemma 4.3.1, p.50 [7]). *Let μ be a positive Borel measure supported on the compact set $\Omega \subset \mathbb{R}^d$. Let $\mathbf{x} \notin \Omega$ and $\delta > 0$ such that $\text{dist}(\mathbf{x}, \Omega) \geq \delta$. Then, for any $n \in \mathbb{N}, n > 0$, and considering the equality (4), we have:*

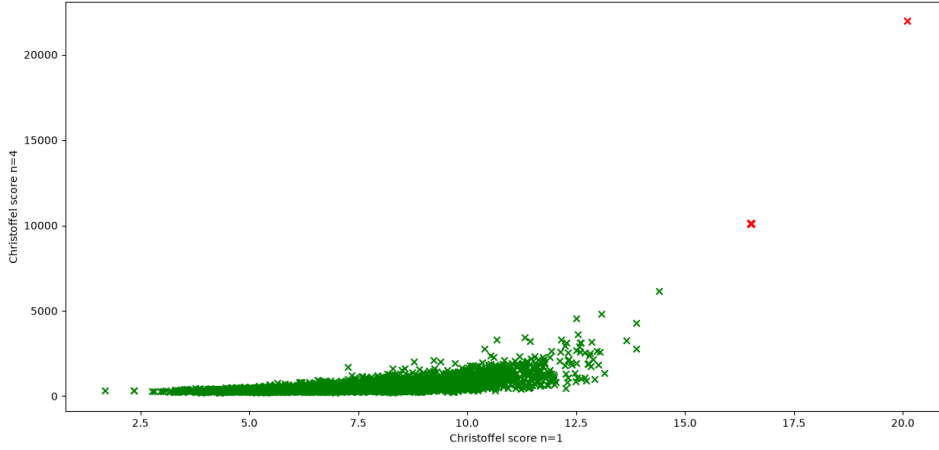
$$\frac{\Lambda_n^\mu(\mathbf{x})^{-1}}{s_d(n)} \geq 2^{\frac{\delta n}{\delta + \text{diam}(\Omega)}} - 3n^{-d} \left(\frac{d}{e}\right)^d \exp\left(-\frac{d^2}{n}\right) \quad (10)$$

Theorem 3 clearly shows that $\Lambda_n^\mu(\mathbf{x})^{-1}$ has exponential growth with n outside the support.

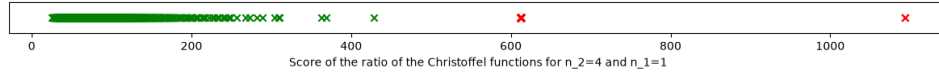
This theoretical growth with respect to the parameter n has already been leveraged by DyCG [4], which proposes a parameter-free outlier detection approach for data streams. DyCG evaluates the difference between two eCFs computed for distinct values of the parameter n , normalized by the difference between the corresponding values of n . For a fixed $x \in \mathbb{X}$, this score computes the slope of the function depending on n .

3.2 The Christoffel function growth ratio

To leverage the dichotomy property stated by Theorems 2 and 3, and given that $\Lambda_n^\mu(\mathbf{x})^{-1}$ is strictly positive, we propose to define the following *CF growth ratio*, based on two different values of n , namely n_1 and n_2 , $n_1 < n_2$ to compare eCF scores.



■ **Figure 2** The figure illustrates the eCF obtained for $n_1 = 1$ alongside that obtained for $n_2 = 4$. Inliers are in green and outliers are in red. Outlier and inlier scores are clearly distinguishable.



■ **Figure 3** The figure presents the values of the ratio between the two eCFs computed for $n_1 = 1$ and $n_2 = 4$. Inliers are represented in green, while the outliers are represented in red. A clear separation is observed, outliers are associated with much higher scores than inliers.

► **Definition 4** (CF growth ratio). *Given $\mathbf{x} \in \mathbb{X}$, the CF growth ratio is defined for $n_1, n_2 \in \mathbb{N}^*$, $n_1 < n_2$ as follows:*

$$\frac{\Lambda_{n_2}^\mu(\mathbf{x})^{-1}}{\Lambda_{n_1}^\mu(\mathbf{x})^{-1}} \quad (11)$$

A ratio was preferred to a normalized difference, such as the one used in DyCF, for two main reasons:

1. The ratio eliminates the effect of the absolute scale of the scores and avoids normalization, enabling a direct comparison of the degree's impact on the metric without bias related to the magnitude of the raw values,
2. The CF growth ratio comes with two interesting asymptotic properties.

The relative magnitude of the eCFs obtained for two different values of n can be captured through their ratio. Figure 2 illustrates the eCF score for $n_1 = 1$ along the horizontal axis and the eCF score for $n_2 = 4$ along the vertical axis. This figure highlights high values for outliers, denoted by red crosses, while inliers, denoted by green crosses, have lower values. The growth ratio characterizes the relative change of the CF as n increases, whereas DyCG [4] characterizes the absolute change. Figure 3 displays the ratio between two eCFs for $n_1 = 1$ and $n_2 = 2$ for inliers, in green, and outliers, in red. Consistently to the dichotomy property and Figure 2, outlier samples have the highest score.

► **Corollary 5.** *Given $\mathbf{x} \in \mathbb{X}$, and $n_1, n_2 \in \mathbb{N}^*$, $n_1 < n_2$, the CF growth ratio of $\mathbf{x}$ behaves like:*

$$\frac{\Lambda_{n_2}^\mu(\mathbf{x})^{-1}}{\Lambda_{n_1}^\mu(\mathbf{x})^{-1}} \sim \left(\frac{n_2}{n_1} \right)^p, \quad (12)$$

where $p \in \mathbb{N}^*$ is a constant.

Proof. Given $\mathbf{x} \in \mathbb{X}$ and $n \in \mathbb{N}^*$, from Theorem 2, there exist $C_d \in \mathbb{R}^{*+}$ and $p \in \mathbb{N}^*$ such that, $\Lambda_n^\mu(\mathbf{x})^{-1}$ behaves like:

$$\Lambda_n^\mu(\mathbf{x})^{-1} \sim C_d n^p \quad (13)$$

Taking $n_1, n_2 \in \mathbb{N}^*$, $n_1 < n_2$, we have $\Lambda_{n_1}^\mu(\mathbf{x})^{-1} \sim C_d(n_1)^p$ and $\Lambda_{n_2}^\mu(\mathbf{x})^{-1} \sim C_d(n_2)^p$, which proves Corollary 5. $\blacktriangleleft$

► **Remark 6.** Given $\mathbf{x} \in \mathbb{X}$, and $n_1, n_2 \in \mathbb{N}^*$, $n_1 \rightarrow \infty$ and $n_2 \rightarrow \infty$, the CF growth ratio of $\mathbf{x}$ behaves asymptotically like:

$$\frac{\Lambda_{n_2}^\mu(\mathbf{x})^{-1}}{\Lambda_{n_1}^\mu(\mathbf{x})^{-1}} \sim \left(\frac{n_2}{n_1}\right)^d, \quad (14)$$

► **Corollary 7.** Given $\mathbf{x} \notin \mathbb{X}$, and $n_1, n_2 \in \mathbb{N}^*$, $n_1 < n_2$, the CF growth ratio of $\mathbf{x}$ behaves like:

$$\frac{\Lambda_{n_2}^\mu(\mathbf{x})^{-1}}{\Lambda_{n_1}^\mu(\mathbf{x})^{-1}} \sim e^{(n_2-n_1)b}, \quad (15)$$

where $b \in \mathbb{R}^{*+}$ is a constant.

Proof. Given $\mathbf{x} \notin \mathbb{X}$ and $n \in \mathbb{N}^*$, from Theorem 3, there exist $a, b \in \mathbb{R}^{*+}$ such that $\Lambda_n^\mu(\mathbf{x})^{-1}$ behaves like:

$$\Lambda_n^\mu(\mathbf{x})^{-1} \sim ae^{nb} \quad (16)$$

Taking $n_1, n_2 \in \mathbb{N}^*$, $n_1 < n_2$, we have $\Lambda_{n_1}^\mu(\mathbf{x})^{-1} \sim ae^{n_1b}$ and $\Lambda_{n_2}^\mu(\mathbf{x})^{-1} \sim ae^{n_2b}$, which proves Corollary 7. $\blacktriangleleft$

3.3 Near-anomaly definition

Given that the eCF inherits the properties of the CF, this section leverages the growth properties of the CF given by Theorems 2 and 3 and the two Corollaries 5 and 7 to define near-anomalies in a practical setting.

► **Definition 8** (eCF growth ratio). Given $\mathbf{x} \in \mathbb{X}$, the eCF growth ratio $\mathcal{G}_{n_2/n_1}(\mathbf{x})$ is defined for $n_1, n_2 \in \mathbb{N}^*$, $n_1 < n_2$ as follows:

$$\mathcal{G}_{n_2/n_1}(\mathbf{x}) = \frac{\Lambda_{n_2}^{\mu_N}(\mathbf{x})^{-1}}{\Lambda_{n_1}^{\mu_N}(\mathbf{x})^{-1}} \quad (17)$$

The following definition defines *near-anomalies* thanks to the eCF growth ratio.

► **Definition 9** (Near-anomaly definition). Given the eCF growth ratio for $n_1, n_2 \in \mathbb{N}^*$, $n_1 < n_2$, a data point $\mathbf{x} \in \mathbb{X}$ is defined as a near-anomaly if $\mathcal{G}_{n_2/n_1}(\mathbf{x}) > \tau_{na}$, where $\tau_{na} \in \mathbb{R}$ is a given threshold.

This definition proposes to define near-anomalies as data points that are inliers but whose eCF shows a growth significantly superior to core inliers. Interestingly, in the following section, we derive a method to set automatically the near-anomaly threshold τ_{na} .

3.4 Automatic determination of the near-anomaly threshold

The results provided in this section allow to derive a method to set automatically the near-anomaly threshold.

From Corollaries 5 and 7, the eCF growth ratio $\mathcal{G}_{n_2/n_1}(\mathbf{x})$ is expected to remain relatively low for inliers ($\mathbf{x} \in \mathbb{X}$, where $\mathbb{X}$ corresponds to the support of μ_N) while being much higher for outliers/anomalies ($\mathbf{x} \notin \mathbb{X}$). Near-anomalies are inliers but they lie in a range in which the eCF growth ratio is significantly superior to that of core inliers. Let us define as $\mathbb{X}_c \subset \mathbb{X}$ the set of core inliers and by $\mathbb{X}_{na} \subset \mathbb{X}$ the set of near-anomalies that we want to identify. For this purpose, there is a theoretical result that can be useful. The following lemma indeed gives the theoretical mean of the eCF on the support of the distribution.

► **Lemma 10** (Theoretical mean value of eCF in the support). *The theoretical mean value of the eCF over the support of μ_N is $s_d(n)$.*

Proof.

$$\begin{aligned}
 \int \Lambda_n^{\mu_N}(\mathbf{x})^{-1} d\mu_N(\mathbf{x}) &= \int K_n^{\mu_N}(\mathbf{x}, \mathbf{x}) d\mu_N(\mathbf{x}) \quad (\text{from Equation (4)}) \\
 &= \sum_{i=1}^{s_d(n)} \int p_i(\mathbf{x})^2 d\mu_N(\mathbf{x}) \quad (\text{from Equation (2)}) \\
 &= \sum_{i=1}^{s_d(n)} 1 \quad (\text{as } (p_i)_{i=1}^{s_d(n)} \text{ is an orthonormal basis of } \mathbb{R}_n[\mathbf{x}]) \\
 &= s_d(n)
 \end{aligned} \tag{18}$$

◀

To leverage this result, we propose to use the Chebyshev inequality [10] to identify the inliers low and high eCF growth ratio ranges. According to this inequality, for random distributions, at least $100(\frac{k^2-1}{k^2})\%$ of the samples are in the range given by the following interval:

$$[\bar{x} - k \times \sigma; \bar{x} + k \times \sigma], \tag{19}$$

where $\bar{x}$ and σ are the mean and the standard deviation of the distribution, respectively. We further assume that the interval upper bound $(\bar{x} + k \times \sigma)$ can be used as a threshold to identify high values and hence near-anomalies, which would amount to no more than $100(\frac{1}{k^2})\%$ of the samples.

Let us apply Chebyshev inequality to our problem, which translates in the following definition.

► **Definition 11** (Core inliers and near-anomaly samples). *Let us define $\mathbb{E}[\mathcal{G}_{n_2/n_1}]$ and $\sigma(\mathcal{G}_{n_2/n_1})$ as the mean and the standard deviation of the eCF growth ratio $\mathcal{G}_{n_2/n_1}$, core inliers and near-anomaly samples are characterized as follows:*

$$\forall \mathbf{x} \in \mathbb{X}_c \Leftrightarrow \mathcal{G}_{n_2/n_1}(\mathbf{x}) \in]0; \mathbb{E}[\mathcal{G}_{n_2/n_1}] + k \times \sigma(\mathcal{G}_{n_2/n_1})], \tag{20}$$

$$\forall \mathbf{x} \in \mathbb{X}_{na} \Leftrightarrow \mathcal{G}_{n_2/n_1}(\mathbf{x}) > \mathbb{E}[\mathcal{G}_{n_2/n_1}] + k \times \sigma(\mathcal{G}_{n_2/n_1}). \tag{21}$$

From Definition 11, it comes that the threshold τ_{na} that we want to determine is:

$$\tau_{na} = \mathbb{E}[\mathcal{G}_{n_2/n_1}] + k \times \sigma(\mathcal{G}_{n_2/n_1}). \tag{22}$$

In consequence, our problem comes down to determining $\mathbb{E}[\mathcal{G}_{n_2/n_1}]$ and $\sigma(\mathcal{G}_{n_2/n_1})$. The main result below provides a way to compute τ_{na} thanks to Lemma 10, which provides the theoretical mean value of eCF in the support, and some empirical computations.

► **Theorem 12** (Practical near-anomaly threshold). *Consider $n_1, n_2 \in \mathbb{N}^*$, $n_1 < n_2$, $k \in \mathbb{N}^*$, and $m, \Gamma \in \mathbb{R}^{+*}$, then a practical near-anomaly threshold $\tilde{\tau}_{na}$ is given by:*

$$\tilde{\tau}_{na} = \frac{s_d(n_2)}{m} + k \times \Gamma \quad (23)$$

$$\text{where } s_d(n_2) = \binom{n_2 + d}{n_2}, \quad (24)$$

$$m = \min \left\{ \left(\Lambda_{n_1}^{\mu_N}(\mathbf{x}) \right)^{-1}, \mathbf{x} \in \mathbb{X} \right\}, \quad (25)$$

$$\Gamma = \sqrt{\text{var} \left\{ \mathcal{G}_{n_2/n_1}(\mathbf{x}), \mathbf{x} \in \mathbb{X} \right\}}. \quad (26)$$

Proof. From (22), we have $\tau_{na} = \mathbb{E}[\mathcal{G}_{n_2/n_1}] + k \times \sigma(\mathcal{G}_{n_2/n_1})$ and we have:

$$\mathbb{E}[\mathcal{G}_{n_2/n_1}] = \int \mathcal{G}_{n_2/n_1}(\mathbf{x}) d\mu_N(\mathbf{x}) = \int \frac{\Lambda_{n_2}^{\mu_N}(\mathbf{x})^{-1}}{\Lambda_{n_1}^{\mu_N}(\mathbf{x})^{-1}} d\mu_N(\mathbf{x})$$

As $\Lambda_{n_1}^{\mu_N}(\mathbf{x})^{-1}$ is lower bounded on μ_N , there exists a constant $m \in \mathbb{R}^{+*}$ such that $\Lambda_{n_1}^{\mu_N}(\mathbf{x})^{-1} \geq m$ [7]. So,

$$\int \frac{\Lambda_{n_2}^{\mu_N}(\mathbf{x})^{-1}}{\Lambda_{n_1}^{\mu_N}(\mathbf{x})^{-1}} d\mu_N(\mathbf{x}) \leq \int \frac{\Lambda_{n_2}^{\mu_N}(\mathbf{x})^{-1}}{m} d\mu_N(\mathbf{x}) = \frac{1}{m} \int \Lambda_{n_2}^{\mu_N}(\mathbf{x})^{-1} d\mu_N(\mathbf{x})$$

and from Lemma 10:

$$\int \frac{\Lambda_{n_2}^{\mu_N}(\mathbf{x})^{-1}}{\Lambda_{n_1}^{\mu_N}(\mathbf{x})^{-1}} d\mu_N(\mathbf{x}) \leq \frac{s_d(n_2)}{m} \Leftrightarrow \tau_{na} \leq \frac{s_d(n_2)}{m} + k \times \sigma(\mathcal{G}_{n_2/n_1}).$$

Because there is no theoretical result providing m and $\sigma(\mathcal{G}_{n_2/n_1})$, we propose to compute them empirically from the data set $\mathbb{X}$ at hand³. Finally,

$$\tau_{na} \leq \frac{s_d(n_2)}{\min \left\{ \left(\Lambda_{n_1}^{\mu_N}(\mathbf{x}) \right)^{-1}, \mathbf{x} \in \mathbb{X} \right\}} + k \times \sqrt{\text{var} \left\{ \mathcal{G}_{n_2/n_1}(\mathbf{x}), \mathbf{x} \in \mathbb{X} \right\}} = \tilde{\tau}_{na}. \quad \blacktriangleleft$$

$\tilde{\tau}_{na}$ is more restrictive than τ_{na} but, as will be seen in the experiments, it provides a good practical threshold to identify near-anomalies. Algorithm 1 proposes the pseudo-code to compute the $\tilde{\tau}_{na}$ threshold.

3.5 Scaling up the Christoffel function

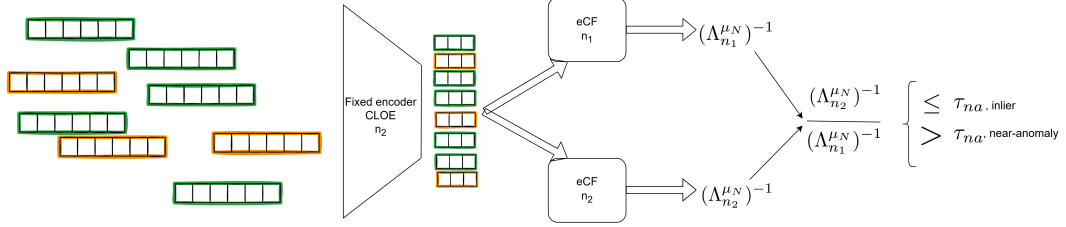
Even for moderate degrees n , the inverse of the Christoffel function is very expensive to compute if not impossible for feature dimensions $d > 8$ because of the necessity to invert the moment matrix $\mathbf{M}_n(\mu_N)$ of size $s_d(n) \times s_d(n)$ in equation (7).

To use the proposed method in high-dimensional settings, we propose to use the Christoffel-based Loss for Autoencoder (CLOE) [2]. CLOE consists of a method that applies the eCF to high-dimensional data for semi-supervised anomaly detection. It allows to reduce the dimension of high-dimensional datasets in a latent space specifically regularized with the CF score in the loss function of the autoencoder (AE).

³ To avoid division by extremely small values, a 99% percentile can be estimated from the validation set.

■ **Algorithm 1** Computation of $\tilde{\tau}_{na}$.

Input: $n_2 > n_1 > 0$,
Input: $k > 0$,
Input: Trained $(\Lambda_{n_1}^{\mu_N})^{-1}$ and $(\Lambda_{n_2}^{\mu_N})^{-1}$,
Input: Validation dataset $X_v = \{x_i\}_{i \in (1, |X_v|)}$
Output: The threshold $\tilde{\tau}_{na}$
 $d \leftarrow \text{dimension of } x_1$
 $\mathcal{G}_{n_2/n_1} \leftarrow \left\{ \frac{\Lambda_{n_2}^{\mu_N}(\mathbf{x})^{-1}}{\Lambda_{n_1}^{\mu_N}(\mathbf{x})^{-1}}, x \in X_v \right\}$ ▷ Equation 17
 $\bar{x} \leftarrow s_d(n_2) = \binom{n_2+d}{n_2}$ ▷ Equation 24
 $\min \leftarrow \min \left\{ \Lambda_{n_1}^{\mu_N}(\mathbf{x})^{-1}, x \in X_v \right\}$ ▷ Equation 25
 $\sigma \leftarrow \sqrt{\text{Var}(\mathcal{G}_{n_2/n_1})}$ ▷ Equation 26
 $\tilde{\tau}_{na} \leftarrow \frac{\bar{x}}{\min_{n_1}} + k \times \sigma$ ▷ Theorem 12
return $\tilde{\tau}_{na}$



■ **Figure 4** With normal data as input, the trained encoder from CLOE delivers low dimension encoded data in the latent space. Both near-anomaly samples, in orange, and core inlier samples, in green, are present. Then two eCF are trained, one with parameter n_1 and the other with n_2 , $n_1 < n_2$. Finally, the ratio of the two eCF scores is compared to the near-anomaly threshold, τ_{na} (in practice $\tilde{\tau}_{na}$), to decide if the data is core inlier or near-anomaly.

CLOE includes a pretraining stage and a joint training stage. The pretraining stage is performed using only the Mean Square Error (MSE) between the input and the output of the AE as the loss function. Then, the joint training stage combines the MSE with the CF-based loss to structure the latent space. For this latter step, the only hyperparameter required is the degree n for the CF computation, which is set to $n = n_{CLOE}$. The trained encoder reduces the data dimension to a lower dimension in the latent space (8 dimensions are proposed in [2]). The latent space data is then used to train the eCF. The threshold for outlier detection is chosen as the maximum value of the eCF.

In this work, CLOE is used as follows. CLOE is trained with normal data, the parameter n_{CLOE} being set to n_2 , which provides a low dimension latent space characterized by the set of moments up to the order n_2 . After this dimension reduction phase, the obtained low dimension latent space data is used to compute the eCF for the two degrees $n = n_1$ and $n = n_2$, $n_1 < n_2$, i.e., to compute $\Lambda_{n_1}^{\mu_N}(\mathbf{x})^{-1}$ and $\Lambda_{n_2}^{\mu_N}(\mathbf{x})^{-1}$, and the CANARI method can be applied to identify near-anomalies. This is illustrated in Figure 4.

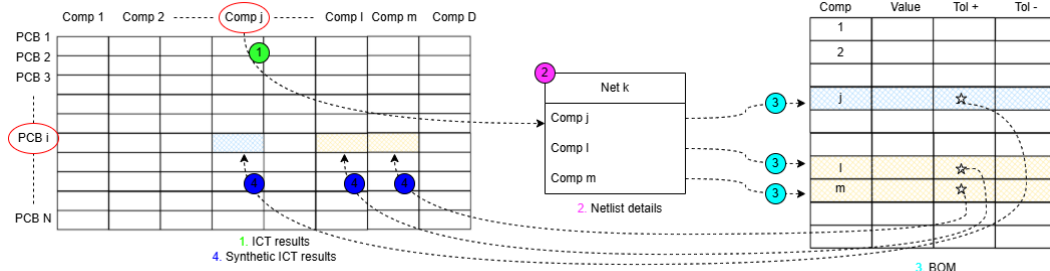
4 Experiments and results

4.1 Datasets

CANARI is applied to real production data acquired in industrial plants. In these plants, PCBs are produced in an almost fully automated way. First, the PCBs are assembled in the front-end line. Components are placed and welded during this step. At the end of this

■ **Table 1** Dimension and size of each dataset.

Dataset	D1	D2	D3	D4	D5	D6	D7	D8
Dimension	26	821	80	691	703	714	1139	18
Nb samples	57183	56334	59947	25827	10091	8421	26729	41151



■ **Figure 5** 1. A PCB product (PCB i) and one of its component (Comp j) are randomly chosen (red circle). 2. Netlist is checked to find the net (k) to which comp j belongs. All the components of the net k (j , l and m) are modified. 3. The value of these components and their upper and lower tolerances Tol+ and Tol-, respectively, are obtained from the Bill Of Materials (BOM), from which the upper and lower limits, l^+ and l^- respectively, are computed. 4. l^+ or l^- is randomly chosen. The corresponding ICT results of the concerned components are modified to appear below or above their limit.

line, ICT is performed. This test consists of injecting current into the PCB at many points on the circuit (between 10 and 2000, depending on the size of the PCB) and measuring the voltage and the current intensity to check the conformance of the PCB. Test engineers define limits for each test, and the PCB continues the manufacturing process if all its ICT results are within the limits. Otherwise, the PCB is removed from the production line. The second line is the back-end line where the PCB is encapsulated in an appropriate box. Finally, the PCB can be delivered to the customer.

The ICT results constitute tabular data with dimensions ranging from 10 to 1200, depending on the number of test points for each PCB. 8 different PCBs produced in 2023 were considered for this study, resulting in eight datasets D1 to D8. The ICT results may be PASS or FAIL. Only products that pass the ICT constitute the datasets. The dimensions of these products vary between 18 and 1139. The details of the dimension and the size for each dataset are indicated in Table 1. Each dataset is divided into 3 different sets: the training set, composed of 1150 samples, the validation set, composed of 4600 samples, and the testing set composed of the rest of the samples.

To evaluate the method, synthetic anomalies were injected into the datasets, as illustrated in Figure 5. Starting from real datasets and known ICT threshold values and upper and lower tolerances Tol+ and Tol-, respectively, from the Bill Of Materials (BOM), 10% of the samples in each test dataset were modified. First, a component is randomly chosen. Then the upper limit l^+ or lower limit l^- is randomly chosen. The ICT result for this component is modified to be close to the limit l . The order of magnitude o of the ICT result is computed with :

$$o = 10^{\lfloor \log(l^+ - l^-) \rfloor}. \quad (27)$$

The final value of the modified component is $l \pm o \times \tau_{synth}$, where τ_{synth} is the disturbed magnitude of the near anomaly samples. Finally, knowing the design of the PCB, all the components connected by the first net in the netlist, where the component belongs, are modified according to the same upper or lower decision. With this process, synthetic datasets are created taking into account the connection of the different components on the PCB and the ICT.

4.2 Baselines and Metrics

CLOE with double threshold is considered as the first baseline. The original threshold is considered to compute the new threshold to detect near-anomalies. 5%, 20% and 50% of the original threshold are used as the double threshold to detect near-anomaly samples.

Then, CANARI with different thresholds is used as the second baseline. The new thresholds are the ratio of the theoretical means of the two eCFs, τ_{mean} , and the empirical percentile 99th of the ratio, τ_{upper} .

F1-score and Matthews Correlation Coefficient (MCC) [8] are the two metrics used to evaluate CANARI. F1-score, given in Definition 13, is computed with the recall and precision metrics. Its lower value is 0 and its best value is 1.

► **Definition 13 (F1-score).** *Let TP be the true positives, TN the true negatives, FP the false positives, and FN the false negatives. The metric F1-score is defined as:*

$$F1_{-score} = \frac{2TP}{2TP + FP + FN} = 2 \frac{precision \cdot recall}{precision + recall}. \quad (28)$$

MCC, given in Definition 14, computes the Pearson product moment correlation coefficient in a contingency matrix method. Its worst value is -1, and its best value is 1, 0 corresponds to a random classification. This metric is stronger against unbalanced datasets than F1-score and accuracy [3].

► **Definition 14 (MCC).** *Let TP be the true positives, TN the true negatives, FP the false positives, and FN the false negatives. The metric MCC is defined as:*

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \quad (29)$$

The testing set is composed of two classes: core inliers and synthetic near-anomalies. Note that the core inliers testing dataset may also be corrupted by real unknown near-anomalies because it is based on real world data, so comparing the metrics for baselines and CANARI only makes sense on our datasets.

4.3 Results

4.3.1 Observation of the growth of the eCF with n

The goal of this subsection is to visually assess the eCF growth patterns for fail, pass, and near-anomaly samples. Pretraining and joint-training steps of CLOE are performed for $n_{CLOE} = 4$. Then the eCF is computed for n varying from 1 to 5. Figure 6 (a) illustrates this experiment with PCBs that fail the ICT represented by the red graphs (only 2 red graphs because only 2 such PCBs) and PCBs that pass the ICT represented by green graphs. The testing set includes fail products in this experiment. The fail samples (in red in the

■ **Table 2** Correspondence between the parameter k and the proportion of samples outside the upper bound according to the Chebyshev inequality [10].

k	1	$\sqrt{2}$	1.5	2	$2\sqrt{2}$	3	4	5	10
Maximum percentage	100%	50%	44.5%	25%	12.5%	11.11%	6.25%	4%	1%

■ **Table 3** F1-score for the 8 datasets D1 to D8 for our method and the baselines method.

Method	D1	D2	D3	D4	D5	D6	D7	D8	Mean
CANARI	0.76$\pm$0.1	0.49$\pm$0.06	0.64$\pm$0.15	0.45$\pm$0.04	0.41$\pm$0.06	0.50$\pm$0.06	0.50$\pm$0.25	0.72$\pm$0.08	0.56
CANARI- τ_{mean}	0.16 $\pm$ 2.10 $\cdot$ 10 $^{-6}$	0.21 $\pm$ 0.002	0.16 $\pm$ 10 $^{-6}$	0.31 $\pm$ 0.01	0.20 $\pm$ 0.004	0.22 $\pm$ 0.003	0.03 $\pm$ 0.003	0.07 $\pm$ 4.10 $^{-6}$	0.17
CANARI- τ_{upper}	0.25 $\pm$ 0.0004	0.21 $\pm$ 0.0009	0.22 $\pm$ 0.001	0.14 $\pm$ 0.01	0.07 $\pm$ 0.006	0.17 $\pm$ 0.02	0.49 $\pm$ 0.06	0.45 $\pm$ 0.001	0.25
CLOE $_{5\%}$	0.16 $\pm$ 0.02	0.06 $\pm$ 0.006	0.04 $\pm$ 0.0004	0.17 $\pm$ 0.01	0.15 $\pm$ 0.01	0.17 $\pm$ 0.01	0.34 $\pm$ 0.03	0.34 $\pm$ 0.04	0.18
CLOE $_{20\%}$	0.26 $\pm$ 0.02	0.09 $\pm$ 0.002	0.08 $\pm$ 0.002	0.28 $\pm$ 0.02	0.16 $\pm$ 0.01	0.24 $\pm$ 0.01	0.37 $\pm$ 0.04	0.39 $\pm$ 0.03	0.23
CLOE $_{50\%}$	0.45 $\pm$ 0.01	0.19 $\pm$ 0.005	0.27 $\pm$ 0.02	0.39 $\pm$ 0.004	0.25 $\pm$ 0.003	0.31 $\pm$ 0.02	0.41 $\pm$ 0.05	0.45 $\pm$ 0.03	0.34

image) exhibit the greatest increase, even for low values of n . Then, between $n = 3$ and $n = 4$, the graphs of some pass samples show a change in growth pattern, these products are the near-anomaly products we are looking for. The other pass samples have a polynomial growth from $n = 1$ to $n = 5$.

With this experiment, the change in the growth pattern with the parameter n is visible. Part of the distribution is clearly different from the rest of the distribution.

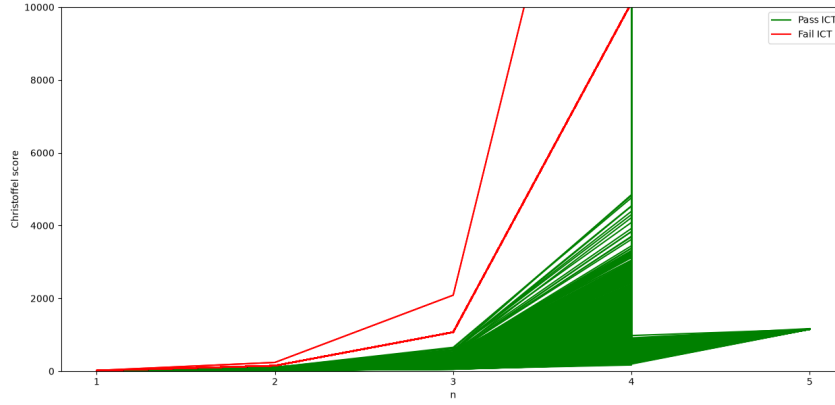
4.3.2 Evaluation of the proposed method

In this section, and for computational reasons, n_1 is fixed to 1 and n_2 is fixed to 4. Different values for the parameter k are tested, namely $k \in \{1, \sqrt{2}, 1.5, 2, 2\sqrt{2}, 3, 4, 5, 10\}$. According to the Chebyshev inequality [10], this values correspond to an upper bound on the proportion of samples falling below the defined threshold τ_{na} , as summarized in Table 2. A slightly smaller proportion falls below $\tilde{\tau}_{na}$. The value of τ_{synth} is fixed to 10^{-2} to create the disturbed magnitude of the near anomaly samples.

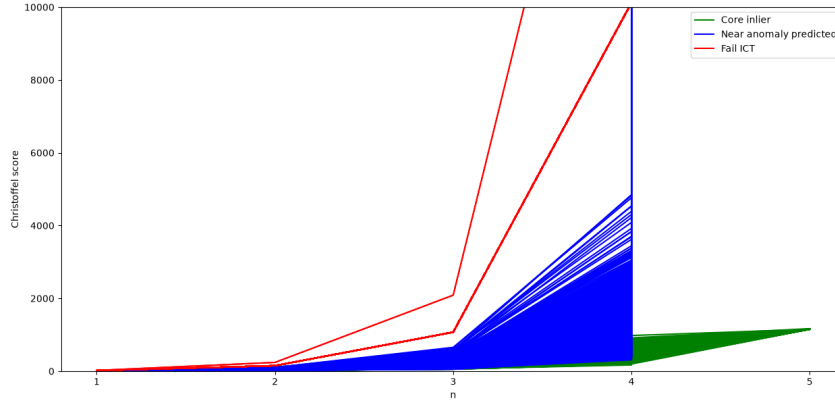
Figure 6 (b) shows the growth of the eCF with n varying from 1 to 5. The blue graphs correspond to the near-anomaly samples detected by CANARI while the green ones correspond to the samples detected as core inliers. We can observe that the detected near-anomaly samples correspond to samples whose growth lies between that of fail PCB samples and core inlier PCB samples. This observation confirms that CANARI detects samples with a growth between fail samples and core inlier samples.

Then synthetic datasets were created for PCBs as described in Section 4.1. Five runs are performed with different random seeds. The random seed is the same for each dataset and each method to ensure reproducibility across the same run. The mean and the standard deviation across these five runs are displayed in Table 3 for the F1-score metric and in Table 4 for the MCC metric.

The performance obtained by CANARI on the datasets D1 to D8 is better than double thresholded CLOE with different percentage thresholds for both metrics. For datasets D1 to D8, CANARI improves the F1-score by 65.8% and the MCC by 34.3% comparing to the CLOE methods with percentage. Classic CANARI obtains better performances than CANARI- τ_{mean} and CANARI- τ_{upper} . This shows that the proposed threshold $\tilde{\tau}_{na}$ can actually detect near anomaly samples.



(a) Pass ICT in green, and Fail ICT in red.



(b) Predicted core inliers (green) and near anomalies (blue) and real fail samples (red).

■ **Figure 6** The figure highlights the evolution of the eCF as n varies from 1 to 5. (a) displays the true label, PASS or FAIL, with the samples that succeeded ICT in green and the samples that failed ICT in red. In this dataset, only two products have failed the ICT, hence only two red graphs. (b) displays the predicted label, with core inliers in green and near anomalies in blue. In red, we find the two Fail ICT samples kept to compare their graph growth. For $n = 5$, core inliers have slightly different scores around 1500, hence the “strange” value concentration due to the scale. Between $n = 4$ and $n = 5$, the near anomaly sample scores increase so sharply that their values are out of the figure for $n = 5$, producing a “strange” blue line that appears to be vertical at $n = 4$ in the figure. Red graphs, i.e., those of Fail ICT samples, exhibit a faster growth than other graphs. The difference between the growth of the green and blue graphs, i.e., core inliers and near anomalies respectively, really occurs in the interval between $n = 3$ to $n = 4$. Over the interval from $n = 1$ to $n = 5$, we can hence distinguish a polynomial growth for the green graphs, i.e., those of core inliers, compared to an exponential growth for the red graphs, i.e., those of Fail ICT samples, the near anomaly graphs growth standing in between.

■ **Table 4** MCC for the 8 datasets D1 to D8 for our method and the baselines method.

Method	D1	D2	D3	D4	D5	D6	D7	D8	Mean
CANARI	0.76$\pm$0.09	0.48$\pm$0.07	0.61$\pm$0.17	0.47$\pm$0.04	0.39$\pm$0.08	0.49$\pm$0.04	0.50$\pm$0.25	0.73$\pm$0.08	0.55
CANARI- τ_{mean}	0.09 $\pm$ 10 ⁻⁵	0.15 $\pm$ 0.003	0.09 $\pm$ 9.10 ⁻⁵	0.28 $\pm$ 0.01	0.17 $\pm$ 0.006	0.19 $\pm$ 0.005	0.06 $\pm$ 0.001	0.07 $\pm$ 9.10 ⁻⁶	0.14
CANARI- τ_{upper}	0.37 $\pm$ 0.003	0.32 $\pm$ 0.001	0.32 $\pm$ 0.002	0.21 $\pm$ 0.02	0.11 $\pm$ 0.01	0.23 $\pm$ 0.03	0.48 $\pm$ 0.05	0.53 $\pm$ 0.0006	0.32
CLOE _{5%}	0.27 $\pm$ 0.01	0.16 $\pm$ 0.001	0.14 $\pm$ 0.001	0.27 $\pm$ 0.01	0.22 $\pm$ 0.02	0.28 $\pm$ 0.01	0.40 $\pm$ 0.04	0.43 $\pm$ 0.03	0.27
CLOE _{20%}	0.36 $\pm$ 0.02	0.20 $\pm$ 0.003	0.19 $\pm$ 0.004	0.37 $\pm$ 0.01	0.24 $\pm$ 0.02	0.34 $\pm$ 0.01	0.10 $\pm$ 0.04	0.47 $\pm$ 0.02	0.28
CLOE _{50%}	0.52 $\pm$ 0.008	0.30 $\pm$ 0.003	0.37 $\pm$ 0.02	0.45 $\pm$ 0.004	0.32 $\pm$ 0.005	0.39 $\pm$ 0.01	0.43 $\pm$ 0.05	0.52 $\pm$ 0.02	0.41

4.4 Ablation and sensitivity studies

An ablation study is performed to evaluate the robustness of CANARI. This section relies exclusively on MCC as the evaluation metric. First, Table 5 reports a sensitivity analysis of parameter k across datasets. The optimal values for k across the different datasets are $\sqrt{2}$, 1.5 and 2. Accordingly, the parameter k should be selected among these three values.

Second, Figure 7 compares different pairs of parameters n_1 and n_2 , under the constraint $n_1 < n_2$. This heatmap shows the mean MCC associated with each possible combination of n_1 and n_2 across all datasets. The best combination is the one used in CANARI, namely $n_1 = 1$ and $n_2 = 4$.

Then, Table 6 compares CLOE with the kernelized inverse Christoffel function (KIC) [1] and the univariate Christoffel function [5] as approaches for eCF computation in high dimensional datasets before applying CANARI⁴. To compute the threshold $\tilde{\tau}_{na}$, the best value of k is selected separately for each method and dataset. For KIC [1] and UCF [5] and for some datasets, the minimum value or the 99% percentile of the eCF $\Lambda_{n_1}^{\mu_N}(\mathbf{x})^{-1}$ can be very low, in the order of 10⁻⁶⁸, and the ratio between the two Christoffel functions cannot be computed. These datasets are therefore not included in the present analysis. CLOE consistently achieves the best results across all datasets. Only the KIC method yields similar performance on the last dataset.

Finally, Table 7 presents a comparison of several values of τ_{synth} , corresponding to the disturbance magnitude used for generating the synthetic near-anomaly dataset. Except for one dataset with $\tau_{synth} = 0.1$, CANARI is able to detect synthetic near-anomaly samples generated with amplitudes between 0.1 and 10⁻⁴, which illustrates the robustness of the method that does not need to define a strict near-anomaly threshold for the synthetic near anomalies.

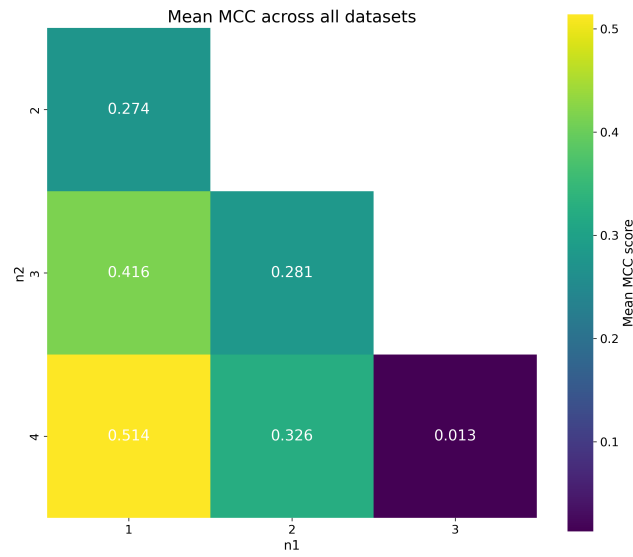
5 Conclusion and Future Work

This paper presents CANARI and introduces the near-anomaly concept. Near-anomalies are samples that, while not yet anomalous, lie close to the boundary of the distribution and are likely to transition into anomalies in the near future. CANARI is an unsupervised near-anomaly detection method based on the eCF that leverages theoretical properties of the CF distinguishing its growth with the parameter n for data in the support and data outside the support. CANARI is constructed with two eCFs with two different values for the parameter n from which the growth can be assessed. A threshold based on the Chebyshev

⁴ The implementation used for these two methods is proposed in the framework ChRISToffel Anomaly Locator available at <https://github.com/fgrivet/CRISTAL>

■ **Table 5** MCC for the 8 datasets D1 to D8 for different values of the parameter k .

k	D1	D2	D3	D4	D5	D6	D7	D8
1	0.743	0.342	0.539	0.418	0.293	0.442	0.318	0.677
$\sqrt{2}$	0.750	0.446	0.610	0.445	0.351	0.494	0.415	0.733
1.5	0.759	0.435	0.609	0.441	0.344	0.487	0.402	0.725
2	0.712	0.478	0.606	0.469	0.387	0.493	0.472	0.687
$2\sqrt{2}$	0.640	0.461	0.565	0.462	0.326	0.450	0.496	0.602
3	0.632	0.451	0.551	0.443	0.325	0.448	0.501	0.601
4	0.536	0.398	0.458	0.428	0.257	0.437	0.496	0.575
5	0.479	0.343	0.386	0.386	0.257	0.425	0.476	0.552
10	0.053	0.138	0.088	0.051	0.022	0.108	0.434	0.151



■ **Figure 7** Heatmap of the mean of the MCC on all the datasets for different values of n_1 and n_2 . The best combination is obtained for $n_1=1$ and $n_2=4$, which are the values that are used in CANARI.

inequality and on theoretical properties of the eCF is proposed to detect near-anomaly samples. In high-dimensional settings, CANARI can be coupled with CLOE [2] to reduce the data dimension.

To evaluate this method, industrial datasets composed of ICT from a PCB manufacturing process are considered. Synthetic datasets are created to simulate near-anomaly samples based on real industrial scenarios. Experiments confirm that CANARI detects near-anomaly samples more efficiently than a classical anomaly detector with double thresholds. Moreover, CANARI with the presented threshold achieves better performance than CANARI with other thresholds.

A sensitive analysis and an ablation study were conducted to assess the robustness of CANARI. The optimal values of parameter k are identified through this study. The selected combination of n_1 and n_2 corresponds to the one that yields the best average performance across all datasets. The variation in the amplitude used to generate the synthetic near-anomalies dataset confirms the stability of CANARI. Finally, CLOE is compared with two

■ **Table 6** MCC for the 3 datasets D1, D3, and D8 for different methods to reduce dimension before applying CANARI.

Method to reduce dimension	D1	D3	D8
$CANARI_{CLOE}$	0.76 ± 0.09	0.61 ± 0.17	0.73 ± 0.08
$CANARI_{KIC}$	0.56 ± 0.01	0.31 ± 0.09	0.69 ± 0.007
$CANARI_{UCF}$	0.31 ± 0.08	0.41 ± 0.11	0.29 ± 0.26

■ **Table 7** MCC for the 8 datasets D1 to D8 for different values of τ_{synth} .

τ_{synth}	D1	D2	D3	D4	D5	D6	D7	D8
0.1	0.61	0.33	0.69	0.34	0.40	0.45	0	0.57
10^{-2}	0.76	0.48	0.61	0.47	0.39	0.49	0.50	0.73
10^{-3}	0.62	0.34	0.73	0.34	0.48	0.47	0.59	0.58
10^{-4}	0.62	0.33	0.73	0.36	0.49	0.47	0.61	0.55

alternative methods for computing the eCF in high-dimensional dataset. The performance obtained by the two other approaches support the choice of CLOE made in this paper. Moreover, the two alternative methods fail to compute the ratio on five datasets because of very small eCF values.

The CANARI method can be adapted to any quality process with many tests with defined limits. These quality processes are commonly used in industry, where thresholds are defined on the basis of physical models. Such physical models fail to consider the production distribution and consequently reduce the ability to detect other problems. It can be used to anticipate future failures in products that initially look normal but can have some invisible defects and cause issues at a later step.

References

- 1 Armin Askari, Forest Yang, and Laurent El Ghaoui. Kernel-based outlier detection using the inverse christoffel function. *arXiv preprint arXiv:1806.06775*, 2018. [arXiv:1806.06775](#).
- 2 Léa Billet, Louise Travé-Massuyès, Elodie Chanthery, and Alexandre Gaffet. Cloe: Christoffel loss autoencoder for anomaly detection. *The 16th International Conference on Information Science and Technology*, 2026.
- 3 Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):6, 2020.
- 4 Kévin Ducharlet, Louise Travé-Massuyès, Jean-Bernard Lasserre, Marie-Véronique Le Lann, and Youssef Miloudi. Leveraging the christoffel function for outlier detection in data streams. *International Journal of Data Science and Analytics*, 20(3):1–17, 2024.
- 5 Florian Grivet, Didier Henrion, Jean-Bernard Lasserre, and Louise Travé-Massuyès. Scalable anomaly detection via a univariate christoffel function. *arXiv preprint arXiv:2606.12483*, 2026. [doi:10.48550/arXiv.2606.12483](#).
- 6 Jean B Lasserre and Edouard Pauwels. The empirical christoffel function with applications in data analysis. *Advances in Computational Mathematics*, 45(3):1439–1468, 2019. [doi:10.1007/S10444-019-09673-1](#).
- 7 Jean Bernard Lasserre, Edouard Pauwels, and Mihai Putinar. *The Christoffel–Darboux kernel for data analysis*, volume 38. Cambridge University Press, 2022.

- 8 Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.
- 9 Hossein Mirzaei, Mohammadreza Salehi, Sajjad Shahabi, Efstratios Gavves, Cees GM Snoek, Mohammad Sabokrou, and Mohammad Hossein Rohban. Fake it until you make it: Towards accurate near-distribution novelty detection. In *The eleventh international conference on learning representations*, 2022.
- 10 Friedrich Pukelsheim. The three sigma rule. *The American Statistician*, 48(2):88–91, 1994.
- 11 Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021. doi:10.1109/JPROC.2021.3052449.
- 12 Durgesh Samariya and Amit Thakkar. A comprehensive survey of anomaly detection algorithms. *Annals of Data Science*, 10(3):829–850, 2023.
- 13 Luc PJ Sträter, Mohammadreza Salehi, Efstratios Gavves, Cees GM Snoek, and Yuki M Asano. Generalad: Anomaly detection across domains by attending to distorted features. In *European conference on computer vision*, pages 448–465. Springer, 2024.

FELIX: Model Repair in Numeric Planning

Nir Aharoni 

Ben-Gurion University of the Negev, Beer Sheva, Israel

Yarin Benyamin 

Ben-Gurion University of the Negev, Beer Sheva, Israel

Shiwali Mohan 

Future Concepts Division, SRI International, Menlo Park, CA, USA

Roni Stern 

Ben-Gurion University of the Negev, Israel

Abstract

Automated planning algorithms rely on having a sufficiently accurate domain model to solve planning problems, particularly in complex environments with numeric effects. In real-world settings, however, domain models can become misaligned with the environment over time, i.e., no longer accurately reflect the environment. Even a minor drift in numeric effects can lead to faulty plans and execution failures. In this work, we address the problem of repairing numeric planning domain models by proposing FELIX, a learning-based repair mechanism that restores planning effectiveness by identifying and correcting outdated numeric effects. Rather than re-learning the full environment dynamics from scratch, our approach updates only the affected components of the domain model, enabling efficient adaptation with limited data. FELIX restores planning performance across diverse numeric domains, even under complex model drift. It uniquely handles both nonlinear effect shifts and structural signature changes while maintaining polynomial repair overhead under bounded parameter-type assumptions.

2012 ACM Subject Classification Computing methodologies → Knowledge representation and reasoning

Keywords and phrases Automated diagnosis, learning action models, model repair

Digital Object Identifier 10.4230/OASICS.DX.2026.5

Funding *Roni Stern*: Israel Science Foundation (ISF) #1238/23, Israel Innovation Authority #90011.

1 Introduction

Automated planning is a fundamental challenge in Artificial Intelligence research in which an agent seeks a sequence of actions that transforms an initial state into a goal state. In classical planning, a state is represented using Boolean predicates, and actions have deterministic effects defined by preconditions and outcomes that add or remove these predicates. Numeric planning is an extension of classical planning. While classical planning involves reasoning over Boolean predicates that represent the presence or absence of certain facts in the world, numeric planning allows reasoning over numeric fluents – state variables that take numeric values. For example, instead of representing the quantity of a resource using multiple Boolean predicates (e.g., `has-item-1`, `has-item-2`, `has-item-3`), numeric planning can use a single fluent such as `(item-count)` to represent the total number directly. These fluents can be modified by actions through numeric effects such as incrementing, decrementing, or assigning values. This enables planning in domains involving resources, numeric locations, or measurable quantities. To generate valid and effective plans, both classical and numeric planning rely on an accurate domain model that describes the actions available to the agent, including their preconditions and effects. In numeric planning, the domain model must also specify how actions update numeric fluents, typically through parameterized numeric



© Nir Aharoni, Yarin Benyamin, Shiwali Mohan, and Roni Stern;
licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 5; pp. 5:1–5:18

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

expressions. This increases modeling difficulty, as it requires defining not only which fluents are affected by each action, but also the exact numeric transformations applied to them. Without a reliable model, the planner may generate plans that are invalid or infeasible when executed in the real world. As a result, there has been increasing interest in approaches that reduce manual modeling effort [14, 1, 8] even for numeric action models [23, 27].

However, modeling is rarely a one-time activity. Even models that are initially very accurate may become outdated over time when the environment changes, e.g., when new domain objects are introduced, or action effects are altered. This phenomenon is known as *model drift* in various contexts. Model drift necessitates *model repair* to maintain planning reliability. Most prior work on model repair addresses only classical planning; see [7] for a survey.

To the best of our knowledge, the only prior work on model repair for numeric planning is Hydra [21]. Hydra is an algorithmic framework for a planning-based agent that is designed to detect and adapt to model-environment mismatches. Hydra maintains an internal environment model, and if it detects model inaccuracies, it heuristically searches the space of possible model repairs. Hydra employs a predefined set of model transformation operators, called Model Manipulation Operators (MMOs), each representing a predefined modification to the domain model. For example, an MMO can be to increment or decrement by a fixed amount the value of a specific numeric constant, such as gravity or friction. Hydra repairs numeric models by performing a heuristic search over combinations of these MMOs. Thus, Hydra’s runtime grows exponentially with the number of MMOs that need to be applied. In addition, Hydra requires an external agent – a human operator – to manually specify the MMOs.

In this work, we focus on a specific type of model repair problem: changes in the numeric effects. For this problem, we propose FELIX (Find, Estimate, Learn, Improve, eXecute), a workflow for automatically detecting and repairing numeric effects from observed transitions. By collecting execution data across problem instances and fitting regression models to detected discrepancies, FELIX repairs drifted numeric effects without requiring manually engineered operators or combinatorial search.

Moreover, we aim to learn changes in action signatures rather than assuming they are fixed. A notable algorithm that addresses this challenge is called L1 [3]. While our work focuses on numeric changes, L1 focuses on predicates. The authors of L1 prove that inferring missing action parameters in a logical domain is a combinatorial problem at least as hard as graph isomorphism. Consequently, L1 utilizes SAT encodings to “imagine” missing parameters and induce consistent parameter bindings globally - a process with exponential worst-case search time. In contrast, our approach distinguishes between two types of missing parameters. The first type is identified through discrepancies between the learned model and the environment, while the second is discovered via a systematic search over the different numeric parameter types present in the environment. Since each candidate is considered at most once, the discovery process runs in polynomial time.

Our analysis revealed a trade-off between the dimensionality of state representations and repair efficiency: smaller feature sets enable rapid adaptation from limited transition data, whereas richer representations incorporating higher-degree monomials are necessary to capture complex model-environment deviations. To address this trade-off, we employ multiple isolation strategies that vary in expressive power and allow the repair process to progressively increase representation complexity when needed. Empirical results demonstrate that this adaptive repair strategy captures complex forms of model drift while maintaining data efficiency when the drift is simple. Furthermore, we show that the approach automatically learns changes in action signatures, thereby enabling sustained planning performance in the presence of environmental drift.

2 Background

Numeric planning. addresses planning problems involving both discrete (Boolean) and continuous (numeric) variables. A widely used formalism to represent numeric planning problems is Planning Domain Definition Language (PDDL) 2.1 [10]. A numeric planning problem in PDDL2.1 is defined by a *domain* D and a *problem instance* Π . A domain is a tuple $D = \langle F, X, A \rangle$, where F is a set of Boolean variables, X is a set of numeric variables, and A is a set of actions. A state assigns values to all variables in $F \cup X$. Each action $a \in A$ is described by $\langle \text{name}(a), \text{pre}(a), \text{eff}(a) \rangle$. $\text{pre}(a)$ is the action’s preconditions, which are Boolean assignments and numeric constraints that must hold in state s for a to be applicable. $\text{eff}(a)$ is the action’s effects, which define the state transition that occurs when applying a . This set of preconditions and effects of all actions is referred to as the *action model*. A planning problem instance is specified by $\Pi = \langle I, G \rangle$, where I is the initial state and G is the goal condition. A solution, or *plan*, is a sequence of actions $\pi = \langle a_1, a_2, \dots, a_n \rangle$ that transforms I into a goal-satisfying state s_G . To improve scalability and abstraction, domains are often represented in a *lifted* form. A lifted action $a(\mathcal{V})$ is a schema where $\mathcal{V} = \{v_1, \dots, v_k\}$ is a set of parameters. A grounded action is produced by a substitution $\theta : \mathcal{V} \rightarrow O$, mapping parameters to specific domain objects O . The same definition applies to a lifted fluent $f(\mathcal{V})$ and its grounded counterpart. Lifted representations are essential for scaling to larger problems, as they allow for generalized planning without requiring the explicit grounding of the entire state space.

Action model learning. Manually creating domain models is difficult, slow, and error-prone [20], limiting scalability in real-world settings. Action model learning algorithms infer action models from observations, which are typically execution *trajectories*. The literature on action model learning is broad, covering approaches that learn symbolic models under limited observability [9, 19, 3], as well as methods designed to handle noise [17], safety constraints [15], conditional effects [25], online learning settings [16], and multi-agent settings [24]. Recent work on action model learning extends to numeric domains [27, 23, 22], integrates with Reinforcement Learning [28, 6], and even allows learning from visual inputs [2, 31, 32]. A less studied challenge of learning an action model is to learn the actions’ signature, i.e., their parameters. One of the few algorithms able to do so is L1 [3], which learns classical planning action models from traces without action parameters. [3] demonstrated that inferring these missing parameters is a combinatorial problem at least as hard as graph isomorphism. L1 utilizes SAT-encoding to “imagine” the missing action parameters required to make the planning model logically consistent, effectively outsourcing the combinatorial search and backtracking to the SAT solver.

Planning and Execution Monitoring. In model-based autonomous agents, generating an optimal or satisficing plan is only half the challenge; ensuring its robust delivery in dynamic or partially observable environments requires continuous plan execution monitoring. Early formalizations by [11] established the classic three-tiered monitoring pipeline: discrepancy detection (identifying variances between the model’s predicted state transitions and real-world sensor data), state estimation/diagnosis (determining the root cause of the anomaly), and plan evaluation (determining whether the encountered discrepancy actively violates the validity or optimality of the remaining plan suffix). An example of an architecture that implements this classical execution-monitoring pipeline is Hydra [21], which simulates expected trajectories to detect runtime anomalies.

In our solution, we build upon this execution-monitoring paradigm through the FELIX workflow, but fundamentally change how the agent leverages the initial monitoring phase in order to avoid exhaustive model-space search. Rather than relying on human-engineered search operators like Hydra, FELIX employs discrepancy detection as an informative data-filtering mechanism to decide precisely when to repair and how. At runtime, the agent executes planned actions and utilizes a simulator to generate expected next states. The discrepancy detection module triggers a repair iteration when an observed state variable deviates from predictions beyond a numerical tolerance parameter ϵ (i.e., $|\hat{s}'(y_i) - s'(y_i)| > \epsilon$). Crucially, the monitor determines how the repair should proceed by analyzing the exact subset of deviating state variables $\bar{y}$, lifting and filtering them to isolate only the broken components of the domain model for targeted polynomial regression and signature repair.

Model repair. A model repair algorithm accepts an initial, possibly flawed, domain model, and its task is to modify it according to some set of constraints. This modification may involve updating actions' preconditions and effects, adding or removing actions, or changing their signature. A recent survey on model repair [7] categorizes existing approaches to model repair according to the constraints they enforce: (1) *Solvability*: Repair the domain model so that a valid solution can be found. For example, [12] repairs models by adding missing effects but cannot delete effects or alter preconditions; (2) *Plan validity*: Repair the domain model to make specific plans valid or invalid as solutions. [18] addresses this through hitting set computation for minimal changes; (3) *Plan optimality*: Repair the domain model so that a specified plan becomes optimal; for instance, [13] reconciles models interactively with experts to align costs and optimality; and (4) *Solution-space properties*: Enforcing rules or constraints that every valid solution must satisfy. Environment design [33], for example, modifies the world to ensure all feasible plans are safe or distinguishable within physical limits. Our work falls under the category of Solvability. We assume a set of problems is solvable with a correct domain model, but the current domain model is no longer accurate, leading to execution failures.

Most prior work on this type of model repair has focused exclusively on classical planning. The only exception we are aware of is Hydra [21], a model-space learning framework that adapts to model drift by performing a heuristic-guided search over possible domain modifications to realign the agent's internal model with the environment. Hydra explores modifications via a predefined set of model transformation operators called Model Manipulation Operators (MMOs), each representing a predefined modification to the domain model.

3 Problem Setting and Methodology

We assume a planning-based agent operating in an environment that can be sufficiently modeled as a numeric planning domain $\langle F, X, A \rangle$ with an action model M^* . We assume the environment is deterministic. The agent does not have access to the real action model M^* . Instead, it has access to an action model M that may be different from M^* . The task is to repair M such that it sufficiently approximates the true domain M^* to allow solving planning problems in the domain.

Assumptions

The differences between the agent's model M and the real action model M^* are only in the effects alone, or in a combination of both the action signature and the effects. That is, the action preconditions in M and M^* are the same. Action effects in both M and M^* are

limited to polynomial combinations of the numeric state variables. We assume the agent can always observe its current state, including the values of all variables, both before and after each action is executed.

3.1 The FELIX Workflow

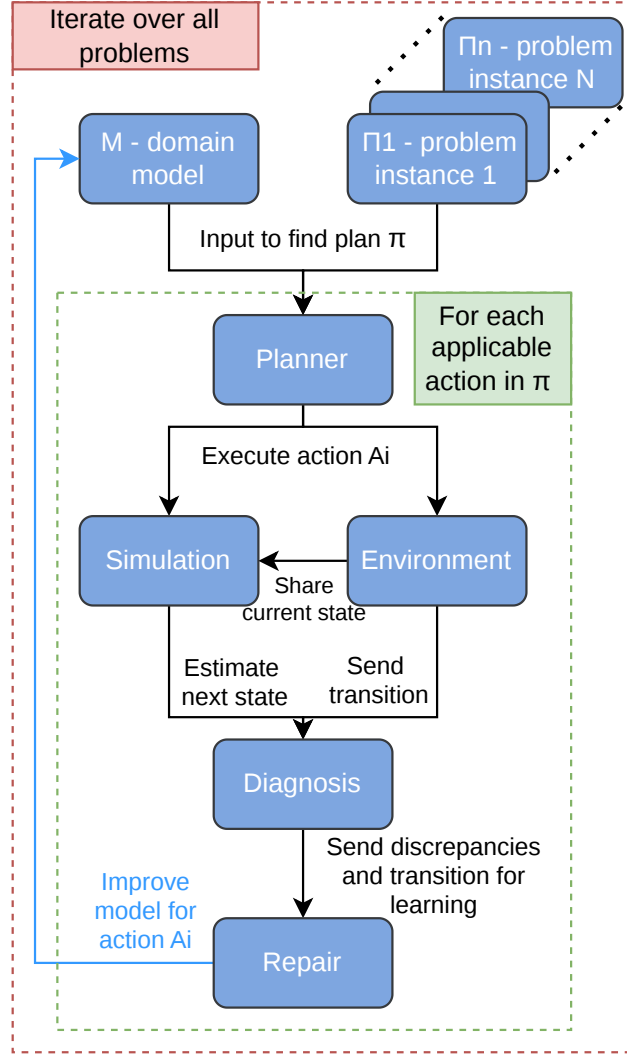
To be able to repair M , the agent must have some information about the (modified) environment. In our work, the agent collects such information by performing actions in the environment and observing their outcome. Specifically, we consider an agent that follows a workflow that we call FELIX (Find, Estimate, Learn, Improve, eXecute). FELIX is an iterative model-repair workflow. Throughout its execution, it maintains an incumbent action model M . Every iteration in FELIX begins with a planning problem $\Pi = \langle I, G \rangle$. Then, FELIX calls a domain-independent planner to synthesize a plan for the given problem using the current domain model, denoted by M . Once a plan is generated, the agent performs every planned action a in the real environment, recording the states visited immediately before and after every action. If an action is inapplicable, it is skipped. Note that the first action is always applicable, as we assume preconditions remain unchanged and only consider novelties in the effects. For every executed action a , FELIX also calls a PDDL simulator to generate the expected next state according to the (possibly incorrect) incumbent action model.

The observed next state s' and the expected next state $\hat{s}'$ are compared, and discrepancies between them are recorded. These discrepancies are diagnosed and passed to a model repair module that modifies the incumbent action model to reconcile these discrepancies. Our primary innovation lies in these diagnosis and repair modules. The complete FELIX workflow is illustrated in Figure 1. This workflow repeats across multiple problem instances, with repaired model components and collected transition data retained for subsequent iterations. Over time, these repairs allow the domain model to more accurately reflect the true behavior of the environment. Next, we describe in detail the key components in the FELIX workflow: diagnosis and repair.

3.2 Diagnosing Model Drift

To effectively repair the action model, the first step is to identify which actions yield outcomes that deviate from model predictions (Algorithm 1). This is done using existing methods for detecting discrepancies between the predicted and observed outcomes of actions [21]. Given an observed transition (s, a, s') obtained from executing the plan in the environment, we compare the predicted successor state $\hat{s}'$ with the observed successor state s' . If the deviation exceeds a numerical tolerance parameter ϵ , i.e., if there exists a state variable y_i such that $|\hat{s}'(y_i) - s'(y_i)| > \epsilon$, we mark action a as inconsistent with the current model.

Next, our method identifies the exact subset of state variables for which the environmental transition deviates from the model prediction. Let $\bar{y}$ be the set of state variables that differ between the predicted state $\hat{s}'$ and the observed state s' , i.e., $\bar{y} = \{y_i \mid |\hat{s}'(y_i) - s'(y_i)| > \epsilon\}$, where y_i ranges over all variables in the domain model. We filter the set of deviating state variables $\bar{y}$ to focus only on those that could be affected by the executed action a . Let $params(a)$ denote the set of parameters of action a , and let x_i be a grounded numeric fluent in the state. We define $\bar{x} = \{x_i \mid params(x_i) \subseteq params(a)\}$, so that $\bar{x}$ contains only those variables whose parameters are consistent with the action's signature and therefore could plausibly be used by a . To keep our dataset consistent, we convert $\langle \bar{y}, \bar{x} \rangle$ to its lifted form by replacing the grounded parameters $\bar{a}$ with the corresponding lifted parameters of the action.



■ **Figure 1** Design of FELIX.

Let $\langle \hat{y}, \hat{x} \rangle$ be the lifted version of $\langle \bar{y}, \bar{x} \rangle$. In each run, we save data pairs $\langle \hat{y}, \hat{x} \rangle$ for every action executed in the real environment. These pairs are collected in an active learning manner (as described in this section) and maintained in a list $\mathcal{D}_a$ for each lifted action $a \in \mathcal{A}$. The variables in $\hat{y}$ indicate the components of the model that produced incorrect predictions and are therefore targeted for relearning, guiding the updates during the model repair process. We illustrate this with an example in the following section.

Variable Isolation Approaches

Model drift may affect variables in ways that range from simple linear adjustments to single parameters to highly nonlinear transformations involving multiple interacting state variables. In such cases, standard linear regression is insufficient to isolate or identify the underlying numeric effects in $\hat{y}$, because the relationships between variables are no longer linear.

Algorithm 1 Resolving Model Drift.

```

1: Input: Plan  $\pi = [a_1, a_2, \dots, a_n]$ 
2: Input: Current domain model  $M$ 
3: Input: Real environment  $M'$ 
4: Input: Initial state  $s$ 
5: Output: Environment change dataset  $\{\mathcal{D}_a\}_{a \in \mathcal{A}}$ 
6: for each  $a \in \pi$  do
7:    $s' \leftarrow M'(s, a)$ 
8:    $\hat{s}' \leftarrow M(s, a)$ 
9:    $\bar{y} \leftarrow \{v \mid v \in \text{Vars}(s'), |\hat{s}'(v) - s'(v)| > \epsilon\}$ 
10:  if  $\bar{y} \neq \emptyset$  then
11:     $\bar{x} \leftarrow \text{Filter}(a, s)$ 
12:     $\langle \hat{y}, \hat{x} \rangle \leftarrow \text{Lift}(\langle \bar{y}, \bar{x} \rangle)$ 
13:     $\mathcal{D}_a \leftarrow \mathcal{D}_a \cup \{\langle \hat{y}, \hat{x} \rangle\}$ 
14:     $\text{Repair}(a, \mathcal{D}_a)$ 
15:  end if
16:   $s \leftarrow s'$ 
17: end for
18: return  $\{\mathcal{D}_a\}_{a \in \mathcal{A}}$ 

```

To address this, we propose methods to manipulate the state variables. These methods vary in the total number of variables they handle and in the complexity of the model drift they can resolve. To understand the methods, consider an environment $\mathcal{M}'$ with state variables $\{x_1, x_2, x_3\}$. Initially, we have an action a whose effect is defined as: $\{x_1 \leftarrow x_1 + 1\}$. Suppose we introduce a novelty by modifying the semantics of action a to instead be: $\{x_1 \leftarrow x_1 + x_2 + 1\}$.

After executing action a in state $s = \{x_1 = 1, x_2 = 1, x_3 = 1\}$, we observe the resulting state $s' = \{x_1 = 3, x_2 = 1, x_3 = 1\}$. According to the original model, we would expect $x_1 = 2$, but we observe $x_1 = 3$. This discrepancy indicates that a novelty has occurred in the semantics of action a .

We then repair the action's effect by using linear regression to predict x_1 from selected variables in the originating state s . In particular, we distinguish between *Changed Variables*, representing the set $\bar{y}$, and *All Variables*, representing the full set of state variables. We denote the changed-variable set for a given effect as $\{x_1\}$ and the complete state-variable set as $\{x_1, x_2, x_3\}$.

To better capture nonlinear numeric effects, we consider polynomial feature combinations (monomials) of the variables, such as x_i^2 or $x_i x_j$, rather than restricting the model to simple linear terms. The complexity of this representation is controlled by a degree parameter k , which limits the highest degree of monomial considered ($k = 1$ corresponds to the linear case). Later, we fit a linear model over these monomials, allowing the framework to represent nonlinear dependencies while keeping the learning problem linear in the parameters. We define *All Monomials of Degree $\leq k$* as the set of all monomial terms whose total degree does not exceed k . This generalizes the *All Variables* setting (corresponding to $k = 1$), which includes only linear terms, by additionally incorporating higher-order and interaction terms. In our example with three variables, we allow the set $\{x_1^{\alpha_1} x_2^{\alpha_2} x_3^{\alpha_3} \mid \alpha_1 + \alpha_2 + \alpha_3 \leq k, \alpha_i \geq 0\}$ to represent the space of monomials used to learn the new effect.

3.3 Repairing the Domain Model

3.3.1 Repair Strategy

Given the same scenario as described previously, we consider three repair strategies: *All Monomials*, *Adaptive Selection*, and *Adaptive Selection with Signature Repair*. Each strategy estimates the new numeric effects given a different set of rules.

All Monomials Repair

This strategy expands the state variables to include all monomials over the set of state variables up to degree k . In the running example with variables $\{x_1, x_2, x_3\}$ and $k = 2$, the learned regression model takes the form:

$$\begin{aligned} x'_1 = & m_1 x_1 + m_2 x_2 + m_3 x_3 \\ & + m_{12} x_1 x_2 + m_{13} x_1 x_3 + m_{23} x_2 x_3 \\ & + m_{11} x_1^2 + m_{22} x_2^2 + m_{33} x_3^2 + c \end{aligned}$$

Adaptive Selection Repair

This strategy dynamically selects the best model variant from a given set of candidates by maximizing the fit metric R^2 , which measures how well the model explains the observed data, with 1.0 indicating a perfect fit, with ties broken in favor of the model using fewer features. Given a degree parameter k , the candidate set will include linear models based on Changed Variables, All Variables, and All Monomials up to degree k . For example, when the true environmental dynamics are linear, if a model using a smaller subset of variables achieves a near-perfect fit, Adaptive Selection prefers this simpler representation over more complex polynomial alternatives, thereby reducing overfitting and improving interpretability.

3.3.2 Repairing the Action Signature

As discussed in 3.2, we filter out grounded fluents that contain parameters that do not appear in the action's current signature, as they cannot be expressed within that signature. If an environment change introduces a new parameter within the effects, the true effect of the action cannot be learned using the existing action signature. Consequently, we employ a repair mechanism to identify and integrate these missing parameters into the action signature.

We distinguish between two types of model drifts that introduce new parameters, each of which requires a different handling strategy. To illustrate these, consider a base action `increase(?c - counter)` with the effect: $g(?c) \leftarrow g(?c) + 1$, where g is a fluent that receives a counter.

Type 1: Affected

The first type of model drift adds an effect that changes the value of a fluent involving a new parameter. For example, consider the model drift: `increase(?c - counter ?a - modifier)  $\Rightarrow$   $\{g(?c) \leftarrow g(?c) + h(?a), h(?a) \leftarrow h(?a) + 1\}$` . In this scenario, the model drift introduces a parameter a , utilizes it within the fluent h (which receives a modifier), and the value of $h(?a)$ is **affected**. Following the execution of this action with grounded constants c and a , where a was selected by the environment without the agent's knowledge, we receive the observation $\bar{y} = \{g(c), h(a)\}$, as defined in Section 3.2. Using $\bar{y}$, we identify whether missing

parameters exist in our signature (see Algorithm 2). Our repair strategy iterates through all parameters in $\bar{y}$ (in this example, $\{a, c\}$). For any parameters not currently present in the action signature, their types are automatically added; the specific type is inferred from the fluent definition (in this example, $?a$ – modifier is inferred from h and is appended to the action signature).

■ **Algorithm 2** Type 1 missing parameter repair.

```

1: Input: Parameters of grounded action  $a_g - \mathcal{P}_{a_g}$ 
2: Input: Different functions -  $\bar{y}$  ▷ As described in 3.2
3:  $\mathcal{P}_{used} \leftarrow \text{ExtractParameters}(\bar{y})$  ▷ Extract parameters from functions
4: for each  $p \in \mathcal{P}_{used}$  do
5:   if  $p \notin \mathcal{P}_{a_g}$  then
6:      $\mathcal{P}_a \leftarrow \mathcal{P}_a \cup \{type(p)\}$  ▷ Add new parameter type to the action signature
7:   end if
8: end for
9: return  $model$ 

```

Type 2: Affects

The second type uses a fluent containing a new parameter to modify the value of an existing fluent. For example, the model drift $\text{increase}(?c - \text{counter } ?a - \text{modifier}) \Rightarrow g(?c) \leftarrow g(?c) + h(?a)$. Here, the parameter $?a$ is required to evaluate $h(?a)$, which in turn **affects** $g(?c)$. If the current repair model does not yet account for parameters of the type “modifier”, it will initially fail to create an accurate model for $g(?c)$.

Solving type 2

Unlike type 1, in type 2, the new parameter is not affected and only **affects** the deviating fluent. As a result, the discrepancy is observed in the affected fluent (in our example, $g(?c)$), while the underlying cause lies in the missing dependency on the fluent with the new parameter (in our example $h(?a)$). How can we repair it (see Algorithm 3):

1. Initial Attempt: We attempt to model $g(?c)$ using the existing parameters. With our most complex repair strategy (all monomials), we require $n + 1$ independent observations from $\mathcal{D}_a$.
2. Evaluation: If the model fails to adapt, defined by an R^2 value below $1 - 10^{-4}$, we conclude that the current action signature is insufficient.
3. The system then augments the action signature by adding new parameter types drawn from the available domain types, bounded by a maximum combination size l . This procedure is repeated, testing previously untried parameter type combinations in sequence, until a model achieving a satisfactory fit is obtained.

3.4 Theoretical Analysis

► **Theorem 1** (Expressiveness of polynomial regression for numeric effects). *Let f be any continuous function on a closed interval $[a, b]$. Then a regression model that represents numeric effects as a linear combination of monomials $1, x, x^2, \dots, x^k$ can approximate f arbitrarily well as $k \rightarrow \infty$, given sufficient data.*

Algorithm 3 Type 2 missing parameter repair.

```

1: Input: Parameter types of action  $a$  -  $\mathcal{P}_a$  ▷ according to our model
2: Input: Available domain parameter types -  $\mathcal{P}_{all}$ 
3: Input: Tried domain parameter types -  $\mathcal{P}_t$ 
4: Input: Environment change dataset -  $\mathcal{D}_a$ 
5:  $model \leftarrow \text{fit\_model}(\mathcal{D}_a)$ 
6:  $r^2 \leftarrow \text{evaluate}(\mathcal{D}_a, model)$ 
7: if  $r^2 < 1 - 10^{-4}$  and  $|\mathcal{D}_a| > |\mathcal{X}|$  then
8:    $\mathcal{P}_a \leftarrow \mathcal{P}_a \setminus \mathcal{P}_t$  ▷ Revert previous attempt
9:    $\mathcal{P}_{new} \leftarrow \text{next}(\mathcal{P}_{all} \setminus \mathcal{P}_a)$  ▷ Next set
10:   $\mathcal{P}_a \leftarrow \mathcal{P}_a \cup \mathcal{P}_{new}$  ▷ Add  $\mathcal{P}_{new}$  to a's signature
11:   $\mathcal{P}_t \leftarrow \mathcal{P}_{new}$  ▷ Set current try
12:   $\mathcal{D}_a \leftarrow \emptyset$  ▷ Reset dataset for new signature
13: end if
14: return  $model$ 

```

Proof. Assume our model includes all monomials $1, x, x^2, \dots, x^k$. Any polynomial of degree $m \leq k$ can then be written exactly as $p(x) = a_0 + a_1x + \dots + a_mx^m$ by choosing appropriate coefficients. Moreover, for any continuous function f on $[a, b]$ and any $\varepsilon > 0$, there exists a polynomial p such that $|f(x) - p(x)| < \varepsilon$ for all $x \in [a, b]$ by the Weierstrass Approximation Theorem [30]. Therefore, as $k \rightarrow \infty$ and with sufficient data to estimate coefficients, the model can exactly represent any polynomial effect and approximate any continuous effect to any desired degree of accuracy. ◀

Completeness

By iterating over candidate parameter types and updating the action signature, FELIX ensures that all relevant variables can be included in the effect model. Combined with monomial regression, this guarantees that the model can eventually represent any polynomial numeric effect up to a chosen degree k that depends on the correct set of parameters. The parameter-swap procedure explores candidate parameter combinations in a combinatorial fashion. When the number of candidate parameter types and the action arity are bounded, this search is polynomial in the input size. In the general case, however, the search over parameter combinations can be exponential. Note that while the algorithm achieves completeness for any arbitrary degree k , searching this unrestricted space inherently requires exponential time due to the combinatorial growth of the monomials.

Runtime Complexity

At the agent level, each iteration requires invoking an external planner, a problem class that is PSPACE-hard. However, the computational overhead incurred by the repair mechanism itself is polynomial. When dealing with a missing action parameter, we iterate over candidate domain parameter types and refit a model when necessary. Assuming the total number of available domain parameter types and the action arity are bounded, the number of iterations of the parameter-type swap loop is polynomial. Within each iteration, the algorithm performs model fitting and evaluation over the dataset $\mathcal{D}_a$. Under standard assumptions that model fitting and evaluation are polynomial in the size of the dataset and the number of parameter types, each iteration runs in polynomial time. Consequently, the additional runtime introduced by the parameter type swap and model fitting procedure is polynomial in the input size.

■ **Table 1** Size of Each Domain.

Domain	Lifted Actions	Lifted Fluents	Functions	Object Types
Expedition	4	2	3–4*	2–3*
Minecraft-pogo	6	0	7–8*	0–2*
Sailing	8	1	4	2
Drone	8	1	14–15*	1–2*

*Depends on model drift (e.g Lv.1 and Lv.2 novelties in Table 2).

4 Experimental Results

We conducted experiments on four domains: **Expedition**, **Sailing**, **Drone**, and **Minecraft-Pogo**. The first three domains are from the International Planning Competition (IPC) 2023 [29]. The Minecraft-Pogo domain is a recently introduced numeric planning domain in which an agent operates in the popular Minecraft game and is tasked with crafting a Pogo-Stick [4].¹ The size of each domain, in terms of lifted actions, lifted fluents, and functions, is listed in Table 1.

For each domain, we developed a custom problem generator to create random instances. These domains were selected to demonstrate the robustness and versatility of FELIX, including planning domains from established numeric planning benchmarks and recently introduced, less-studied planning problems.

Model Drift Complexity

We evaluate FELIX using five sets of model drift. The first three don’t introduce new parameters to the action, and are categorized by their difficulty as *Easy*, *Medium*, and *Hard*. These labels reflect the structural complexity of the model drift introduced: *Easy* scenarios involve linear shifts in existing action effects, while *Medium* and *Hard* sets introduce increasing degrees of non-linearity and latent dependencies. The other two sets, *Lv. 1* and *Lv. 2* introduce a new parameter to the action’s signature and effect. *Lv. 1* introduces a parameter of type 1 in the action, *Lv. 2* introduces a parameter of type 2 (see 3.3.2). Each set contains three distinct scenarios, for a total of 15 environment changes per domain. Examples of these cases are listed in Table 2, with a complete list provided in the supplementary material.

Baseline and Competing Algorithms

We evaluated All Monomials Repair, Adaptive Selection Repair, and Adaptive Selection with Signature Repair strategies. All methods were run with $k = 2$ (see Section 3.3.1) and with $l = 1$ configured for the signature repair combination limit (see Section 3.3.2). Performance was measured against *No-Repair* agent, which utilizes the original model without any mechanism to detect or address model-environment mismatches.

We did not compare FELIX directly with Hydra [21] because Hydra relies on manually engineered Model Manipulation Operators (MMOs) that define the search space of possible repairs. Constructing MMO sets capable of expressing the nonlinear polynomial drifts and

¹ Specifically, we use the fully numeric version of the environment, referred to as the “Item Counts” modeling of the problem. For more details, see [5].

■ **Table 2** Representative examples of model-environment mismatches in the Sailing, Expedition, and Minecraft-Pogo domains across three levels of drift complexity.

Domain	Sailing	Expedition	Minecraft-Pogo
Action	<i>go_x</i> (=direction)	<i>move_forward</i>	<i>get_log</i>
Original Effect	Moves toward x	Costs 1 supply	Gets 1 log, decreases tree count by 1
Easy	The actions <i>go_south_west</i> and <i>go_north_east</i> are switched	Costs 2 supplies	Gets 0.5 log, decreases tree count by 0.5
Medium	A person drifts by: $d = d + 0.3 \cdot \text{drift_factor}$	Costs $0.15 \cdot \text{sled_capacity} + 1$ supplies	Gets $0.05 \cdot \text{trees_in_map}$ logs, decreases same value from tree count
Hard	A person drifts by: $d = d + 0.001 \cdot d \cdot \text{drift_factor} + 0.1$	Costs $2 \cdot \text{factor}^2 + 1$ supplies. Here, factor is a fluent	Gets $0.05 \cdot \text{mine_factor}^2$ logs, decreases same value from tree count
Lv. 1	A person drifts. The parameter “person” isn’t originally in the action	Costs $3 \cdot \text{factor_value}$ supplies, <i>factor_value</i> is increased by 0.02. Uses new parameter “factor”	Gets $1 \cdot \text{axe_value}$, <i>axe_value</i> is reduced by 0.01. Uses new parameter “factor”
Lv. 2	The ship travel speed is based on the person’s location. The parameter “person” isn’t originally in the action	Costs $3 \cdot \text{factor_value}$ supplies. Uses the new parameter “factor”	Gets $1 \cdot \text{axe_value}$. Uses new parameter ‘axe’

signature modifications evaluated in our experiments would require substantial domain-specific engineering, and no standard benchmark MMO library currently exists for such settings. Consequently, a controlled empirical comparison across all evaluated drift scenarios is not feasible. We also did not compare against learning-from-scratch approaches because FELIX performs online repair while continuing to solve planning problems, whereas full relearning approaches typically assume offline training traces.

Implementation Details

For planning, we utilized Nyx, a Python-based, domain-independent PDDL+ planner [26]. While our current experiments focus on PDDL2.1 domains, Nyx was selected for its versatility; it provides a robust suite of search algorithms, including Greedy Best-First Search (GBFS) and Breadth-First Search (BFS), which were essential for our methodology. Furthermore, its Python-based implementation allowed for easier integration with the rest of our codebase, and its native support for PDDL+ facilitates a seamless transition to more complex temporal and continuous domains in future work. Notably, the planner’s capacity for PDDL+ does not impact the search performance or behavior for the PDDL2.1 domains used in this study. We configured Nyx to solve problems in each domain as follows. For Sailing, Drone, and Expedition, we used a custom, domain-specific heuristic combined with GBFS. For Minecraft-Pogo, we used a simple BFS. This was decided for faster planning, as all domains (except Minecraft-Pogo) were slow in planning. We preferred faster planning, as our focus is not on planning itself; it is on model repair.

A time limit of 60 seconds was imposed on the planner, after which we declared a failure. All experiments were conducted on a Windows 11 machine with Processor: 11th Gen Intel® Core™ i7-1165G7 @ 2.80GHz, 4 cores, 8 logical processors, and 32GB RAM.

Experimental Design

For each domain, we generated 50 problem instances. Each domain–drift combination was evaluated across these instances. We excluded problems where model drift rendered the task unsolvable for an oracle planner using the correct domain model.

For each repair algorithm, we iterated sequentially over the problem instances. For each problem, we performed the following procedure:

1. Run the planner using the current domain model to solve the problem.
2. Execute the resulting plan in the real environment and collect observations of any deviations. Notably, the environment maintains the autonomy to handle incomplete action calls; if an action is received with missing parameters, it completes the grounded action using the first available type-compatible parameter.
3. Record the outcome (success or failure; planner timeouts are counted as failures).
4. Apply the evaluated repair algorithm to update the domain model based on the collected observations.
5. Proceed to the next problem using the updated domain model.

4.1 Results

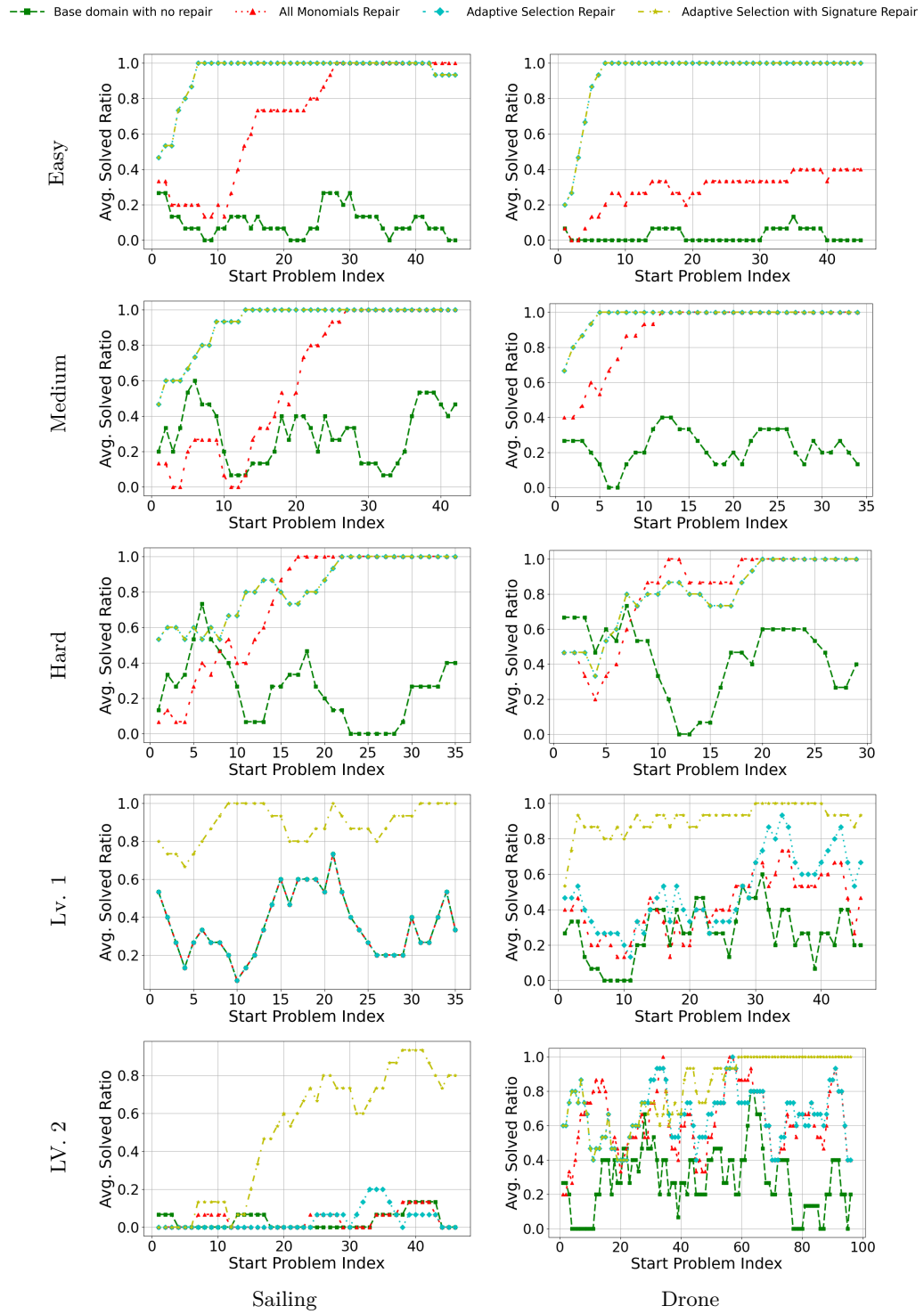
Figures 2 and 3 show our results across all domains and drift complexity levels. Each arranged in a 5x2 grid of plots, Where each column corresponds to a domain and each row corresponds to a specific level of model-environment mismatch (*Easy*, *Medium*, *Hard*, *Lv. 1*, and *Lv. 2*). In the *Easy*, *Medium*, and *Hard* settings, *Adaptive Selection Repair* is subsumed by *Adaptive Selection with Signature Repair*.

In each plot, the x-axis represents the problem index (excluding problems rendered unsolvable by any environment change within that complexity level). For each combination of a problem, a model drift instance, and a repair algorithm, the outcome is recorded as a binary value: 1 for success and 0 for failure. The y-axis displays the success rate based on these outcomes, calculated as a moving average with a window size of 5 and averaged over the three specific model-environment mismatches within that category. Thus, each plot illustrates the performance of each repair algorithm as the system encounters an increasingly diverse set of problem instances.

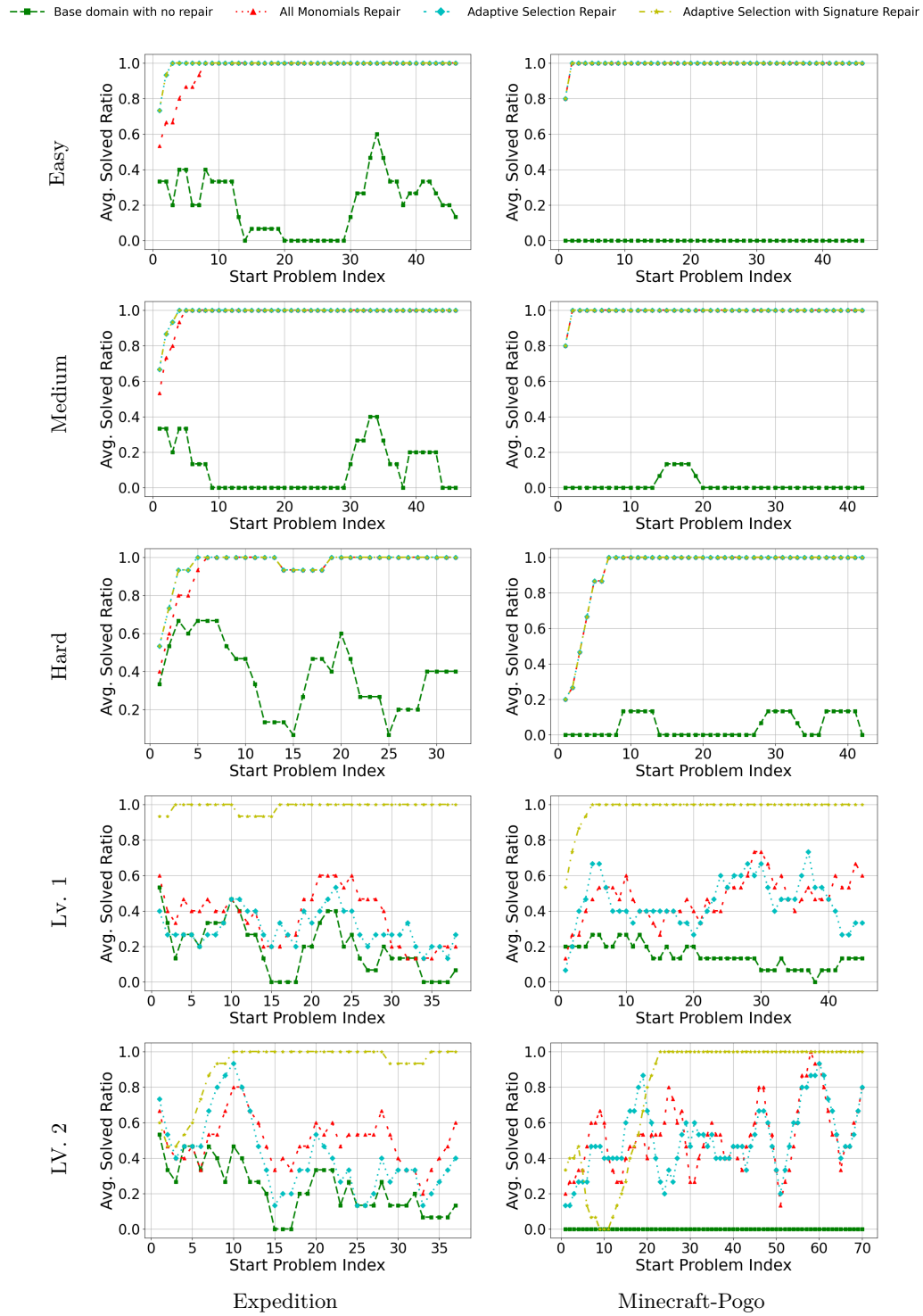
In the *Easy* and *Medium* sets, both *Adaptive* variants achieved near-perfect success almost immediately. In contrast, the *All Monomials* strategy lagged in the *Sailing* and *Drone* domains. This was particularly evident in the easy-drone graph. There, many actions produced by the model were inapplicable, hindering the data collection required for the *All Monomials* strategy, which needs more observations to adapt efficiently.

In the *Hard* set, all three strategies required more time to adjust due to the increased complexity. However, they eventually converged, reaching similar near-maximum success rates over time.

In the *Lv. 1* set, the *Adaptive Selection with Signature Repair* managed to adapt after a few problems. In contrast, the *Adaptive Selection* and *All Monomials* strategies did not adapt as effectively, though they still slightly outperformed the no-repair baseline in the *Drone*, *Minecraft-Pogo*, and *Expedition* domains.



■ **Figure 2** Evaluation results for the Sailing and Drone domains across Easy, Medium, Hard, Lv. 1, and Lv. 2 difficulties.



■ **Figure 3** Evaluation results for Expedition and Minecraft-Pogo domains across Easy, Medium, Hard, Lv. 1, and Lv. 2 difficulties.

In the Lv. 2 set, *Adaptive Selection with Signature Repair* required more problems to achieve adaptation than in Lv. 1. This was most noticeable in Minecraft-Pogo, where the system tried an incorrect parameter before finally using the correct one, and in Drone, where adaptation was delayed by slow data collection. Consequently, we evaluated those domains across 100 problems rather than 50, as *Adaptive with signature support* required more data to demonstrate adaptation. Across all domains, the *Adaptive Selection* and *All Monomials* strategies failed to adapt as effectively.

It is important to note that fully repairing the model does not always correlate with consistently solving problems. In many cases, a strategy solves problems before fully repairing the model by acquiring an incomplete numeric-effect function. In some instances, a strategy may have avoided an action due to an ineffective repair that made it less favorable during search, or simply because the action was not required.

Overall, the *Adaptive Selection with Signature Repair* method shows the most consistent adaptation across all model drift sets and domains. It performs comparably to the best repair in each case, combining the strengths of multiple strategies. In some domains and level 2 model drifts, *All Monomials* and *Adaptive Selection* outperform the no-repair baseline. This occurs, for example, in Drone, Expedition, and Minecraft-Pogo. This demonstrates that even without the correct signature, it is possible to approximate real-environment effects using the feature set available under the incorrect signature.

5 Conclusion and Future Work

In this work, we introduced FELIX, a learning-based framework for repairing numeric model drift in planning domains. By detecting discrepancies between predicted and observed transitions at runtime, FELIX selectively updates outdated numeric effects while preserving unaffected components of the domain model. This targeted repair mechanism enables agents to maintain planning performance without requiring full model relearning or explicit identification of the underlying environmental change.

Across multiple domains and levels of drift, FELIX consistently restored planning performance. In particular, the Adaptive Selection with Signature Repair strategy demonstrated rapid recovery under simple drift and steady improvement under more complex drift, highlighting the advantage of combining signature repair with data-driven effect learning.

For future work, we plan to improve FELIX by replacing the current brute-force parameter search with a type-guided heuristic, narrowing the candidate parameters for faster and more efficient repair. We also aim to enhance the adaptive repair strategy by integrating symbolic regression, allowing the framework to discover more complex and non-polynomial functional forms beyond the fixed monomial basis. More broadly, our research sits at the intersection of symbolic model repair, numeric planning, and reliable autonomy. By addressing model discrepancies at run-time, this work contributes to the broader goal of building resilient AI systems.

References

- 1 Ankuj Arora, Humbert Fiorino, Damien Pellier, Marc Etivier, and Sylvie Pesty. A review of learning planning action models. *Knowledge Engineering Review*, 33, 2018. doi:10.1017/S0269888918000188.
- 2 Masataro Asai, Hiroshi Kajino, Alex Fukunaga, and Christian Muise. Classical planning in deep latent space. *Journal of Artificial Intelligence Research*, 74:1599–1686, 2022. doi:10.1613/JAIR.1.13768.

- 3 Tomáš Balyo, Martin Suda, Lukáš Chrpá, Dominik Šafránek, Stephan Gocht, Filip Dvořák, Roman Barták, and G Michael Youngblood. Planning domain model acquisition from state traces without action parameters. In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 812–822, 2024.
- 4 Yarin Benyamin, Argaman Mordoch, Shahaf Shperberg, Wiktor Piotrowski, and Roni Stern. Crafting a pogo stick in minecraft with heuristic search. In *Proceedings of the International Symposium on Combinatorial Search*, volume 17, pages 261–262, 2024.
- 5 Yarin Benyamin, Argaman Mordoch, Shahaf S Shperberg, and Roni Stern. Model learning to solve minecraft tasks. In *PRL Workshop in ICAPS*, 2023.
- 6 Yarin Benyamin, Argaman Mordoch, Shahaf S Shperberg, and Roni Stern. Integrating reinforcement learning, action model learning, and numeric planning for tackling complex tasks. *arXiv preprint arXiv:2502.13006*, 2025. doi:10.48550/arXiv.2502.13006.
- 7 Pascal Bercher, Sarath Sreedharan, and Mauro Vallati. A survey on model repair in ai planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2025.
- 8 Ethan Callanan, Rebecca De Venezia, Victoria Armstrong, Alison Paredes, Tathagata Chakraborti, and Christian Muise. MACQ: A Holistic View of Model Acquisition Techniques. In *ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 2022.
- 9 Stephen N Cresswell, Thomas L McCluskey, and Margaret M West. Acquiring planning domain models using locm. *The Knowledge Engineering Review*, 28(2):195–213, 2013. doi:10.1017/S0269888912000422.
- 10 Maria Fox and Derek Long. Pddl2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003. doi:10.1613/JAIR.1129.
- 11 Christian Fritz. Execution monitoring – a survey. Technical report, University of Toronto, Department of Computer Science, 2005.
- 12 Alba Gragera, Raquel Fuentetaja, Ángel García-Olaya, and Fernando Fernández. A planning approach to repair domains with incomplete action effects. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 33, pages 153–161, 2023. doi:10.1609/ICAPS.V33I1.27190.
- 13 Sachin Grover, Sailik Sengupta, Tathagata Chakraborti, Aditya Prasad Mishra, and Subbarao Kambhampati. Radar: automated task planning for proactive decision support. *Human-Computer Interaction*, 35(5-6):387–412, 2020. doi:10.1080/07370024.2020.1726751.
- 14 Sergio Jiménez, Tomás De la Rosa, Susana Fernández, Fernando Fernández, and Daniel Borrajo. A review of machine learning for automated planning. *The Knowledge Engineering Review*, 27(4):433–467, 2012. doi:10.1017/S026988891200001X.
- 15 Brendan Juba, Hai S. Le, and Roni Stern. Safe learning of lifted action models. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 379–389, 2021. doi:10.24963/KR.2021/36.
- 16 Leonardo Lamanna, Alessandro Saetti, Luciano Serafini, Alfonso Gerevini, and Paolo Traverso. Online learning of action models for pddl planning. In *IJCAI*, pages 4112–4118, 2021. doi:10.24963/IJCAI.2021/566.
- 17 Leonardo Lamanna and Luciano Serafini. Action model learning from noisy traces: a probabilistic approach. In *ICAPS*, pages 342–350, 2024. doi:10.1609/ICAPS.V34I1.31493.
- 18 Songtuan Lin, Alban Grastien, Rahul Shome, and Pascal Bercher. Told you that will not work: Optimal corrections to planning domains using counter-example plans. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(25):26596–26604, April 2025. doi:10.1609/aaai.v39i25.34861.
- 19 Alan Lindsay, Jonathon Read, Joao F Ferreira, Thomas Hayton, Julie Porteous, and PJ Gregory. Framer: Planning models from natural language action descriptions. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2017.

- 20 Thomas Leo McCluskey, Tiago Stegun Vaquero, and Mauro Vallati. Engineering knowledge for automated planning: Towards a notion of quality. In *Proceedings of the Knowledge Capture Conference, K-CAP*, pages 14:1–14:8, 2017. doi:10.1145/3148011.3148012.
- 21 Shiwali Mohan, Wiktor Piotrowski, Roni Stern, Sachin Grover, Sookyung Kim, Jacob Le, Yoni Sher, and Johan de Kleer. A domain-independent agent architecture for adaptive operation in evolving open worlds. *Artificial Intelligence*, 334:104161, 2024. doi:10.1016/J.ARTINT.2024.104161.
- 22 Argaman Mordoch, Yarin Benyamin, Shahaf S Shperberg, Brendan Juba, and Roni Stern. Online learning of numeric action models for planning. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems*, pages 1491–1499, 2026.
- 23 Argaman Mordoch, Brendan Juba, and Roni Stern. Learning safe numeric action models. In *AAAI*, pages 12079–12086. AAAI Press, 2023. doi:10.1609/AAAI.V37I10.26424.
- 24 Argaman Mordoch, Ori Karat, Lea Shmilovich, Yarin Benyamin, Brendan Juba, and Roni Stern. Safe learning of multi-agent action models from concurrent joint action observations. *Journal of Artificial Intelligence Research*, 86, 2026. doi:10.1613/JAIR.1.20494.
- 25 Argaman Mordoch, Enrico Scala, Roni Stern, and Brendan Juba. Safe learning of pddl domains with conditional effects. In *International Conference on Automated Planning and Scheduling*, volume 34, pages 387–395, 2024. doi:10.1609/ICAPS.V34I1.31498.
- 26 Wiktor Piotrowski and Alexandre Perez. Real-world planning with pddl+ and beyond. *arXiv preprint arXiv:2402.11901*, 2024. doi:10.48550/arXiv.2402.11901.
- 27 José Á Segura-Muros, Raúl Pérez, and Juan Fernández-Olivares. Discovering relational and numerical expressions from plan traces for learning action models. *Applied Intelligence*, pages 1–17, 2021.
- 28 Sarath Sreedharan and Michael Katz. Optimistic exploration in reinforcement learning using symbolic model estimates. *Advances in Neural Information Processing Systems*, 36:34519–34535, 2023.
- 29 Ayal Taitler, Ron Alford, Joan Espasa, Gregor Behnke, Daniel Fišer, Michael Gimelfarb, Florian Pommerening, Scott Sanner, Enrico Scala, Dominik Schreiber, et al. The 2023 international planning competition, 2024.
- 30 K Weierstrass. *Über die analytische darstellbarkeit sogenannter willkürlicher functionen einer reellen veränderlichen*, sitzungsber, akad, 1885.
- 31 Kai Xi, Stephen Gould, and Sylvie Thiébaux. Neuro-symbolic learning of lifted action models from visual traces. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pages 653–662, 2024. doi:10.1609/ICAPS.V34I1.31528.
- 32 Kai Xi, Stephen Gould, and Sylvie Thiébaux. Learning lifted action models from unsupervised visual traces. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 36, pages 687–696, 2026.
- 33 Haoqi Zhang, Yiling Chen, and David Parkes. A general approach to environment design with one agent. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2002–2008, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

Geometric Feature Selection for Interpretable SVM Classifiers in Cyber-Physical Systems

Leonie Hatte 

LAAS-CNRS, Université de Toulouse, INSA, UPS, France

Pauline Ribot 

LAAS-CNRS, Université de Toulouse, INSA, UPS, France

Elodie Chanthery 

LAAS-CNRS, Université de Toulouse, INSA, UPS, France

Abstract

Identifying mode-transition conditions in Cyber-Physical Systems (CPS) is fundamental to model-based diagnosis, but learning them from scarce, multivariate time-series data requires boundary expressions that remain interpretable in physical variables. A central challenge is that some features are actively detrimental: their inclusion distorts the orientation of the learned boundary, degrading robustness to future crossings without affecting training accuracy. We introduce SVM-GRFE (Geometric Recursive Feature Elimination for SVM), a feature selection method that operates on polynomial monomials rather than raw variables, jointly assessing accuracy and geometric criteria that capture each monomial's structural role in boundary orientation; the boundary is back-projected analytically to an explicit polynomial inequality in physical variables. We prove SVM-GRFE returns an interpretable, geometrically stable, and robust boundary. On seven synthetic scenarios, SVM-GRFE achieves 33%–85% monomial reduction, matches or outperforms SVM-RFE in five out of seven scenarios and consistently outperforms SVM-RFE under noise at equal complexity, most notably under heteroscedastic noise.

2012 ACM Subject Classification Computing methodologies → Feature selection; Computing methodologies → Continuous models; Computing methodologies → Representation of mathematical functions; Computing methodologies → Representation of polynomials

Keywords and phrases Feature Selection, SVM, Geometric boundary learning, CPS

Digital Object Identifier 10.4230/OASICS.DX.2026.6

Supplementary Material *Software (Source Code)*: <https://gitlab.laas.fr/lhatte/dx-26-svm-grfe>, archived at `swh:1:dir:73c6aebeaad3cf0e2b4f896dd0a64ad80d74811b`

Acknowledgements *Use of AI-assisted tools* (Claude, Anthropic) as a writing assistance tool during the drafting and editing of this manuscript. All scientific content, mathematical claims, experimental results, and conclusions were produced, verified, and are the sole responsibility of the authors.

1 Introduction

Cyber-Physical Systems (CPS) combine continuous physical processes with discrete digital control, producing complex multivariate dynamics. In many industrial applications, from manufacturing plants to energy systems, the behavior of a CPS can be decomposed into a succession of operating modes, each characterized by its own continuous dynamics. Accurate identification of these modes and of the conditions governing transitions between them is a prerequisite for reliable model-based fault diagnosis and prognosis [3].

In model-based diagnosis, the behavioral model of the system is used to identify the root cause of observed anomalies by comparing system observations to expected mode behaviors. A central prerequisite is an accurate characterization of the conditions under which the system transitions between modes. These conditions must be *interpretable*: expressed as



© Leonie Hatte, Pauline Ribot, and Elodie Chanthery;
licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 6; pp. 6:1–6:20

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

explicit inequalities in the original physical variables of the system, directly readable by a domain expert and reusable without further computation in a monitoring, diagnostic, or control pipeline. Interpretability is not merely a convenience in this context, it is a diagnostic requirement. An interpretable condition identifies which physical variables govern the transition, enabling future fault isolation and reducing the diagnostic hypothesis space. A condition that involves spurious variables may incorrectly implicate sensors that are not causally linked to the transition.

Consider an aircraft during climb. If a learning algorithm infers a transition condition involving an irrelevant sensor (the outside temperature for example), the classifier may achieve excellent training accuracy while producing a boundary that is sensitive to sensor noise or drift. Such errors can compromise the reliability of model-based diagnosis. Not all instrumented variables are causally involved in a given transition, and including those that are not introduces a subtler problem than mere redundancy: these variables are *detrimental* (Definition 2) in the sense that their inclusion pulls the learned boundary away from the true transition condition.

This geometric distortion reduces the robustness of the boundary to transition future crossings: a transition observed under noise, measurement drift, or slow variations in the continuous dynamics may no longer be correctly classified, causing spurious fault detections or missed transitions, reducing the resilience of the diagnostic pipeline. Detecting and removing *detrimental* features is therefore central to transition condition learning, and cannot be achieved by accuracy-based criteria alone: a *detrimental* feature may distort the boundary orientation without reducing classification accuracy on the training data, a distinction that purely accuracy-based selection criteria cannot capture.

In practice, a CPS is observed as a multivariate Time Series (TS) $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, where each observation $\mathbf{x}_i \in \mathbb{R}^m$ represents the simultaneous measurement of m physical variables at time step i . Recovering an interpretable, exportable, geometrically stable expression under perturbation from data is both a classification problem and a modeling challenge. The available data is scarce: only the two short time segments immediately before and after the transition are informative, typically comprising a few hundred observations, which renders cross-validation (CV) unreliable as a model¹ selection strategy [2]. These conditions are formalized as Assumptions (A1)–(A4) in Section 3. With so few points, each CV fold contains too few samples to be representative of the data distribution, and the resulting accuracy estimates exhibit high variance, making classifier comparison unreliable. Data points are furthermore temporally ordered and locally correlated.

Support Vector Machines (SVMs) are particularly well-suited to this problem: their decision boundary is explicit and can be exported as a polynomial inequality. They can represent complex transition conditions when combined with polynomial feature expansion, and their maximum-margin formulation provides a natural measure of boundary robustness [5]. However, standard SVM feature selection (FS) methods such as SVM-RFE [7] operate on raw input variables and do not account for the geometric properties of the boundary essential for transition condition generalization. Moreover, in a polynomial feature space, the relevant unit of selection is the monomial rather than the physical variable: a monomial may couple several physical variables simultaneously, and its removal eliminates a coupling term from the boundary expression rather than a physical variable.

This paper introduces SVM-GRFE (Geometric Recursive Feature Elimination for SVM), an FS method tailored to transition condition learning from TS data. SVM-GRFE extends the SVM-RFE framework to operate on polynomial monomials rather than on raw physical

¹ We call *model* the learned classifier.

variables, enabling the analytical back-projection of the learned classifier to an explicit polynomial inequality in physical variables. SVM-GRFE has three main proved properties: it returns an interpretable boundary, it is robust to bounded perturbations and ensures geometrical stability. Its central contribution is the detection and removal of *detrimental* monomials, that distort the boundary orientation without accuracy benefit, through geometric criteria that capture the structural role of each monomial in the learned boundary, a distinction that purely accuracy-based selection criteria cannot make. SVM-GRFE can therefore be integrated directly into model learning pipelines. The method is evaluated on synthetic boundary scenarios of increasing complexity against baseline FS methods, and its robustness to five types of measurement noise is assessed at equal model complexity.

The remainder of this paper is organized as follows. Section 2 presents background on SVMs and reviews related work on SVM-based FS methods and their limitations. Section 3 formalizes the problem and details the SVM-GRFE algorithm. Section 4 presents the experimental protocol, results, and noise robustness analysis over the case study. Section 5 concludes and outlines directions for future work.

2 Background and related work

2.1 Support Vector Machines

Support Vector Machines (SVMs) are supervised binary classifiers [5]: given a labeled training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \{-1, +1\}$, a hard-margin SVM seeks a boundary expression $\mathbf{w}^\top \mathbf{x} + b = 0$ that separates the two classes while maximizing the *margin*, defined as the distance between the boundary and the closest training points of each class. Here, $\mathbf{w} \in \mathbb{R}^m$ is the weight vector, which is normal to the boundary and determines its orientation in the learning feature space, and $b \in \mathbb{R}$ is the bias term, which controls its offset from the origin. These closest points are called *support vectors* and are the only observations that determine the boundary; all other training points are irrelevant to the boundary's position. Maximizing the margin is equivalent to minimizing $\|\mathbf{w}\|^2$, which yields a boundary that is as far as possible from both classes and is therefore robust to small perturbations in the data.

When the two classes are not perfectly separable, the hard-margin formulation has no solution. The soft-margin SVM [5] relaxes this constraint by allowing a controlled number of misclassifications, penalized by a regularization parameter $C \in \mathbb{R}_*^+$. A large value of C forces the classifier to minimize training errors at the risk of overfitting, while a small value of C tolerates more misclassifications in favor of a wider, more generalizable margin.

Linear SVMs produce linear decision boundaries, which is insufficient for the complex, non-linear transition conditions that arise in CPS. Non-linear SVMs with kernels can represent more complex boundaries, but their decision function $f(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$ is an implicit representation defined only via training-point evaluations in a high or infinite-dimensional feature space, not as a closed-form expression in physical variables; it therefore cannot be exported without the full training set and cannot be inspected by a domain expert, making the learned boundary unsuitable for a transition condition on the physical variables of the system. Several approaches have been proposed to recover interpretability from SVM models. ISVMFD (Interpretable SVM for Functional Data) [9], is an interpretable SVM for functional data that produces a function whose shape can be inspected to identify relevant time intervals. However, this framework operates in infinite-dimensional function spaces and relies on the functional structure of the input: it does not extend to multivariate TS and the learned boundary cannot be exported as an inequality in physical variables. pTsm-SVM [4] incorporates data-mined two-dimensional linear rules as inequality constraints into the SVM

optimization problem to enhance interpretability. The result, however, is a set of local rules used to explain individual predictions, not a global analytical boundary expression: the underlying decision function remains a kernel SVM that cannot be exported and reused directly as a boundary expression. In our context, an interpretable expression is a strict requirement: the learned boundary must be a polynomial inequality in physical variables, analytically exportable and directly reusable in a CPS diagnostic or prognostic pipeline.

We therefore adopt a linear SVM combined with an explicit polynomial feature expansion defined by the monomial set $\Phi = \{\varphi_1, \dots, \varphi_{N_{\text{poly}}}\}$ of degree d , where:

$$N_{\text{poly}} = \binom{m+d}{d} - 1 \quad (1)$$

is the number of monomials in the polynomial expansion of total degree at most d , excluding the constant bias term b , and each $\varphi_j : \mathbb{R}^m \rightarrow \mathbb{R}$ is a monomial of the polynomial feature expansion. We therefore create non-linear features from the physical variables of the system. The SVM then operates linearly in the expanded polynomial feature space, and the learned boundary in physical variables can be analytically recovered as a polynomial inequality, retaining the ability to represent non-linear transition conditions. Those properties are central to the interpretability objectives of SVM-GRFE (Section 3.3).

From now on, the *variables* refer to the physical variables of the system while *features* or *monomials* refer to the expanded polynomial features in $\mathbb{R}^{N_{\text{poly}}}$. Furthermore, variables in bold refer to vectors.

2.2 Related work on feature selection: methods and limitations

In the context of boundary learning for CPS, FS serves a purpose that goes beyond model simplification: a monomial may distort the orientation or translation of the learned boundary in physical space, yielding a boundary that is not robust to future crossings of the transition. FS is therefore motivated by three complementary objectives: (i) geometric stability: removing monomials whose inclusion rotates or translates the boundary without accuracy benefit; (ii) robustness: choosing the most relevant monomials to make the boundary less sensitive to noise and slow variations in the continuous dynamics; and (iii) interpretability defined as follows.

► **Definition 1** (Interpretability of the learned boundary). *The learned boundary $g_{\Phi}(\mathbf{x}) \geq 0$ is said to be interpretable if it can be expressed as a closed-form polynomial inequality in the physical variables $x_0, \dots, x_{m-1}$, directly readable by a domain expert and analytically exportable for reuse in a CPS modeling, monitoring, or diagnostic pipeline.*

The goal is not to reduce the number of features for its own sake, but to identify and remove monomials that are actively *detrimental* to the geometric quality of the learned boundary.

► **Definition 2** (Detrimental monomial). *A monomial $\varphi_j \in \Phi$ is said to be detrimental if its removal from the active feature set Φ does not degrade classification accuracy and simultaneously reduces the geometric distortion of the learned boundary.*

FS for SVM has been studied extensively, and two main families of methods are generally distinguished [6, 8]: filter approaches select features independently of the classifier while wrapper approaches evaluate feature subsets by directly measuring the classifier’s accuracy on held-out data. Filter approaches, which are agnostic to the classifier, are not considered further; the remainder of this section focuses on wrapper methods.

Wrapper approaches are known to outperform filter methods when the goal is to optimize the performance of a specific classifier rather than to rank features by generic statistical criteria [8]. Importantly, [8] shows that maximizing accuracy and selecting the most relevant features for a given problem are not equivalent objectives and can lead to different chosen subsets. They define as *relevant* a feature for which its removal can degrade the performance of the classifier. Relevance is further distinguished into three levels: a feature is *strongly relevant* if its removal always degrades the optimal classifier; *weakly relevant* if it is not strongly relevant but belongs to some optimal feature subset; and *irrelevant* if its presence never affects the optimal classifier regardless of the subset considered. Their work also establishes that strongly relevant, weakly relevant, and irrelevant features require different treatment, a taxonomy that motivates the multi-criteria categorization adopted in our method SVM-GRFE (see Section 3).

Among wrapper strategies, backward elimination starts from the full set and removes variables recursively, capturing interactions more naturally but at higher computational cost. Forward selection on the other hand starts from an empty feature set and adds variables one at a time, offering computational efficiency at the cost of missing feature interactions. Forward selection is not included as a primary baseline in our work for two reasons. First, the first monomial selected anchors the solution: any subsequent monomial is evaluated conditionally on this initial choice, which introduces a dependence that is particularly problematic when monomials are correlated, as is systematically the case in polynomial expansions. Second, and more fundamentally, forward selection cannot identify *detrimental* monomials since it only adds monomials that improve the current model and never considers removal.

A well-documented limitation of wrapper methods is their susceptibility to overfitting in the feature subset space, caused by repeated use of accuracy estimates [8]. This problem is exacerbated when the number of available samples is small: with so few points, each CV fold contains too few samples to be representative of the data distribution, and the resulting accuracy estimates exhibit high variance, making classification model comparison unreliable [2]. In our setting, the training data consists of a local neighborhood of size $n_{\text{train}} \in \mathbb{N}$, which may comprise as few as a few tens of points. CV is therefore not abandoned for computational reasons, but because it is statistically unreliable with so few data. Unlike a CV fold, which partitions a single observation pool so that train and test share the same recording and local correlations, our train/test split originate from two distinct crossings, recorded under potentially different operating conditions. X_{eval} is therefore an independent realization of the same physical phenomenon, not a held-out subset of the same recording: the resulting accuracy estimate measures generalization to a future crossing (the method's actual operational objective) and its variance reflects the system's natural between-crossing variability rather than the arbitrary choice of partition. SVM-RFE (Recursive Feature Elimination), introduced by [7] for the classification of cancerous tissues from very few samples but a high number of features, addresses FS for linear SVM by recursively eliminating the feature j whose squared weight w_j^2 is smallest, eliminating the feature that contributes least to the margin, and retraining the classifier after each removal. SVM-RFE was shown to automatically reduce redundancy while producing smaller and more accurate subsets. However, SVM-RFE was designed for the case where features coincide with input variables: the weight vector $\mathbf{w}$ has one component per physical variable, so ranking features is equivalent to ranking variables. In SVM-GRFE, by contrast, the SVM operates in an expanded polynomial feature space and $\mathbf{w}$ has one component per monomial of the polynomial feature space. Ranking monomials is therefore not equivalent to ranking physical variables. Furthermore, SVM-RFE relies solely on a weight-based criterion, without accounting for the geometric properties (orientation and translation) of the learned boundary in physical space.

More recent work has explored metaheuristic approaches to FS, combining population-based optimization algorithms (Grey Wolf Optimizer (GWO), Salp Swarm Algorithm (SSA), Particle Swarm Optimization (PSO), and others) with classifiers such as SVM, random forests, or neural networks [10]. These methods identify optimal feature subsets jointly with classifier hyperparameters, evaluated by CV and tuned by Optuna [1]. Their results confirm that FS does not degrade SVM accuracy and yields significant reductions in model complexity and improved interpretability. However, population-based methods require a large number of model evaluations and a reliable CV estimate, making them unsuitable for the small-sample constraint considered here.

Three limitations are shared across the reviewed methods and motivate the design of our new method SVM-GRFE. First, feature is systematically equated with input variable, whereas in a polynomial feature space the relevant unit of selection is the monomial. Second, no method incorporates geometric criteria to assess the structural role of a feature in the learned boundary, relying exclusively on accuracy-based scoring. Third, none of the reviewed methods require producing an exportable expression of the boundary in physical variables, which is a requirement in the context of transition learning for CPS.

3 SVM-GRFE: Geometric Recursive Feature Elimination for SVM

3.1 Problem formulation

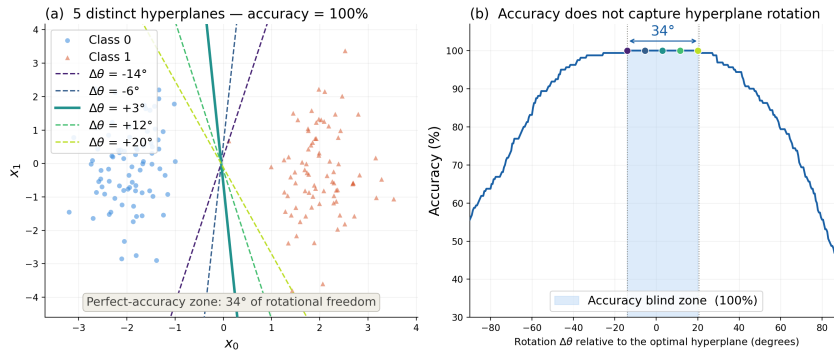


Figure 1 Geometric instability of the decision boundary: five different boundaries achieve perfect classification accuracy (subfig.(a)) while exhibiting substantial mutual rotation (subfig.(b)).

In CPS, the transition conditions between functioning modes are generally complex, non-linear and multivariate.

Accuracy alone is an insufficient criterion for FS when the geometry of the learned boundary matters. As exemplified in Fig. 1, subfig.(a), five distinct boundaries can achieve perfect classification accuracy on the same two-dimensional dataset while exhibiting substantial mutual rotation (subfig.(b)). In the context of boundary learning, such rotational instability yields structurally different expression in physical variables, undermining robustness of the learned boundary. A monomial that induces such rotation without improving accuracy is not merely useless: it is *detrimental* (Definition 2) and should be removed. We are looking for the monomials that ensure what is defined as the geometric stability of the learned boundary.

► **Definition 3** (Geometric stability of the learned boundary). *The learned boundary $g_\Phi(\mathbf{x}) \geq 0$ is said to be geometrically stable with respect to a monomial $\varphi_j \in \Phi$ if its removal does not significantly alter the orientation or the position of the boundary in physical space. More*

formally, let τ_θ^{low} and τ_ρ^{low} thresholds defined in Eq. 4. The learned boundary is geometrically stable iff $\theta^{(-j)} \leq \tau_\theta^{\text{low}}$ and $\rho^{(-j)} \leq \tau_\rho^{\text{low}}$, with $\theta^{(-j)}$ the angle between the orientation of the ablated boundary $g_{\Phi \setminus \{\varphi_j\}}(\mathbf{x})$ and the full boundary $g_\Phi(\mathbf{x})$, and $\rho^{(-j)}$ the normalized plane distance between the two boundaries.

Indeed, the decision boundary must not only classify the training dataset correctly, but also faithfully generalize to future crossings. A boundary that is accurate on the training dataset yet poorly oriented because of detrimental variables that would impact negatively the orientation of the boundary, may fail to generalize to future crossings. Not all features contribute equally to this geometric fidelity: some monomials are irrelevant and can be discarded without consequence, while others actively distort the orientation of the boundary and are therefore detrimental to the robustness of the learned boundary. Identifying and removing such detrimental monomials is the central objective of SVM-GRFE.

► **Definition 4 (Robustness of the learned boundary).** *The boundary condition $g_\Phi(\mathbf{x}) \geq 0$ is said to be robust at a point $\mathbf{x}$ if there exists a perturbation bound $\varepsilon(\mathbf{x}) > 0$ such that for any perturbed observation $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ satisfying $\|\boldsymbol{\delta}\| \leq \varepsilon(\mathbf{x})$, the sign of g_Φ is preserved: (i.e. $\text{sign}(g_\Phi(\tilde{\mathbf{x}})) = \text{sign}(g_\Phi(\mathbf{x}))$). Such perturbations may include measurement noise, sensor drift, and slow variations of the underlying continuous dynamics.*

The system is observed via a multivariate time series $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ with $\mathbf{x}_i \in \mathbb{R}^m$. An observable mode change occurs at a transition instant t^* . We extract from $\mathcal{X}$ a pre-transition training segment and a post-transition training segment. We denote by $\xi^- = \{\mathbf{x}_{-k^-}, \dots, \mathbf{x}_{-1}\}$ the last $k^- \in \mathbb{N}^*$ observations of $\mathcal{X}$ before t^* and by $\xi^+ = \{\mathbf{x}_1, \dots, \mathbf{x}_{k^+}\}$ the first $k^+ \in \mathbb{N}^*$ observations of $\mathcal{X}$ after and during t^* , with $n_{\text{train}} = k^- + k^+ = |X_{\text{train}}|$. The training dataset is constructed as:

$$X_{\text{train}} = \{(\mathbf{x}, -1) \mid \mathbf{x} \in \xi^-\} \cup \{(\mathbf{x}, +1) \mid \mathbf{x} \in \xi^+\} \quad (2)$$

so that class -1 corresponds to the pre-transition mode and class $+1$ to the post-transition mode. Once X_{train} is constructed, the temporal ordering of the points is discarded: the SVM treats X_{train} as an unordered set of labeled points in $\mathbb{R}^m$, whose only role is to determine the support vectors and the resulting decision boundary.

The method's design and guarantees are conditioned on the following assumptions, characteristic of transition learning from CPS multivariate time series.

- **(A1) Data scarcity.** The training dataset X_{train} comprises n_{train} labeled observations drawn from a single transition crossing. This assumption renders cross-validation unreliable as a model selection strategy: with so few points, each CV fold contains too few samples to be representative of the data distribution, and the resulting accuracy estimates exhibit high variance, making classifier comparison unreliable [2].
- **(A2) Held-out crossing availability.** At least two crossings of the same transition are available, yielding a segment X_{eval} disjoint from X_{train} and recorded at a different time instant. When no such crossing is available, $X_{\text{eval}} = X_{\text{train}}$ is used as a fallback, at the cost of measuring training fit rather than generalization.
- **(A3) Transition stationarity.** The guard condition governing the transition is stable across crossings: X_{eval} is a representative evaluation of the generalization of the learned boundary to future crossings of the same transition.
- **(A4) Bounded model complexity.** The polynomial degree $d^* \leq d_{\text{max}}$ and the number of physical variables m are moderate, consistent with physically interpretable CPS dynamics.

3.2 Proposed method

SVM-GRFE is a sequential pipeline: first, a full-feature boundary is learned, and then monomials are recursively eliminated according to accuracy and geometric criteria. The central building block throughout is the `train_SVM` function (Alg. 2, Section 3.2.2), which trains a polynomial SVM with hyperparameters (d^*, C^*) (previously identified by grid search, results in the associated git repository²), and exports the resulting boundary as an explicit polynomial inequality in physical variables. The objective of SVM-GRFE is therefore not to reduce the number of physical variables, but to identify a reduced subset $\Phi^* \subseteq \Phi$ by eliminating monomials whose presence degrades the geometric stability of the boundary, such that $g_{\Phi^*}(\mathbf{x}) \geq 0$. We first describe the full SVM-GRFE algorithm in its two elimination phases, then the training SVM function.

3.2.1 SVM-GRFE full algorithm

Alg. 1 explains the Geometric Recursive Feature Elimination for SVM (SVM-GRFE). At first, a feature set Φ is initialized as the complete set of N_{poly} (Equation (1)) monomials of degree at most d^* (line 1). A full-feature boundary $g_{\Phi}(\mathbf{x})$ is then trained via `train_SVM` (Alg. 2, Section 3.2.2) and its accuracy $\text{acc}_{\text{eval}}^{(\text{full})}$ on the evaluation dataset X_{eval} is recorded as the reference accuracy (line 2). SVM-GRFE then proceeds in two phases.

Phase 1: recursive elimination. At each iteration, every remaining monomial $\varphi_j \in \Phi$ is individually removed and an ablated model $g_{\Phi \setminus \{\varphi_j\}}(\mathbf{x})$ is retrained on $\Phi \setminus \{\varphi_j\}$ (lines 5–9). Three scores are computed for each ablated model (line 8, Eq. 3): the signed accuracy variation $\Delta^{(-j)}$, the cosine angle $\theta^{(-j)}$ between the orientation of the ablated boundary $g_{\Phi \setminus \{\varphi_j\}}(\mathbf{x})$ and the full boundary $g_{\Phi}(\mathbf{x})$, and the normalized plane distance $\rho^{(-j)}$ between the two boundaries, characterizing respectively the accuracy impact, the rotational displacement, and the translational displacement induced by removing φ_j .

We now use the notation $\square^{(\text{ref})}$ when referencing the reference model and $\square_{\text{proj}(-j)}^{(\text{ref})}$ for the element of the reference model projected onto the subspace $\Phi \setminus \{\varphi_j\}$ by discarding the component corresponding to φ_j . The three scores are defined as:

$$\cos(\theta^{(-j)}) = \frac{\mathbf{w}^{(-j)} \cdot \mathbf{w}_{\text{proj}(-j)}^{(\text{ref})}}{\|\mathbf{w}^{(-j)}\| \|\mathbf{w}_{\text{proj}(-j)}^{(\text{ref})}\|}, \quad \rho^{(-j)} = \frac{|b^{(-j)} - b_{\text{proj}(-j)}^{(\text{ref})}|}{\|\mathbf{w}^{(-j)}\|}, \quad \Delta^{(-j)} = \text{acc}_{\text{eval}}^{(-j)} - \text{acc}_{\text{eval}}^{(\text{ref})} \quad (3)$$

A value of $\cos(\theta^{(-j)})$ close to 1 indicates that the boundary orientation is preserved; a small value indicates a significant rotation. The distance $\rho^{(-j)}$ measures the translational displacement of the boundary normalized by $\|\mathbf{w}^{(-j)}\|$.

The three scores are then compared to adaptive thresholds that partition the candidate monomials into elimination categories: ε_{acc} separates accuracy-neutral removals from accuracy-impactful ones, while $\tau_{\theta}^{\text{low/high}}$ and $\tau_{\rho}^{\text{low/high}}$ distinguish geometrically negligible, moderate, and significant displacements. All thresholds are computed from the empirical distributions of the scores across all candidate removals at the current iteration (line 10),

Algorithm 1 Geometric Recursive Feature Elimination for SVM (SVM-GRFE).

Input: $C^* \in \mathbb{R}_+^+$, $d^* \in \mathbb{N}_*$ ▷ SVM hyperparameters
 $X_{\text{train}} \in \mathbb{R}^{n_{\text{train}} \times m}$, $y_{\text{train}} \in \{-1, +1\}^{n_{\text{train}}}$ ▷ train set
 $X_{\text{eval}} \in \mathbb{R}^{n_{\text{eval}} \times m}$, $y_{\text{eval}} \in \{-1, +1\}^{n_{\text{eval}}}$ ▷ evaluation set
Output: $\Phi^* \subseteq \Phi$ ▷ reduced monomial set
 $g_{\Phi^*}(\mathbf{x}) \geq 0$ ▷ boundary in physical variables

- 1: $\Phi \leftarrow \{\varphi_1, \dots, \varphi_{N_{\text{poly}}}\}$, $N_{\text{poly}} = \binom{m+d^*}{d^*} - 1$ ▷ feature set
- 2: $g_{\Phi}(\mathbf{x}) \leftarrow \text{train_SVM}(\Phi)$ ▷ Alg. 2
- 3: $\text{acc}_{\text{eval}}^{(\text{ref})} \leftarrow \text{accuracy}(g_{\Phi}(\mathbf{x}), X_{\text{eval}}, y_{\text{eval}})$

Phase 1: recursive elimination

- 4: **while** $|\Phi| > 1$ **do**
- 5: **for each** $\varphi_j \in \Phi$ **do**
- 6: $g_{\Phi \setminus \{\varphi_j\}}(\mathbf{x}) \leftarrow \text{train_SVM}(\Phi \setminus \{\varphi_j\})$ ▷ Alg. 2
- 7: $\text{acc}_{\text{eval}}^{(-j)} \leftarrow \text{accuracy}(g_{\Phi \setminus \{\varphi_j\}}(\mathbf{x}), X_{\text{eval}}, y_{\text{eval}})$
- 8: Compute $\Delta^{(-j)}, \theta^{(-j)}, \rho^{(-j)}$ ▷ Eq. 3
- 9: **end for**
- 10: Compute $\varepsilon_{\text{acc}}, \tau_{\theta}^{\text{low}}, \tau_{\theta}^{\text{high}}, \tau_{\rho}^{\text{low}}, \tau_{\rho}^{\text{high}}$ ▷ Eq. 4
- 11: $\mathcal{D}_{\text{detrimental}} \leftarrow \{\varphi_j \in \Phi : \Delta^{(-j)} \geq +\varepsilon_{\text{acc}}\}$ ▷ set of detrimental features
- 12: $\mathcal{D}_{\text{geom}} \leftarrow \{\varphi_j \in \Phi \setminus \mathcal{D}_{\text{detrimental}} : (\theta^{(-j)} \geq \tau_{\theta}^{\text{high}} \text{ or } \rho^{(-j)} \geq \tau_{\rho}^{\text{high}}) \text{ and } \Delta^{(-j)} \geq -\varepsilon_{\text{acc}}\}$ ▷ set of geometry destabilizing features
- 13: **if** $\mathcal{D}_{\text{detrimental}} \cup \mathcal{D}_{\text{geom}} = \emptyset$ **then break**
- 14: $\varphi^* \leftarrow \begin{cases} \arg \max_{\varphi_j \in \mathcal{D}_{\text{detrimental}}} \Delta^{(-j)} & \text{if } \mathcal{D}_{\text{detrimental}} \neq \emptyset \\ \arg \max_{\varphi_j \in \mathcal{D}_{\text{geom}}} (\tilde{\theta}^{(-j)} + \tilde{\rho}^{(-j)}) & \text{otherwise} \end{cases}$ ▷ with normalized scores $\tilde{\theta}, \tilde{\rho}$
- 15: $\Phi \leftarrow \Phi \setminus \{\varphi^*\}$
- 16: $g_{\Phi}(\mathbf{x}) \leftarrow \text{train_SVM}(\Phi)$ ▷ Alg. 2
- 17: $\text{acc}_{\text{eval}}^{(\text{ref})} \leftarrow \text{accuracy}(g_{\Phi}(\mathbf{x}), X_{\text{eval}}, y_{\text{eval}})$
- 18: **end while**

Phase 2: batch removal of useless monomials

- 19: **if** $|\Phi| > 1$ **then**
- 20: $\mathcal{D}_{\text{useless}} \leftarrow \{\varphi_j \in \Phi : \theta^{(-j)} \leq \tau_{\theta}^{\text{low}} \text{ and } \rho^{(-j)} \leq \tau_{\rho}^{\text{low}} \text{ and } \Delta^{(-j)} \geq -\varepsilon_{\text{acc}}\}$ ▷ reuse last Phase 1 scores
- 21: **if** $\mathcal{D}_{\text{useless}} \neq \emptyset$ **then**
- 22: $\Phi \leftarrow \Phi \setminus \mathcal{D}_{\text{useless}}$
- 23: $g_{\Phi}(\mathbf{x}) \leftarrow \text{train_SVM}(\Phi)$ ▷ Alg. 2
- 24: **end if**
- 25: **end if**
- 26: **return** $\Phi^* \leftarrow \Phi$, $g_{\Phi^*}(\mathbf{x}) \geq 0$

using quantiles to avoid introducing additional hyperparameters:

$$\begin{aligned}
 \varepsilon_{\text{acc}} &= \max\left(\frac{1}{n_{\text{eval}}}, Q_{75}\left(\left\{\left|\Delta^{(-j)}\right|\right\}_{\varphi_j \in \Phi}\right)\right) \\
 \tau_{\theta}^{\text{low}} &= Q_{25}\left(\left\{\theta^{(-j)}\right\}_{\varphi_j \in \Phi}\right) & \tau_{\theta}^{\text{high}} &= Q_{75}\left(\left\{\theta^{(-j)}\right\}_{\varphi_j \in \Phi}\right) \\
 \tau_{\rho}^{\text{low}} &= Q_{25}\left(\left\{\rho^{(-j)}\right\}_{\varphi_j \in \Phi}\right) & \tau_{\rho}^{\text{high}} &= Q_{75}\left(\left\{\rho^{(-j)}\right\}_{\varphi_j \in \Phi}\right)
 \end{aligned} \tag{4}$$

The accuracy tolerance ε_{acc} is floored at $1/n_{\text{eval}}$, with n_{eval} the number of points of the evaluation dataset X_{eval} , the smallest observable accuracy difference on X_{eval} , to prevent a single misclassified point from being interpreted as a meaningful accuracy change and triggering elimination. All thresholds are data-driven and require no tuning.

Two elimination candidate sets are then constructed (lines 11–12).

► **Definition 5** (Accuracy-detrimental monomial). *A monomial $\varphi_j \in \Phi$ is accuracy-detrimental if its removal improves classification accuracy beyond the tolerance threshold: $\Delta^{(-j)} \geq +\varepsilon_{acc}$. The set of accuracy-detrimental monomials is $\mathcal{D}_{detrimental} = \{\varphi_j \in \Phi : \Delta^{(-j)} \geq +\varepsilon_{acc}\}$.*

► **Definition 6** (Geometrically destabilizing monomial). *A monomial $\varphi_j \in \Phi \setminus \mathcal{D}_{detrimental}$ is geometrically destabilizing if its presence distorts the boundary orientation without contributing to separability: its removal causes a significant displacement of the boundary, in orientation ($\theta^{(-j)} \geq \tau_{\theta}^{high}$) or in position ($\rho^{(-j)} \geq \tau_{\rho}^{high}$), without degrading classification accuracy ($\Delta^{(-j)} \geq -\varepsilon_{acc}$). Its elimination therefore improves the geometric stability of the learned boundary.*

$$\mathcal{D}_{geom} = \left\{ \varphi_j \in \Phi \setminus \mathcal{D}_{detrimental} : \left(\theta^{(-j)} \geq \tau_{\theta}^{high} \text{ or } \rho^{(-j)} \geq \tau_{\rho}^{high} \right) \text{ and } \Delta^{(-j)} \geq -\varepsilon_{acc} \right\}$$

If both $\mathcal{D}_{detrimental}$ and $\mathcal{D}_{geom}$ sets are empty, Phase 1 terminates (line 13). Otherwise, the most detrimental or destabilizing monomial $\varphi^* \in \mathcal{D}_{detrimental} \cup \mathcal{D}_{geom}$ is selected for elimination (line 14): accuracy-detrimental monomials are prioritized by $\Delta^{(-j)}$, and geometrically destabilizing monomials by the sum of their normalized scores $\tilde{\theta}^{(-j)} + \tilde{\rho}^{(-j)}$. The selected monomial φ^* is dropped (line 15), the SVM model is retrained (line 16), and the reference accuracy is updated (line 17).

X_{eval} denotes a held-out evaluation set used to assess accuracy during elimination. Two choices are possible depending on data availability:

- **Test set (recommended):** $X_{eval} = X_{test}$, constructed from one or more future crossings of the same transition: segments $\tilde{\xi}^-$ and $\tilde{\xi}^+$ recorded at a different time than those used to build X_{train} , such that $X_{train} \cap X_{test} = \emptyset$. This provides an unbiased estimate of the generalization of the learned condition to unseen crossings.
- **Training set (fallback):** $X_{eval} = X_{train}$ when no held-out future crossing is available. Accuracy score $\Delta^{(-j)}$ measures training fit rather than generalization, which may lead to less reliable elimination decisions.

In the experiments of Section 4.2, $X_{eval} = X_{test}$, using a fixed train/test split in place of CV, which is unreliable in the small-sample constraint considered here [2].

Phase 2: batch removal of useless monomials. Upon termination of Phase 1, the scores from its last iteration are reused directly to identify the set of useless monomials $\mathcal{D}_{useless}$ (line 20): monomials for which both $\theta^{(-j)}$ and $\rho^{(-j)}$ are below their respective low thresholds and whose removal does not degrade accuracy. These monomials contribute neither to the boundary orientation, its position, nor its accuracy, and are removed in a single batch (lines 21–24), followed by a single retraining (line 23).

This two-phase design eliminates, in Phase 1, monomials that are either accuracy-detrimental (Definition 5) or geometrically destabilizing (Definition 6), and removes in batch in Phase 2 those that are useless (neither accuracy-detrimental nor geometrically destabilizing). These three populations are necessarily disjoint (Proposition 12), ensuring that each monomial is processed by at most one elimination criterion.

The final reduced monomial set Φ^* and the corresponding boundary $g_{\Phi^*}(\mathbf{x}) \geq 0$ are returned (line 26). To ensure that all SVM models trained during SVM-GRFE are consistently oriented and therefore geometrically comparable, the sign of g_{Φ^*} is fixed by evaluating it at the centroid μ of X_{train} in physical space (details in Section 3.2.2 and Algorithm 2).

The monomials retained in Φ^* at the end of Phase 2 are those for which no elimination criterion was triggered: they are neither accuracy-detrimental, nor geometrically destabilizing, nor useless under the adaptive thresholds computed at their respective iteration. This does

not imply that retained monomials are *necessary* in a strict sense (that is, that their removal would necessarily degrade accuracy or geometry). Two sources prevent this stronger claim. First, the adaptive thresholds are defined relative to the score distributions of the current iteration: a monomial may be retained simply because other monomials exhibit larger geometric effects at that step, keeping it below the elimination threshold. Second, the sequential nature of the elimination introduces a path dependence: the scores $\theta^{(-j)}$ and $\rho^{(-j)}$ are recomputed at each iteration on the current model, so a monomial that was not eliminated may have been eliminated under a different elimination order. The set Φ^* should therefore be interpreted as a *sufficient* reduced feature set: one for which no monomial was identified as removable without violating the accuracy or geometry constraints of the method, rather than as a minimal or necessary one.

3.2.2 SVM training function

■ **Algorithm 2** `train_SVM`(S).

Input: $S = \{\gamma_1, \dots, \gamma_{|S|}\}$ ▷ feature set
 $X_{\text{train}} = \{\mathbf{x}_1, \dots, \mathbf{x}_{n_{\text{train}}}\}$, $X_{\text{train}} \in \mathbb{R}^{n_{\text{train}} \times m}$ ▷ train set
 $y_{\text{train}} \in \{-1, +1\}^{n_{\text{train}}}$ ▷ train classes
 $C^* \in \mathbb{R}_*^+$, $d^* \in \mathbb{N}_*$ ▷ SVM hyperparameters

Output: $g_S(\mathbf{x}) \geq 0$ ▷ boundary in physical variables on input feature set S

1: Compute $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^m$ ▷ mean and standard deviation of X_{train}
2: Compute $Z_{\text{train}} = \{\mathbf{z}_1, \dots, \mathbf{z}_{n_{\text{train}}}\}$, $Z_{\text{train}} \in \mathbb{R}^{n_{\text{train}} \times m}$ ▷ standardize X_{train} by $\boldsymbol{\mu}, \boldsymbol{\sigma}$
3: $\Psi_{\text{train}} \leftarrow [\gamma_j(\mathbf{z}_i)]_{i,j} \in \mathbb{R}^{n_{\text{train}} \times |S|}$, $i \in [1, m], j \in [1, |S|]$ ▷ matrix of polynomial expansion

Compute decision function: $f(\mathbf{z}) = \mathbf{w}_f^\top \Psi(\mathbf{z}) + b_f$ in standardized space

4: $(\mathbf{w}_f, b_f) \leftarrow \text{LinearSVC}(C^*, \Psi_{\text{train}}, y_{\text{train}})$, $\mathbf{w}_f \in \mathbb{R}^{|S|}$, $b_f \in \mathbb{R}$ ▷ learn boundary

Back-project to decision function $g_S(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ in physical variables

5: $g_S(\mathbf{x}) \leftarrow \text{backproject}(\mathbf{w}_f, b_f, \boldsymbol{\mu}, \boldsymbol{\sigma}, S)$ ▷ Eq. 5
6: **if** $g_S(\boldsymbol{\mu}) < 0$ **then** ▷ $\boldsymbol{\mu}$ is the centroid of X_{train} in physical space
7: $g_S(\mathbf{x}) \leftarrow -g_S(\mathbf{x})$ ▷ orient so that $g_S(\boldsymbol{\mu}) \geq 0$
8: **end if**
9: **return** $g_S(\mathbf{x}) \geq 0$

Algorithm 2 details the `train_SVM` function, which is called throughout SVM-GRFE. It takes as fixed inputs any feature set S , training dataset X_{train} and its associated classes $y_{\text{train}} \in \{-1, +1\}^{n_{\text{train}}}$, degree d^* , regularization parameter C^* ; and returns an oriented boundary $g_S(\mathbf{x}) \geq 0$ in physical variables.

The variable-wise mean $\boldsymbol{\mu} \in \mathbb{R}^m$ and standard deviation $\boldsymbol{\sigma} \in \mathbb{R}^m$ of X_{train} are first computed (line 1), and the dataset is standardized to zero mean and unit variance, producing $Z_{\text{train}} \in \mathbb{R}^{n_{\text{train}} \times m}$ (line 2). Standardization is necessary because the polynomial expansion would otherwise produce monomials with very different numerical magnitudes depending on the physical units of the variables, which would distort the SVM weight vector $\mathbf{w}$. The polynomial feature matrix $\Psi_{\text{train}} \in \mathbb{R}^{n_{\text{train}} \times |S|}$ is then constructed by evaluating each monomial $\gamma_j \in S$ on each standardized observation $\mathbf{z}_i \in Z_{\text{train}}$ (line 3): entry (i, j) of Ψ_{train} is $\gamma_j(\mathbf{z}_i)$. A `LinearSVC` (Linear Support Vector Classifier) with regularization parameter C^* is fitted on $(\Psi_{\text{train}}, y_{\text{train}})$, producing the weight vector $\mathbf{w}_f \in \mathbb{R}^{|S|}$ and intercept $b_f \in \mathbb{R}$ of the decision function $f(\mathbf{z}) = \mathbf{w}_f^\top \Psi(\mathbf{z}) + b_f$ in standardized space (line 4).

The $\text{backproject}(\mathbf{w}_f, b_f, \boldsymbol{\mu}, \boldsymbol{\sigma}, S)$ function (line 5) converts the SVM decision function $f(\mathbf{z}) = \mathbf{w}_f^\top \Psi(\mathbf{z}) + b_f$ defined in standardized space, into an explicit polynomial $g_S(\mathbf{x})$ in physical variables by applying the substitution $z_i = (x_i - \mu_i)/\sigma_i$ analytically to each monomial $\gamma_j \in S$:

$$g_S(\mathbf{x}) = \sum_{j=1}^{|S|} w_{f,j} \gamma_j \left(\frac{\mathbf{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \right) + b_f = \mathbf{w}^\top \mathbf{x} + b \quad (5)$$

where $\mathbf{w} \in \mathbb{R}^{|S|}$ and $b \in \mathbb{R}$ are the coefficients of the resulting polynomial in $\mathbf{x}$, obtained by expanding and collecting the terms of $\sum_j w_{f,j} \gamma_j \left(\frac{\mathbf{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \right) + b_f$, and are distinct from $\mathbf{w}_f \in \mathbb{R}^{|S|}$ and $b_f \in \mathbb{R}$, which are the SVM weights in the standardized polynomial feature space.

To ensure that all returned SVM models are consistently oriented and therefore geometrically comparable, the sign of g_S is fixed by evaluating it at the centroid $\boldsymbol{\mu}$ of X_{train} in physical space (Algorithm 2). If $g_S(\boldsymbol{\mu}) < 0$, the sign of g_S is flipped (line 7), so that class +1 always contains the centroid of X_{train} . The oriented boundary $g_S(\mathbf{x}) \geq 0$ is returned (line 9).

The resulting inequality $g_S(\mathbf{x}) \geq 0$ is a polynomial inequality in the original state variables that can be directly reused as a transition condition in a CPS modeling, monitoring, diagnostic, or control pipeline, and the retained monomials explicitly identify which physical variables and their interactions govern the mode transition, allowing a diagnostic expert to determine which sensors are causally involved and which are irrelevant, reducing the diagnostic hypothesis space.

3.3 Properties of SVM-GRFE

SVM-GRFE satisfies two key properties that jointly motivate its design: the learned boundary is *interpretable*, in that it is expressed as a closed-form polynomial inequality in physical variables, and *robust*, in that the maximum-margin formulation provides a formal guarantee of classification stability under bounded perturbations. These two properties are stated and proved below, followed by a discussion of how geometric stability criteria connect them in the feature selection procedure.

► **Proposition 7** (SVM-GRFE returns an interpretable learned boundary). *Let $\Phi^* \subseteq \Phi$ be the reduced monomial set returned by SVM-GRFE. The learned boundary $g_{\Phi^*}(\mathbf{x}) \geq 0$ is a polynomial inequality in the physical variables $x_0, \dots, x_{m-1}$, whose coefficients are recovered analytically from the SVM weight vector $\mathbf{w}$ via the back-projection procedure of Alg. 2 (line 5). It is directly human-readable and analytically exportable for reuse in a CPS model for diagnosis and prognosis.*

Proof. By construction of backproject (Eq. 5), the decision function $f(\mathbf{z}) = \mathbf{w}_f^\top \Psi(\mathbf{z}) + b_f$ defined in the standardized polynomial feature space is converted into an explicit polynomial $g_{\Phi^*}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ in physical variables by substituting $z_i = (x_i - \mu_i)/\sigma_i$ analytically into each monomial $\varphi_j \in \Phi^*$ and collecting terms. The result is a finite sum of monomials in $x_0, \dots, x_{m-1}$ with explicit real coefficients, readable without computational evaluation of kernel functions. ◀

► **Definition 8** (Physical-space margin). *Let $g_{\Phi}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ be the decision function of the SVM in physical space. The margin at a point $\mathbf{x} \in \mathbb{R}^m$ is defined as the perpendicular distance from $\mathbf{x}$ to the decision boundary $\{\mathbf{x}' \in \mathbb{R}^m \mid g_{\Phi}(\mathbf{x}') = 0\}$:*

$$d(\mathbf{x}) = \frac{|g_{\Phi}(\mathbf{x})|}{\|\mathbf{w}\|} \quad (6)$$

► **Proposition 9** (SVM-GRFE returns a boundary robust to bounded perturbations). *Let $\mathbf{x} \in \mathbb{R}^m$ be a correctly classified point. The boundary $g_\Phi(\mathbf{x}) \geq 0$ is robust at $\mathbf{x}$ with $\varepsilon(\mathbf{x}) = d(\mathbf{x}) = |g_\Phi(\mathbf{x})|/\|\mathbf{w}\|$.*

Proof. The perturbed decision function reads:

$$g_\Phi(\mathbf{x} + \boldsymbol{\delta}) = \mathbf{w}^\top(\mathbf{x} + \boldsymbol{\delta}) + b = g_\Phi(\mathbf{x}) + \mathbf{w}^\top \boldsymbol{\delta} \quad (7)$$

The predicted class is preserved if and only if $g_\Phi(\mathbf{x})$ and $g_\Phi(\mathbf{x} + \boldsymbol{\delta})$ have the same sign. By the Cauchy–Schwarz inequality, $|\mathbf{w}^\top \boldsymbol{\delta}| \leq \|\mathbf{w}\| \cdot \|\boldsymbol{\delta}\|$. If $\|\boldsymbol{\delta}\| < \gamma(\mathbf{x}) = |g_\Phi(\mathbf{x})|/\|\mathbf{w}\|$, then:

$$|\mathbf{w}^\top \boldsymbol{\delta}| \leq \|\mathbf{w}\| \cdot \|\boldsymbol{\delta}\| < \|\mathbf{w}\| \cdot \frac{|g_\Phi(\mathbf{x})|}{\|\mathbf{w}\|} = |g_\Phi(\mathbf{x})| \quad (8)$$

so $g_\Phi(\mathbf{x} + \boldsymbol{\delta})$ cannot change sign, and the predicted class is preserved. ◀

► **Remark 10.** Proposition 9 is stated in physical space and $\boldsymbol{\delta} \in \mathbb{R}^m$ is a perturbation in physical space representing measurement noise, sensor drift, or slow variations in the continuous dynamics of the system. The perturbation tolerance (the margin in physical space) $d(\mathbf{x})$ increases with the distance of $\mathbf{x}$ to the boundary and decreases with $\|\mathbf{w}\|$: a smaller weight vector norm yields a wider margin and greater robustness. Note that the SVM maximum-margin criterion maximizes the margin in the standardized polynomial feature space, not in physical space; the physical-space margin $d(\mathbf{x})$ is therefore an indicator of robustness in the physical variables, evaluated empirically in Section 4.4.

Proposition 9 shows that robustness is directly controlled by the physical-space margin $d(\mathbf{x}) = |g_\Phi(\mathbf{x})|/\|\mathbf{w}\|$. The maximum-margin formulation of the SVM maximizes the minimum margin over the training set, providing the largest possible robustness guarantee on the training dataset. Rather than maximizing the margin directly during FS, which would require solving a joint optimization problem, SVM-GRFE uses the angular deviation $\theta^{(-j)}$ and the normalized plane distance $\rho^{(-j)}$ as proxies for geometric stability: a monomial whose removal significantly rotates or translates the boundary is deemed structurally relevant, and its elimination is avoided unless it is accuracy-detrimental (Section 3.2).

Accuracy and robustness are therefore complementary but not equivalent objectives: a boundary that correctly classifies all points may nonetheless exhibit a small physical-space margin $d(\mathbf{x})$, and remain fragile to perturbations of moderate magnitude. This distinction motivates the joint accuracy-and-geometry criterion of SVM-GRFE, which preserves the margin by avoiding geometrically destabilizing eliminations even when they are accuracy-neutral.

► **Proposition 11** (SVM-GRFE per-step elimination guarantee). *At each iteration of Phase 1, the eliminated monomial φ^* satisfies at least one of the following conditions:*

1. $\Delta^{(-j^*)} \geq +\varepsilon_{acc}$: accuracy does not decrease, (accuracy-detrimental case)
 2. $\Delta^{(-j^*)} \geq -\varepsilon_{acc}$: accuracy decreases by at most ε_{acc} . (geometrically destabilizing case)
- In particular, no monomial whose removal simultaneously degrades accuracy beyond ε_{acc} and causes a significant geometric displacement is ever eliminated.*

Proof. By construction, $\varphi^* \in \mathcal{D}_{\text{detrimental}} \cup \mathcal{D}_{\text{geom}}$. If $\varphi^* \in \mathcal{D}_{\text{detrimental}}$, then $\Delta^{(-j^*)} \geq +\varepsilon_{acc}$ by definition, so 1. holds. If $\varphi^* \in \mathcal{D}_{\text{geom}} \setminus \mathcal{D}_{\text{detrimental}}$, then $\Delta^{(-j^*)} \geq -\varepsilon_{acc}$ by the membership condition of $\mathcal{D}_{\text{geom}}$, so 2. holds. Since any eliminated monomial belongs to one of these two sets, no monomial outside $\mathcal{D}_{\text{detrimental}} \cup \mathcal{D}_{\text{geom}}$ is ever removed. ◀

► **Proposition 12** (Disjointness of elimination sets). *The three elimination sets $\mathcal{D}_{\text{detrimental}}$, $\mathcal{D}_{\text{geom}}$, and $\mathcal{D}_{\text{useless}}$ are pairwise disjoint:*

$$\mathcal{D}_{\text{detrimental}} \cap \mathcal{D}_{\text{geom}} = \emptyset, \quad \mathcal{D}_{\text{detrimental}} \cap \mathcal{D}_{\text{useless}} = \emptyset, \quad \mathcal{D}_{\text{geom}} \cap \mathcal{D}_{\text{useless}} = \emptyset$$

Proof. $\mathcal{D}_{\text{detrimental}} \cap \mathcal{D}_{\text{geom}} = \emptyset$ follows directly from the definition of $\mathcal{D}_{\text{geom}}$, which requires $\varphi_j \in \Phi \setminus \mathcal{D}_{\text{detrimental}}$. $\mathcal{D}_{\text{geom}} \cap \mathcal{D}_{\text{useless}} = \emptyset$: a monomial in $\mathcal{D}_{\text{useless}}$ satisfies $\theta^{(-j)} \leq \tau_{\theta}^{\text{low}}$ and $\rho^{(-j)} \leq \tau_{\rho}^{\text{low}}$, while membership in $\mathcal{D}_{\text{geom}}$ requires $\theta^{(-j)} \geq \tau_{\theta}^{\text{high}}$ or $\rho^{(-j)} \geq \tau_{\rho}^{\text{high}}$. Since $\tau_{\theta}^{\text{low}} < \tau_{\theta}^{\text{high}}$ and $\tau_{\rho}^{\text{low}} < \tau_{\rho}^{\text{high}}$ by construction ($Q_{25} < Q_{75}$), no monomial can satisfy both conditions simultaneously. $\mathcal{D}_{\text{detrimental}} \cap \mathcal{D}_{\text{useless}} = \emptyset$: $\mathcal{D}_{\text{useless}}$ requires $\Delta^{(-j)} \geq -\varepsilon_{\text{acc}}$ while $\mathcal{D}_{\text{detrimental}}$ requires $\Delta^{(-j)} \geq +\varepsilon_{\text{acc}}$; any monomial satisfying the latter is eliminated in Phase 1 before Phase 2 is reached. ◀

► **Proposition 13** (SVM-GRFE global guarantee of accuracy). *The per-step guarantee ensures that no single elimination degrades accuracy by more than ε_{acc} , but does not bound the cumulative loss across all iterations, since ε_{acc} is recomputed at each step and individual losses may accumulate. The total accuracy variation is therefore evaluated empirically in Section 4.2.*

3.4 Computational complexity of SVM-GRFE

We derive the computational complexity of Alg. 1 in terms of the number of **LinearSVC** training calls, which dominate the total cost. Let m denote the number of physical variables, d^* the polynomial degree, n_{train} the training dataset size, and N_{poly} the initial number of monomials in Φ . A single **LinearSVC** training on n_{train} points with p features costs $\mathcal{O}(n_{\text{train}} \cdot p)$.

Phase 1: cost per step. At elimination step $s \geq 1$, the current monomial set Φ contains $p_s = N_{\text{poly}} - (s - 1)$ monomials. The inner loop (lines 5–9) trains one ablated model per candidate monomial: for each $\varphi_j \in \Phi$, **train_SVM** is called on $p_s - 1$ features, at cost $\mathcal{O}(n_{\text{train}} \cdot p_s)$. Since there are p_s candidates, the total cost of the inner loop at step s is:

$$C_s^{\text{inner}} = \mathcal{O}(n_{\text{train}} \cdot p_s) \times p_s = \mathcal{O}(n_{\text{train}} \cdot p_s^2) \quad (9)$$

After the inner loop, the selected monomial φ^* is eliminated and the model is retrained once on $p_s - 1$ features (line 16), at cost $\mathcal{O}(n_{\text{train}} \cdot p_s)$ which is dominated by C_s^{inner} . The total cost of step s is therefore $\mathcal{O}(n_{\text{train}} \cdot p_s^2)$.

Phase 1: total cost. Let $T \leq N_{\text{poly}} - 1$ be the number of elimination steps before the break condition at line 13. Summing the per-step costs:

$$C^{\text{Phase 1}} = \sum_{s=1}^T \mathcal{O}(n_{\text{train}} \cdot p_s^2) = \mathcal{O}\left(n_{\text{train}} \cdot \sum_{s=1}^T p_s^2\right) \quad (10)$$

Since $p_s \leq N_{\text{poly}}$ for all s , each term satisfies $p_s^2 \leq N_{\text{poly}}^2$, giving the upper bound:

$$C^{\text{Phase 1}} \leq \mathcal{O}(n_{\text{train}} \cdot N_{\text{poly}}^2 \cdot T) \quad (11)$$

Worst case. When $T = N_{\text{poly}} - 1$, every monomial is eliminated one by one. Substituting $p_s = N_{\text{poly}} - (s - 1)$:

$$\sum_{s=1}^{N_{\text{poly}}-1} p_s^2 = \sum_{p=2}^{N_{\text{poly}}} p^2 = \frac{N_{\text{poly}}(N_{\text{poly}} + 1)(2N_{\text{poly}} + 1)}{6} - 1 = \Theta(N_{\text{poly}}^3) \quad (12)$$

The worst-case complexity of Phase 1 is therefore $\Theta(n_{\text{train}} \cdot N_{\text{poly}}^3)$.

Phase 2: negligible cost. Phase 2 (lines 19–24) reuses the scores $\Delta^{(-j)}$, $\theta^{(-j)}$, $\rho^{(-j)}$ computed during the last iteration of Phase 1, without any additional `train_SVM` call. It incurs at most one retraining on the final reduced set $\Phi \setminus \mathcal{D}_{\text{useless}}$ at line 23, at cost $\mathcal{O}(n_{\text{train}} \cdot p_T)$. This is dominated by $C^{\text{Phase 1}}$ and does not affect the overall complexity.

Effective complexity. In practice, early stopping at line 13 terminates Phase 1 after a small number of steps $T \ll N_{\text{poly}}$, giving an effective cost of $\mathcal{O}(n_{\text{train}} \cdot N_{\text{poly}}^2)$. Since $N_{\text{poly}} = \mathcal{O}(m^{d^*})$ for fixed d^* , the method scales polynomially in the number of physical variables and remains tractable for the moderate dimensions ($m \leq 10$, $d^* \leq 3$) typical of physical system dynamics.

4 Case study

We evaluate SVM-GRFE on seven synthetic boundary scenarios² *B1* to *B7* of increasing complexity, ranging from univariate linear thresholds to multivariate polynomial boundaries in up to five physical variables.

Synthetic data are used deliberately: standard benchmark datasets do not satisfy the constraints specific to our problem (few samples, physically meaningful variables, complex continuous dynamics, and a decision boundary whose expression matters as much as its accuracy). Each scenario generates labeled trajectory data of $n_{\text{train}} \approx 300$ training points and $n_{\text{test}} \approx 200$ test points per transition, reflecting the small-data constraint that motivates our design choices. In all experiments, $X_{\text{eval}} = X_{\text{test}}$, a fixed held-out split disjoint from X_{train} , used in place of CV.

From a diagnostic perspective, the seven scenarios represent transition conditions of increasing complexity between operating modes of a CPS.

4.1 Baselines

Three methods are compared throughout this section:

- **FULL:** the unpruned SVM model trained on the full monomial set Φ , serving as the accuracy reference;
- **SVM-RFE:** standard SVM-RFE applied in the polynomial feature space, which eliminates monomials one by one according to their squared SVM weight w_j^2 , $j \in |\Phi|$;
- **SVM-GRFE:** the proposed method.

FULL retains all N_{poly} monomials and serves as the accuracy reference. **SVM-RFE** baseline is truncated to the same cardinality $|\Phi^*|$ as **SVM-GRFE**, enabling an iso-complexity comparison that isolates the effect of the selection criterion from the effect of model complexity.

² For each scenario, the associated git repository <https://gitlab.laas.fr/lhatte/dx-26-svm-grfe> provides the full true boundary expression, the number of physical variables, the true polynomial degree, the selected hyperparameters, and the dataset sizes; we summarize only the aggregate complexity trends in this article.

■ **Table 1** Comparison of FULL, SVM-GRFE, and SVM-RFE across all seven scenarios; best accuracy per scenario in **bold**.

Boundary	N_{poly}	True monomials	FULL	SVM-GRFE		SVM-RFE	
			Accuracy	Accuracy	Φ^*	Accuracy	Φ^*
$B1$	2	x_0, x_1	0.995	0.995	x_0	0.995	x_0
$B2$	5	x_0^2, x_1^2	0.990	0.930	x_1, x_0^2, x_1^2	0.990	x_0, x_1, x_0^2
$B3$	2	$x_0 x_1$	0.990	0.990	x_0	0.990	x_0
$B4$	3	x_0^2, x_1, x_2	0.994	0.982	x_0	0.988	x_1
$B5$	9	$x_0^3, x_0 x_1^2$	0.975	0.990	x_0, x_1^2, x_0^3	0.980	x_0, x_1, x_0^3
$B6$	3	$x_0 x_1, x_1 x_2, x_0^2$	0.994	0.994	x_1, x_2	0.994	x_1, x_2
$B7$	20	x_2	0.950	0.995	$x_2, x_2 x_3, x_3^2$	0.920	x_0, x_2, x_4

Three stopping criteria are commonly used in the literature for SVM-RFE: stopping at a fixed cardinality specified a priori; stopping when cross-validated accuracy begins to degrade; eliminating all but one feature to produce a complete ranking [7]. The second criterion requires reliable CV estimates, which are unavailable in our small-sample constraint. The third produces a single-monomial output that is systematically too aggressive for polynomial boundaries. We therefore adopt the first criterion, stopping SVM-RFE at the same monomial cardinality $|\Phi^*|$ as SVM-GRFE, which enables a fair comparison that isolates the effect of the selection criterion (geometric vs. weight-based) from the effect of model complexity. All methods use the same hyperparameters (d^*, C^*) selected by grid search on a fixed train/test split, ensuring that observed differences in accuracy and geometry are attributable solely to the feature selection strategy (see results in the git repository²).

4.2 SVM-GRFE results and comparison to baseline methods

Table 1 summarizes the results for all three methods (FULL, SVM-RFE, SVM-GRFE) across the seven scenarios. For each method we report the selected monomial set Φ^* and the test accuracy. The best accuracy per scenario is in **bold**.

For $B1$, $B3$, and $B6$, (scenarios with $d^* = 1$) SVM-GRFE and SVM-RFE converge to the same monomial set and produce identical boundaries with no accuracy loss relative to FULL. For $B3$ and $B6$, the true boundary involves degree-2 monomials absent from the candidate set since $d^* = 1$: both methods identify the best available linear approximation within the constrained feature space. These low-complexity scenarios confirm that the geometric criterion of SVM-GRFE does not introduce unnecessary eliminations when the feature space is small and well-conditioned.

For $B2$ and $B4$, SVM-RFE outperforms SVM-GRFE. In $B4$, SVM-GRFE retains x_0 (Accuracy = 0.982) while SVM-RFE retains x_1 (Accuracy = 0.988, boundary $8.738 x_1 - 4.247 \geq 0$), which is the more discriminative variable according to the unpruned model weights ($6.142 x_1 + 2.080 x_2 + 1.142 x_0 - 3.878 \geq 0$). In $B2$, SVM-GRFE incurs a difference in accuracy of -0.060 by eliminating x_0 , whose small but non-negligible weight ($0.307 x_1 - 0.089 x_0^2 - 0.032 x_0 + \dots \geq 0$) is underestimated by the geometric criterion. Both cases correspond to near-tie configurations where variables are individually informative and geometrically comparable: the weight-based criterion of SVM-RFE is better suited to these settings because it directly reflects discriminative power, whereas the geometric criterion of SVM-GRFE is designed to detect structural irrelevance rather than marginal accuracy differences.

For $B5$ and $B7$, the full model is severely degraded by detrimental monomials that carry large SVM weights, misleading the criterion of SVM-RFE. In $B5$, SVM-GRFE converges to $\{x_0, x_1^2, x_0^3\}$ (Accuracy = 0.990, +1.5% over FULL) by correctly identifying x_1

■ **Table 2** Geometric indicators θ , ρ , $1/\|\mathbf{w}\|$ for SVM-GRFE and SVM-RFE relative to FULL.

Boundary	FULL	SVM-GRFE			SVM-RFE		
	$\frac{1}{\ \mathbf{w}\ }$	θ	ρ	$\frac{1}{\ \mathbf{w}\ }$	θ	ρ	$\frac{1}{\ \mathbf{w}\ }$
<i>B1</i>	0.501	49.1°	1.26	0.726	49.1°	1.26	0.726
<i>B2</i>	3.113	24.3°	0.99	2.150	1.2°	0.08	3.217
<i>B3</i>	1.001	53.7°	0.70	1.093	53.7°	0.70	1.093
<i>B4</i>	0.152	80.0°	0.13	0.219	21.1°	0.04	0.114
<i>B5</i>	0.007	58.7°	0.89	0.074	72.6°	1.31	0.103
<i>B6</i>	0.234	26.5°	0.10	0.203	26.5°	0.10	0.203
<i>B7</i>	0.020	90.0°	6.21	1.412	65.5°	0.12	0.088

as detrimental. SVM-RFE retains x_1 instead of x_1^2 , achieving only Accuracy = 0.980. In *B7*, SVM-GRFE converges to $\{x_2, x_2x_3, x_3^2\}$, correctly discarding all x_4 -related monomials. SVM-RFE retains x_4 (Accuracy = 0.920) because its large weight $|w_{x_4}| = 45.37$ reflects its dominance in the distorted full model rather than its relevance to the true boundary. In both cases, the geometric criterion of SVM-GRFE detects that removing detrimental monomials simultaneously stabilizes the boundary orientation and improves accuracy, a distinction that the weight ranking of SVM-RFE cannot capture.

These results must be interpreted in the context of the selection strategy of d^* and C^* that are selected by grid search that maximize F_1 score on a fixed split, a criterion that is agnostic to boundary geometry and trajectory dynamics. This means that the selected hyperparameters are optimized for classification accuracy, not for the geometric robustness required.

Overall, SVM-GRFE matches or outperforms SVM-RFE in five out of seven scenarios, strictly outperforms the full model in two (*B5*, *B7*), and identifies compact, physically interpretable boundaries in the high-complexity constraint. This is the expected trade-off for a criterion designed to detect structural irrelevance rather than marginal accuracy differences. Its advantage is most pronounced when the monomial space is large, and accuracy-based criteria are misled by detrimental features with large but spurious SVM weights.

4.3 Geometric analysis of learned boundaries

Table 2 reports the geometric indicators of SVM-GRFE and SVM-RFE relative to FULL: the angular deviation θ (degrees), the scaled plane distance ρ which measures the translational displacement between the two boundaries in units of $\|\mathbf{w}\|$, and the SVM margin $1/\|\mathbf{w}\|$ in physical space. The margin quantifies the robustness of the learned boundary to perturbations in the feature space, as formalized in Section 3.3, Proposition 9: a larger margin implies a wider robustness guarantee around the boundary. θ and ρ are computed between the final selected boundary $g_{\Phi^*}(\mathbf{x})$ and the unpruned reference $g_{\Phi}(\mathbf{x})$ using Eq. 3, with $\Phi \setminus \{\varphi_j\}$ replaced by Φ^* .

For *B1*, *B3*, and *B6*, both methods produce geometrically identical boundaries (θ and ρ equal), consistent with their identical monomial selections. For *B4*, SVM-GRFE diverges strongly from FULL ($\theta = 80.0^\circ$) while SVM-RFE stays close ($\theta = 21.1^\circ$), consistent with their respective monomial selections. For *B2*, SVM-RFE stays very close to the full model ($\theta = 1.2^\circ$, $\rho = 0.08$) while SVM-GRFE diverges ($\theta = 24.3^\circ$, $\rho = 0.99$), consistent with SVM-GRFE's erroneous elimination of x_0 in this near-tie configuration. For *B5*, SVM-GRFE ($\theta = 58.7^\circ$) is closer to FULL than SVM-RFE ($\theta = 72.6^\circ$), yet achieves higher accuracy. For *B7*, SVM-GRFE reaches $\theta = 90^\circ$ and $\rho = 6.21$: the boundary is not only rotated but

■ **Table 3** Accuracy at $\varsigma = 0$ and $\varsigma = 0.50$ (averaged over all noise types).

Boundary	$\varsigma = 0$		$\varsigma = 0.50$		Δ_{acc}
	SVM-GRFE	SVM-RFE	SVM-GRFE	SVM-RFE	
	Accuracy	Accuracy	Accuracy	Accuracy	
$B1$	0.995	0.995	0.858	0.849	+0.009
$B2$	0.930	0.990	0.858	0.895	−0.037
$B3$	0.990	0.990	0.875	0.870	+0.005
$B4$	0.982	0.988	0.909	0.902	+0.007
$B5$	0.990	0.965	0.873	0.858	+0.015
$B6$	0.994	0.994	0.935	0.847	+0.088
$B7$	0.995	0.920	0.920	0.815	+0.105
Mean	0.982	0.977	0.890	0.862	+0.028

also substantially translated away from the distorted full model. SVM-RFE diverges less ($\theta = 65.5^\circ$, $\rho = 0.12$) but retains x_4 , achieving only Accuracy = 0.920. This confirms that proximity to the full model is not a reliable indicator of boundary quality in high-dimensional monomial spaces with detrimental features.

The margin column reveals a pattern complementary to accuracy. For FULL in $B5$ and $B7$, the margin collapses to extremely small values (0.007 and 0.020 respectively), indicating that the boundary is nearly tangent to the training points in the distorted polynomial space: any small perturbation of the physical trajectory is likely to cause a misclassification. SVM-GRFE in $B5$ obtains margins of 0.074 (10.6 times the margin of FULL) and 1.412 in $B7$ (70.6 times the margin of the full model), a dramatic improvement that is directly attributable to the elimination of detrimental monomials that dominated the weight vector $\mathbf{w}$ and inflated $\|\mathbf{w}\|$. This confirms the theoretical connection established in Section 3.1: a boundary that is accurate on training data but with a small margin will fail under noise or sensor drift, whereas SVM-GRFE’s geometric criterion explicitly favors boundaries with larger margins by penalizing the elimination of monomials that would destabilize the weight vector. SVM-RFE, by contrast, retains x_4 in $B7$ and achieves a margin of only 0.088, only marginally better than the full model (0.020). For $B4$, SVM-GRFE achieves a margin of 0.219 versus 0.114 for SVM-RFE and 0.152 for the full model, suggesting that despite its lower accuracy, the SVM-GRFE boundary in $B4$ is more robust to future perturbations. For $B1$, $B3$, and $B6$, margins are comparable across methods, consistent with the identical boundaries produced.

4.4 Robustness to measurement noise

To evaluate whether the geometric criterion of SVM-GRFE produces boundaries that are more robust than those of SVM-RFE under realistic perturbations, we evaluate all three methods on corrupted versions of X_{test} while keeping X_{train} clean. Five noise types are tested: additive Gaussian, constant bias, linear drift, salt-and-pepper (random outliers), and heteroscedastic (noise proportional to $|x_{ij}|$). Each is applied at six intensity levels $\varsigma \in \{0, 0.05, 0.10, 0.20, 0.30, 0.50\}$, expressed as a fraction of the per-variable standard deviation of X_{train} . Since FULL uses all N_{poly} monomials while SVM-GRFE and SVM-RFE use a reduced subset, the primary comparison is between SVM-GRFE and SVM-RFE at equal monomial cardinality $|\Phi^*|$.

Table 3 reports the accuracy of SVM-GRFE and SVM-RFE without noise ($\varsigma = 0$) and at the highest noise level ($\varsigma = 0.50$), averaged over all five noise types. Since the five noise types are qualitatively different and may not be equally likely in practice, this average

■ **Table 4** Mean accuracy at $\varsigma = 0.50$ by noise type, averaged over the seven scenarios.

Noise type	FULL	SVM-GRFE	SVM-RFE	Δ_{acc}
	Accuracy	Accuracy	Accuracy	
Gaussian	0.941	0.916	0.911	+0.005
Bias	0.964	0.923	0.927	−0.004
Drift	0.976	0.965	0.946	+0.019
Heteroscedastic	0.840	0.870	0.769	+0.101
Salt-and-pepper	0.794	0.775	0.760	+0.015
Mean	0.903	0.890	0.863	+0.027

should be interpreted as a summary ranking indicator rather than a physically meaningful quantity. The difference $\Delta_{\text{acc}} = \text{Accuracy}_{\text{SVM-GRFE}} - \text{Accuracy}_{\text{SVM-RFE}}$ at $\varsigma = 0.50$ quantifies the robustness advantage of SVM-GRFE over SVM-RFE at equal complexity. SVM-GRFE outperforms SVM-RFE in six out of seven scenarios, with a mean advantage of $\Delta_{\text{acc}} = +0.028$. The advantage is most pronounced in *B6* (+0.088) and *B7* (+0.105): in *B6*, both methods select $\{x_1, x_2\}$ but SVM-GRFE produces a better-oriented boundary; in *B7*, SVM-GRFE correctly discards the detrimental x_4 -related monomials while SVM-RFE retains them. The only scenario where SVM-RFE is more robust is *B2* ($\Delta_{\text{acc}} = -0.037$) with or without noise.

Table 4 breaks down the accuracy at $\varsigma = 0.50$ by noise type, averaged over the seven scenarios. SVM-GRFE outperforms SVM-RFE in four out of five noise types, with the largest advantage for heteroscedastic noise ($\Delta_{\text{acc}} = +0.101$): since noise amplitude scales with measurement magnitude, the detrimental monomials retained by SVM-RFE (such as x_4 -related terms in *B7*) are amplified, while SVM-GRFE has already discarded them. For drift, SVM-GRFE also shows a clear advantage ($\Delta_{\text{acc}} = +0.019$). Bias is the only noise type where SVM-RFE marginally outperforms SVM-GRFE ($\Delta_{\text{acc}} = -0.004$), a difference that falls within the resolution of the accuracy metric at the test set sizes considered. FULL achieves the highest accuracy under all noise types except heteroscedastic, where SVM-GRFE surpasses it (0.870 vs. 0.840), confirming that the elimination of detrimental monomials can improve noise robustness even relative to the unpruned model. Table 5 reports, for each method and noise type, the mean over the seven scenarios of the lowest tested noise level $\varsigma \in \{0.05, 0.10, 0.20, 0.30, 0.50\}$ at which the accuracy first drops by more than 5% relative to the baseline without noise of each method. Since this threshold is averaged over scenarios, the resulting values may fall between two grid points and do not correspond to a single tested level. A value of 0.50 indicates that the 5% threshold was never reached within the tested range. Cells marked with † (see the note in the table) indicate that the threshold was never reached within the range tested. Drift is the most benign perturbation: FULL and SVM-GRFE never reach the 5% threshold, while SVM-RFE degrades at $\varsigma = 0.407$. Salt-and-pepper is the most damaging, with SVM-RFE degrading at $\varsigma = 0.121$ versus 0.143 for SVM-GRFE. For all noise types, SVM-GRFE either matches or outperforms SVM-RFE.

5 Conclusion

This paper introduced SVM-GRFE, a FS method for learning compact, interpretable, and robust transition conditions in CPS from multivariate TS data. By operating on polynomial monomials rather than raw physical variables and combining classification accuracy with geometric criteria, SVM-GRFE identifies detrimental monomials while preserving an explicit

■ **Table 5** Mean noise level at which accuracy first drops by more than 5% relative to the baseline without noise, by noise type.

Noise type	FULL	SVM-GRFE	SVM-RFE
Gaussian	0.471	0.379	0.364
Bias	0.471	0.314	0.343
Drift	$\geq 0.5^\dagger$	$\geq 0.5^\dagger$	0.407
Heteroscedastic	0.300	0.293	0.257
Salt-and-pepper	0.200	0.143	0.121

[†]The 5% threshold was never reached within the tested range $\varsigma \in [0, 0.50]$; the reported value is censored at the upper bound and is not a measured degradation level.

polynomial boundary directly reusable in diagnosis and control. Experiments on seven synthetic scenarios show that SVM-GRFE reduces model complexity while matching or outperforming the full model and standard SVM-RFE in most cases. Robustness evaluations under multiple noise types further demonstrate improved resilience to measurement uncertainty. Future work will focus on geometry-aware hyperparameter selection and validation on real CPS datasets. A case study assessment focusing on a large city will be conducted to examine scaling up. From a diagnostic standpoint, Beyond feature selection, the interpretable polynomial boundary expression produced by SVM-GRFE provides interpretable transition conditions that directly support fault isolation by explicitly identifying the physical variables involved in each mode transition.

References

- 1 T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *KDD*. ACM, 2019.
- 2 U. M Braga-Neto and E. R Dougherty. Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, 20(3):374–380, 2004. doi:10.1093/BIOINFORMATICS/BTG419.
- 3 C. G Cassandras and S. Lafortune. *Introduction to discrete event systems*. Springer, 3rd edition, 2021.
- 4 S. Chen, C. Gao, and P. Zhang. Incorporation of data-mined knowledge into black-box svm for interpretability. *ACM Trans. on Int. Syst. and Tech.*, 14(1):1–22, 2022. doi:10.1145/3548775.
- 5 C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. doi:10.1007/BF00994018.
- 6 H. Frohlich, O. Chapelle, and B. Scholkopf. Feature selection for support vector machines by means of genetic algorithm. In *ICTAI*, pages 142–148. IEEE, 2003.
- 7 I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1–3):389–422, 2002. doi:10.1023/A:1012487302797.
- 8 R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2):273–324, 1997. doi:10.1016/S0004-3702(97)00043-X.
- 9 B. Martin-Barragan, R. Lillo, and J. Romo. Interpretable support vector machines for functional data. *European Journal of Operational Research*, 232(1):146–155, 2014. doi:10.1016/J.EJOR.2012.08.017.
- 10 A. Salhi, R. Alshamrani, A. Althbiti, A. Ismail, M. Abd-ElRahman, and B. M Hassan. Optimizing high dimensional data classification with a hybrid ai driven feature selection framework and machine learning schema. *Scientific Reports*, 15(1):35038, 2025.

HSU TwinFlow: A Living Benchmark for Evaluating Anomaly Detection and Diagnosis Methods in Cyber-Physical Production Systems Using Digital-Twin-Generated Data

Nemanja Hranisavljevic  

Helmut Schmidt University, Hamburg, Germany

Alexander Diedrich  

Helmut Schmidt University, Hamburg, Germany

Lukas Moddemann  

Helmut Schmidt University, Hamburg, Germany

Domenic Schaeffer  

Digital Twin Factory GmbH, Bad Oeynhausen, Germany

Frank Marek  

Novatio Solutions GmbH, Monheim, Germany

Ingo Pill  

Institute of Software Engineering and Artificial Intelligence, TU Graz, Graz, Austria

Oliver Niggemann  

Helmut Schmidt University, Hamburg, Germany

Abstract

We present a living (evolving) benchmark for evaluating data-driven methods for fault-related tasks, including fault (anomaly) detection and diagnosis, based on datasets generated by a realistic digital twin of a cyber-physical production system (CPPS). The digital twin is designed to capture the structural layout, operational behavior, and material flow of a modern production facility in a realistic manner. Scenario definitions, represented as sequences of high-level events, are used to generate labeled model-training and evaluation data from carefully designed fault scenarios. In addition, the datasets include contextual data, such as scenario metadata, product configurations, material-flow information, together with system knowledge such as the component hierarchy, component-type information, and material-flow graph. The proposed benchmark therefore provides a practical basis for method development and qualitative evaluation beyond simplified or isolated test setups. We provide initial benchmark results by evaluating two representative methods for each of the two considered tasks: anomaly detection and fault diagnosis.

2012 ACM Subject Classification Computing methodologies → Artificial intelligence; Computer systems organization → Embedded and cyber-physical systems; Computer systems organization → Dependable and fault-tolerant systems and networks; Computing methodologies → Modeling and simulation; Computing methodologies → Machine learning

Keywords and phrases Benchmark, Digital Twin, Manufacturing, Production Systems, Cyber-Physical Systems, Artificial Intelligence, Anomaly Detection, Diagnosis, System Reconfiguration

Digital Object Identifier 10.4230/OASICS.DX.2026.7

Supplementary Material *Dataset:* <https://github.com/imb-hsu/twinflow>



© Nemanja Hranisavljevic, Alexander Diedrich, Lukas Moddemann, Domenic Schaeffer, Frank Marek, Ingo Pill, and Oliver Niggemann;

licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 7; pp. 7:1–7:14



OpenAccess Series in Informatics

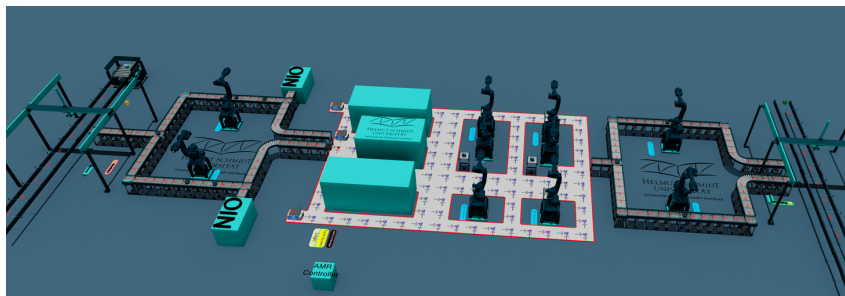
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The realization of robust, resilient and efficient production systems is a central objective in the field of cyber-physical production systems (CPPSs). In this context, various methods for analyzing system operation have been investigated, including anomaly detection, which aims to identify deviations from expected system behavior [9], and fault diagnosis that targets to isolate the root causes of the deviations in order to explain the problem [13, 4]. Although significant progress has been made on these tasks and many methods have been proposed, their evaluation is a quite complex problem [21]. Corresponding experiments are still typically conducted using isolated use cases, task-specific datasets, or simplified system models [12]. This limits the comparability of methods and makes it difficult to assess their suitability for realistic production settings.

One key challenge is the lack of comprehensive labeled evaluation data that enables a realistic, meaningful, and fair assessment [6]. Furthermore, production systems are highly diverse and consist of various interacting components whose integration gives rise to emergent system behavior. As with symbolic approaches based on first-principles reasoning [28, 24] and abductive reasoning [25], this requires a holistic analysis of system operation rather than isolated analysis of individual components [18]. Moreover, new technologies, machines, and devices, such as autonomous mobile robots (AMRs) and advanced industrial robots, are increasingly being introduced into production systems. These developments change the dependencies between system components and variables compared with earlier production systems [8]. As a result, continuously updated benchmarks are needed to evaluate whether proposed solutions remain applicable under evolving conditions [14].

To address this gap, we present *HSU TwinFlow*, a living digital-twin-based benchmark built around a hypothetical production system with *several hundred* components and *several thousand* signals. It encompasses conveyors, industrial robots, autonomous mobile robots (AMRs), and operating stations (see Figure 1). The digital twin captures both the structural layout and the operational logic of a production facility and generates production data from high-level input scenarios that define, for example, product characteristics and component parameters. In addition to nominal operation, the model supports the definition of failure cases affecting different asset types, process stages, and system locations. This allowed us to generate a benchmark dataset comprising representative, labeled scenarios for evaluating anomaly detection and fault diagnosis solutions. This living benchmark can be continuously extended and updated as the digital twin, scenarios, and evaluation requirements evolve.



■ **Figure 1** The *HSU TwinFlow* digital twin is implemented in *Emulate3D* and combines physics-based simulation with components from *Digital Twin Factory* and scenario generation by *Novatio*.

The contributions of this work are threefold:

- We introduce a digital twin of a complex production system comprising multiple interacting asset classes, including conveyors, robots, AMRs, and operating stations.
- We provide training and evaluation datasets with both nominal and faulty behavior, following common anomaly detection and diagnosis requirements.
- We establish an initial benchmark by evaluating two representative anomaly detection methods and two representative fault diagnosis methods.

The remainder of this paper is structured as follows. Section 2 presents related work. Section 3 introduces the digital twin, including the considered asset classes and the operational modeling approach. Section 4 defines the benchmark requirements, as well as the evaluation protocols, for anomaly detection and diagnosis. Section 5 presents the benchmark design, including scenario-based data generation and fault modeling, while Section 6 reports the initial experiments and results obtained with selected methods. Finally, Section 7 summarizes the main conclusions, and outlines limitations and future work.

2 Related Work

Benchmark-based evaluation is essential for comparing fault (anomaly) detection and diagnosis methods under reproducible conditions. Established industrial benchmarks, such as the Tennessee Eastman Process, provide controlled process data and have been widely used to evaluate fault detection methods [5, 10]. Other benchmark datasets focus on specific data characteristics or sensing modalities. For example, multivariate time-series benchmarks support the systematic comparison of anomaly detection and diagnosis methods in cyber-physical systems [9], while datasets such as MIMII provide acoustic observations of normal and anomalous behavior in industrial machines [27]. These benchmarks have contributed significantly to method development, but they typically address specific processes, data modalities, or isolated tasks.

The diagnosis community has also established shared evaluation settings through benchmark competitions. The First International Diagnosis Competition (DXC'09) introduced a common framework for comparing diagnosis algorithms on shared systems, scenarios, ground truth, and evaluation criteria [16]. Subsequent editions, including DXC'10, DXC'11, and DXC'13, continued this effort across industrial, synthetic, software, and application-specific tracks [15, 26, 31]. More recently, the DX Competition 2025 revived this benchmark-oriented competition format with several dedicated fault-diagnosis benchmarks [23]. Domain-specific diagnosis benchmarks have also been proposed, for example for water distribution networks [32]. These works demonstrate the value of shared evaluation environments, while also highlighting the need for benchmarks tailored to the structure and dynamics of specific application domains.

In the CPPS domain, existing benchmarks and datasets address important aspects of fault-related tasks in production systems. Some provide simulation environments and datasets for evaluating AI-based methods for fault handling, diagnosis, reconfiguration, and planning [1, 6, 17]. These works are particularly relevant because they consider production-system characteristics such as interacting assets, modular system structures, and operational dependencies. Related work further addresses artificial sensor-data generation for comparing unsupervised learning methods [33], learning physical concepts in CPSs [30], and scalable modeling of time-dependent production-line sensor data, for example using the Bosch Production Line Performance dataset [14]. However, existing benchmarks typically rely on specific system abstractions, simplified scenarios, or assumptions that limit their representativeness for realistic CPPSs.

Digital twins provide a promising basis for addressing this limitation, as they can closely represent real systems while enabling controllable simulation of complex system behavior [19, 3]. They are particularly relevant for production-oriented benchmarks because they can connect operational data with asset structures, process dependencies, and system context. However, reusable benchmark datasets that combine realistic production behavior, structured digital-twin information, and controlled fault scenarios remain limited.

3 **HSU TwinFlow: A CPPS Digital Twin**

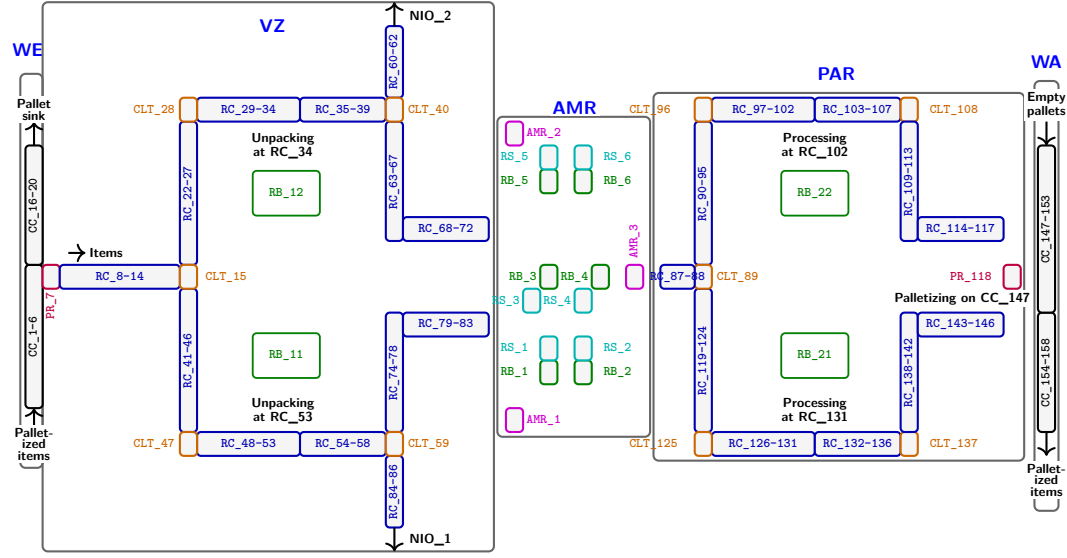
The digital twin was developed by combining modules inspired by several existing real-world systems into a representative industrial production system. Its layout is shown in Figure 2. The system consists of five functional areas, each containing specific asset types:

- **Goods input (WE):** The goods input area represents the start of the plant, where raw items enter the system on pallets. It includes two segments of connected chain conveyors, CC_1-6 and CC_16-20, with creation of pallets with input items at one end and a sink for empty pallets at the other end.
- **Singulation area (VZ):** The singulation area separates incoming lots of palletized items into individual items using the portal robot PR_7. It also comprises roller conveyors RC and chain-lift conveyors CLT, which redirect items from one conveyor line to another. Two six-axis robots, RB_11 and RB_12, unpack items at conveyors RC_34 and RC_53, respectively. The implemented logic further separates defective items and directs them to the non-conforming outputs NIO_1 and NIO_2.
- **Autonomous mobile robots area (AMR):** The autonomous mobile robots area follows a matrix-production concept. It consists of three autonomous robots AMR, which transport items to any of the six robot stations RS. Each robot station has a corresponding robot RB that processes items according to instructions defined for each item type. For example, an item may be routed through RS_1, then RS_3, and finally RS_2.
- **Parallel processing area (PAR):** The parallel processing area distributes items across two parallel processing paths. It includes roller conveyors RC, chain-lift-transfer conveyors CLT, and two robots, RB_21 and RB_22, which perform final processing at RC_102 and RC_131, respectively.
- **Goods output (WA):** The goods output area represents the final output of the plant. Empty pallets arrive on chain conveyors, and a robot places finished products onto them. The loaded pallets then continue through the output area and leave the plant.

The behavior of the complete CPPS is defined by a hybrid simulation model that combines discrete-event production logic with physics-based component behavior. Discrete events represent production and material-flow operations, such as product routing, conveyor transport, robot actions, and fault injection, while continuous component behavior is represented by underlying physics-based simulation models governing variables such as current, speed, and force. As a result, we obtain a rich system-level simulation model of the CPPS.

4 **Benchmark Requirements and Protocols**

This section first introduces the general process knowledge, followed by the task-specific requirements for anomaly detection and diagnosis and the corresponding evaluation protocols used in the benchmark. The implemented production logic can be represented as a directed material flow graph, in which nodes denote system assets and edges represent possible material transitions. This structural information is made available to the evaluated methods as part of the expert knowledge provided by the benchmark.



■ **Figure 2** Two-dimensional layout of the *HSU TwinFlow* production system.

The corresponding knowledge graph is provided as a json file defining the nodes and edges of the material flow graph. It can be used to compare observed system behavior with the intended system design and to infer possible future material movements based on the current state and position of items in the system. A tiny subset of approximately 1000 variables is plotted in Figure 4.

4.1 Anomaly Detection Requirements and Protocols

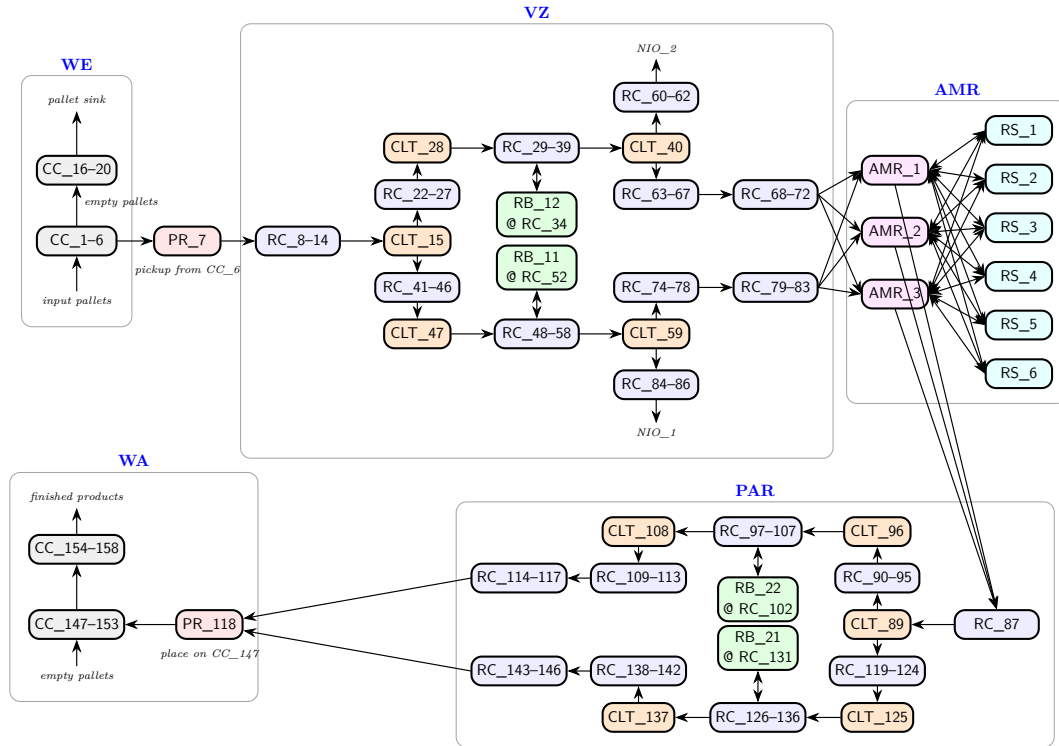
The first task targeted by our benchmark is anomaly detection (AD). AD aims to distinguish nominal system behavior from abnormal operating conditions and is commonly categorized into supervised, semi-supervised, and unsupervised settings [2]. For CPPSs, semi-supervised and unsupervised settings are particularly relevant, as CPPSs typically provide data dominated by normal operation [20]. In this work, normal-only training data corresponds to semi-supervised AD, whereas training data that may contain anomalies corresponds to unsupervised AD.¹

Anomaly detection methods typically compute a continuous anomaly score $as(\mathbf{x}_i)$ for each system observation $\mathbf{x}_i$ at timestamp t_i . A threshold A_{th} is then applied to classify the observation as normal, if $as(\mathbf{x}_i) \leq A_{th}$, or anomalous, if $as(\mathbf{x}_i) > A_{th}$. If a small labeled subset is available, it may support a more informed threshold selection. During evaluation, anomaly detection models are assessed using a labeled test set.

Therefore, we define the anomaly detection requirements for the benchmark as follows:

Req. AD1 The training data shall consist of one or more observation sequences with corresponding timestamps representing nominal system operation. The dataset may contain a small fraction of abnormal data points at unknown timestamps ($\mapsto$ unsupervised AD).

¹ The *HSU TwinFlow* supports both unsupervised and semi-supervised setting based on the selected training data, as will be presented in Section 5.



■ **Figure 3** Material flow diagram of the *HSU TwinFlow* production system, showing the main production areas, system assets (components), and directed material flow between them.

Req. AD2 The test data shall provide observation sequences with point-wise anomaly labels a_j for each timestamp t_j .

Req. AD3 *Optional:* A (small) labeled dataset may be provided to support training or threshold selection.

Furthermore, suitable metrics are required to evaluate different anomaly detection approaches. For this benchmark, we use:

1. *Point-wise anomaly detection balanced accuracy*, defined as:

$$BA = \frac{1}{2} \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right) \quad (1)$$

where TP , TN , FP , and FN denote point-wise (for each timestamp) true positives, true negatives, false positives, and false negatives, respectively.

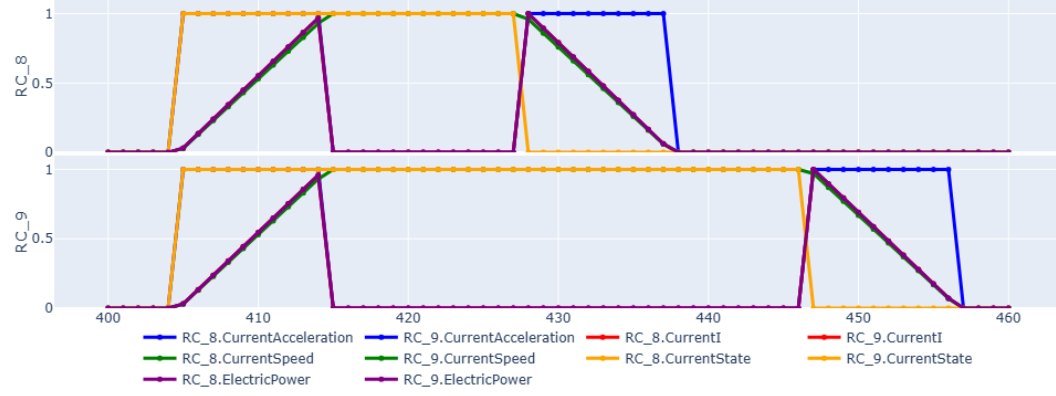
2. *Point-wise anomaly detection F-score*, defined as:

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}, \quad P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN} \quad (2)$$

where P and R denote point-wise precision and recall, respectively.

3. *F1 composite score* common in time-series anomaly detection [9], defined as:

$$F_{c1} = \frac{2 \cdot P \cdot R_{\text{event}}}{P + R_{\text{event}}} \quad (3)$$



■ **Figure 4** Data of conveyors RC_8 and RC_9 during 1 minute.

where P is point-wise, defined above, and:

$$R_{\text{event}} = \frac{TP_{\text{event}}}{TP_{\text{event}} + FN_{\text{event}}} \quad (4)$$

denotes event-wise recall. Here, TP_{event} is the number of true anomalous segments during which at least one time point is correctly detected as anomalous, while FN_{event} is the number of true anomalous segments missed entirely.

4.2 Diagnosis Requirements and Protocols

Fault diagnosis (DX) aims to explain identified abnormal behavior by determining the affected component(s) and, where possible, the corresponding fault type(s). In the following, let $c \in \mathcal{C}$ denote a system component and let $f \in \mathcal{F}$ denote a fault type. A diagnosis label set $\mathcal{D}$ is represented as a set of tuples $d = (c, f)$, where each tuple specifies that component c is affected by fault type f . Here $\mathcal{D}$ contains at most one tuple for one system component to ensure the mutual exclusiveness of faults:

$$\mathcal{D} \subseteq \mathcal{C} \times \mathcal{F}, \quad (c, f_1) \in \mathcal{D} \wedge (c, f_2) \in \mathcal{D} \Rightarrow f_1 = f_2. \quad (5)$$

Components that do not appear in any tuple in $\mathcal{D}$ are assumed to be healthy for the considered segment.

Diagnosis approaches commonly use operational data to characterize symptoms, analyze them with respect to expected system behavior or a model, and infer plausible root causes [20]. However, explicit causal or executable simulation models are often not available during diagnosis in the CPPS domain. For this reason, this benchmark supports the approaches that use historical operational data together with structural system knowledge to infer likely diagnosis hypotheses.

We define the diagnosis requirements as follows:

- Req. DX1** The training data shall predominantly represent normal system operation to support the analysis of nominal system behavior.
- Req. DX2** The test data shall contain both normal and anomalous segments. Each anomalous segment shall be associated with a diagnosis label set identifying the corresponding fault(s).
- Req. DX3** *Optional:* Labeled training data containing anomalous segments may be available to support the analysis of system responses to known faults. If provided, each anomalous segment shall be associated with a diagnosis label set.

Req. DX4 A graph shall represent the hierarchical relationships between system components and variables, and shall relate affected components to observed variables.²

Req. DX5 For each system component or component type, the available system knowledge shall specify the set of applicable fault types.²

In this benchmark, **Req. DX4** is implicitly reflected in the variable identifiers used in the dataset. Each complete variable identifier is formed by combining the component identifier with the corresponding signal name. For example, `RC_48.CurrentSpeed` denotes the `CurrentSpeed` signal of component `RC_48`. Furthermore, **Req. DX5** is addressed by the system knowledge summarized in Table 1.

■ **Table 1** Representative types of faults (failures) for each component type in the *HSU TwinFlow* digital twin.

CC, RC, CLT, RS	PR	AMR	RB
EmergencyStop	EmergencyStop	EmergencyStop	EmergencyStop
MotorFault	MotorFault	MotorFault	MotorFault
SpeedCalibrationFault	SpeedCalibrationFault	SpeedCalibrationFault	SpeedCalibrationFault
WearFault	WearFault	ChargeEmpty	
IncreasedDampingFault	IncreasedDampingFault		
IMeasurementFault			
PeDefect			
PeWrongCalibration			

In this work, diagnosis is evaluated at the segment (sequence) level. Thus, the input to a diagnosis method is a window of operational data, and each segment is associated with a diagnosis label set $\mathcal{D}$. For example, a ground-truth label set may be given as $\mathcal{D} = \{(\text{RC_48}, \text{MotorFault}), (\text{RB_11}, \text{SpeedCalibrationFault})\}$, while a method may return the predicted diagnosis set $\hat{\mathcal{D}} = \{(\text{RC_48}, \text{MotorFault}), (\text{RB_12}, \text{SpeedCalibrationFault})\}$. For localization performance, only the component identifiers in the tuples are compared, whereas for fault-type performance, also the fault types are considered.

To evaluate diagnosis results, we use the following metrics:

- **Localization balanced accuracy:** Measures whether the method correctly identifies the affected components. For each diagnosis instance, the predicted component set $\hat{\mathcal{C}}$ is compared with the ground-truth component set $\mathcal{C}$, derived from the diagnosis tuple sets $\hat{\mathcal{D}}$ and $\mathcal{D}$.
- **Fault-type balanced accuracy:** Measures whether the method correctly identifies the occurring fault types. For each diagnosis instance, the predicted fault-type set $\hat{\mathcal{F}}$ is compared with the ground-truth fault-type set $\mathcal{F}$, derived from the diagnosis tuple sets $\hat{\mathcal{D}}$ and $\mathcal{D}$.
- **Localization F1-score:** The localization F1-score evaluates the overlap between the predicted and ground-truth affected component sets. It is computed from precision and recall over component labels, treating components in $\hat{\mathcal{C}} \cap \mathcal{C}$ as true positives.
- **Fault-type F1-score:** The fault-type F1-score evaluates the overlap between the predicted and ground-truth fault-type sets. It is computed from precision and recall over fault-type labels, treating fault types in $\hat{\mathcal{F}} \cap \mathcal{F}$ as true positives.

² The structural system graph and the mapping from component types to possible fault types are provided by the *HSU TwinFlow* benchmark as separate `json` files.

5 Benchmark Design

The *HSU TwinFlow* follows a scenario-based simulation concept. A scenario specifies a sequence of timed actions and parameter choices that define the execution of a production process. This concept allows the systematic creation of simulation runs under nominal and faulty operating conditions. The scenarios can address individual assets, groups of assets, or complete plant sections.

To address the anomaly detection requirements defined in the previous section, several simulation scenarios are defined. Each scenario specifies when a fault starts, and when the affected component returns to normal operation. The simulations are then executed using these scenario definitions, so that random fault occurrences and repair actions are injected into otherwise nominal system behavior.

In each simulation run, faults (if any) are injected at specified timestamps and repaired after 30 s. Each scenario is defined by a sequence of tuples containing:

- the timestamp at which each fault starts (or ends i.e., when the system is repaired.),
- the affected component,
- the corresponding fault type.

Each file covers a simulated duration of 60 minutes, with operational data sampled with a sampling rate of $f_s = 10$ Hz.

Req. AD1 and **Req. DX1** are fulfilled by simulating three training settings, in which 0%, 1%, and 10% of the total simulated time correspond to faulty segments. These settings provide training data ranging from purely nominal operation to data containing a small fraction of randomly placed faulty intervals (see training files in Figure 5).

Req. AD2/Req. DX2 and **Req. AD3/Req. DX3** are addressed by generating faulty scenarios with explicit fault-injection and repair actions (see evaluation files in Figure 5).

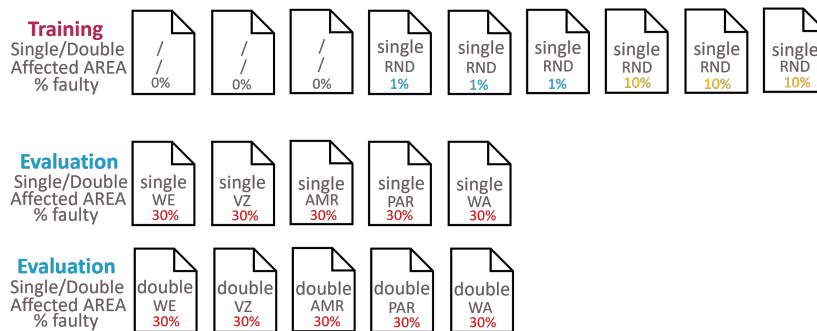


Figure 5 Generated training and evaluation scenarios in the first *Release* of the benchmark datasets.

For evaluation, multiple scenario files are generated. Each file samples faults from one production area only, i.e., from WE, VZ, AMR, PAR, or WA. Within the selected area, faults are sampled uniformly from the applicable components and fault types. For conveyor-based components, a conveyor line segment (of connected conveyors) is sampled first, instead of directly sampling an individual conveyor. In a second step, one conveyor within the selected line segment is sampled uniformly. This avoids overrepresenting long conveyor lines while still allowing faults to occur on individual conveyor components. The benchmark includes both single-fault and two-fault scenarios. The evaluation scenario sequences are generated such that approximately 30% of each sequence corresponds to faulty behavior.

6 Benchmark Demonstration with Representative Methods

Two anomaly detection methods are considered:

1. Range Monitoring [11]
2. Vanilla Autoencoder [7]

Furthermore, two diagnosis methods are evaluated:

1. Structural-Knowledge-Based Fault Diagnosis [29]
2. Case-Based Fault Diagnosis [22]

6.1 Anomaly Detection

- **Range Monitoring:** Range monitoring is a threshold-based anomaly detection approach. For each observable variable j , an expected operating range is defined by a lower and an upper bound. Given an observation vector $\mathbf{x}_i$ at timestamp t_i , the value of variable j is denoted by $x_{i,j}$. The observation is classified as anomalous if at least one variable falls outside its predefined range:

$$x_{i,j} < X_j^{\min} \quad \text{or} \quad x_{i,j} > X_j^{\max} \quad \text{for all } j$$

The method is simple, transparent, and easy to interpret. It is especially suitable for variables with known physical or operational limits. However, range monitoring only detects point-wise deviations and does not capture temporal patterns or correlations between multiple variables.

- **Vanilla Autoencoder:** A vanilla autoencoder is an unsupervised neural-network-based anomaly detection method. It consists of an encoder, which maps an observation vector to a lower-dimensional latent representation, and a decoder, which reconstructs the original input from this representation. During training, the autoencoder learns to reconstruct normal operating data. Given an observation vector $\mathbf{x}_i$ at timestamp t_i , the reconstructed output is given by:

$$\hat{\mathbf{x}}_i = g_\theta(f_\phi(\mathbf{x}_i)),$$

where f_ϕ denotes the encoder, g_θ denotes the decoder, and $\hat{\mathbf{x}}_i$ is the reconstructed observation vector. During evaluation, anomalies are detected using the reconstruction error:

$$e_i = \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|.$$

If the reconstruction error e_i exceeds a predefined threshold, the observation at timestamp t_i is classified as anomalous. Compared with range monitoring, the autoencoder can capture nonlinear relationships between variables, but its results are less interpretable.

Table 2 summarizes the benchmark results of the two anomaly detection methods.

■ **Table 2** Benchmark results of the anomaly detection methods.

Method	BA	F1	F1 Comp.
Range Monitoring	Results are available in the public GitHub repository .		
Vanilla Autoencoder			

6.2 Diagnosis

- **Structural-Knowledge-Based Fault Diagnosis:** Structural-Knowledge-Based Fault Diagnosis uses prior knowledge about the organization of the production system to localize faults. The system structure is represented by relationships between production areas, component types, individual components, and observable variables. After an anomaly is detected in one or more signals, the affected variables are mapped to the corresponding component nodes in the structural knowledge model. Based on the identified component type, the set of possible fault types can be retrieved. This approach does not require an explicit causal model. Instead, it relies on ownership and hierarchy relations:

$$\text{anomalous variable} \rightarrow \text{component} \rightarrow \text{component type} \rightarrow \text{possible faults}$$

Therefore, this method provides an interpretable baseline for mapping detected anomalies to affected components and candidate fault types.

- **Case-Based Fault Diagnosis:** Case-Based Fault Diagnosis uses previously observed fault cases as reference patterns for diagnosing new abnormal behavior. In this work, the reference cases are extracted from the labeled generated files, which are not used by the first diagnosis method. Each reference case contains an observation vector $\mathbf{x}_r$ (or consecutive vectors) together with the corresponding fault label, affected component, component type, and production area. During diagnosis, a test observation vector $\mathbf{x}_i$ is compared with the stored reference cases. The most similar case is retrieved, and its label is used to infer the likely fault type and affected component. A general similarity-based diagnosis step is:

$$c^* = \arg \min_{c_r \in \mathcal{C}} d(\mathbf{x}_i, \mathbf{x}_r)$$

where $\mathbf{x}_i$ is the observation vector, $\mathbf{x}_r$ is the observation vector of a stored case c_r , $\mathcal{C}$ is the set of reference cases, and $d(\cdot, \cdot)$ is a distance or dissimilarity measure. The diagnosis result is derived from the label of the nearest reference case c^* . In this work, $d(\cdot, \cdot)$ is defined as the Euclidean distance.

Compared with Structural-Knowledge-Based Fault Diagnosis, Case-Based Fault Diagnosis can exploit characteristic multivariate signal patterns. However, its performance depends on the coverage and quality of the available reference cases.

Table 3 summarizes the benchmark results of the diagnosis methods.

■ **Table 3** Benchmark results of the diagnosis methods.

Method	Loc. BA	Fault BA	Loc. F1	Fault F1
Structural-Knowledge-Based	Results are available in the public GitHub repository .			
Case-Based				

6.3 Discussion of Results

The benchmark results provide an initial indication of how the evaluated methods perform on the proposed datasets. The selected anomaly detection and diagnosis methods cover different levels of complexity and interpretability, allowing a first comparison between simple baseline

approaches and data-driven alternatives. Overall, the results illustrate the applicability of the benchmark and provide a basis for more detailed analyses of method behavior, failure cases, and performance differences in future evaluations.

7 Conclusion

The proposed benchmark addresses an important gap between simplified benchmark systems and the requirements of applied and theoretical research on industrial CPPSs. Its main strength lies in combining a representative digital twin model with configurable fault scenarios and scenario-based data generation. This enables the systematic creation of both nominal and faulty operating data under controlled and reproducible conditions.

The benchmark provides the structural knowledge required for evaluating and comparing anomaly detection and diagnosis methods, including a material flow graph, component-level system knowledge, configurable scenarios, and quantitative evaluation protocols. In addition, the benchmark is demonstrated using four representative methods, covering both simple interpretable baselines and more data-driven approaches.

A further strength of the proposed setup is its extensibility. Additional fault types, process variants, operating conditions, and reconfiguration strategies can be incorporated without changing the fundamental benchmark design. This makes the benchmark suitable for future work on robust, adaptive, and explainable methods for real-world CPPSs.

By enabling the controlled generation of normal and faulty operating data, the benchmark provides a practical foundation for evaluating fault-related methods in production systems. Future work will extend the benchmark by integrating additional methods, expanding the set of fault scenarios, system reconfiguration, and further analyzing the influence of variable availability and process variability on detection and diagnosis performance.

Data and Code Availability

To support reproducibility and future evaluations of additional methods, the source code, knowledge graphs, scenario configurations, and documentation of our benchmark are available from a GitHub repository: <https://github.com/imb-hsu/twinflow>. The repository also provides links to the training and test dataset files.

It should be noted that the provided dataset contains measurements of all variables available in the CPPS. This level of observability is not necessarily realistic for a real-world CPPS, where only a subset of variables may be accessible. However, providing the full set of variables allows users to select arbitrary subsets for evaluating anomaly detection or diagnosis methods. For example, this enables experiments that analyze how performance changes as the number of available variables is reduced.

References

- 1 Kaja Balzereit, Alexander Diedrich, Jonas Ginster, Stefan Windmann, and Oliver Niggemann. An ensemble of benchmarks for the evaluation of ai methods for fault handling in cpps. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–6. IEEE, 2021. doi:10.1109/INDIN45523.2021.9557516.
- 2 Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. doi:10.1145/1541880.1541882.
- 3 Alexander Diedrich, Christian Kühnert, Georg Maier, Joshua Schraven, and Oliver Niggemann. AITwin: A uniform digital twin interface for artificial intelligence applications. In Daniel Schulz and Christian Bauckhage, editors, *Informed Machine Learning*, pages 41–60. Springer Nature Switzerland, Cham, 2025. doi:10.1007/978-3-031-83097-6_3.

- 4 Brett Dowdeswell, Roopak Sinha, and Stephen G. MacDonell. Finding faults: A scoping study of fault diagnostics for industrial cyber-physical systems. *Journal of Systems and Software*, 168:110638, 2020. doi:10.1016/j.jss.2020.110638.
- 5 James J. Downs and Ernest F. Vogel. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17(3):245–255, 1993. doi:10.1016/0098-1354(93)80018-I.
- 6 Jonas Ehrhardt, Malte Ramonat, René Heesch, Kaja Balzereit, Alexander Diedrich, and Oliver Niggemann. An AI benchmark for diagnosis, reconfiguration & planning. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2022. doi:10.1109/ETFA52439.2022.9921546.
- 7 Benedikt Eiteneuer, Nemanja Hranisavljevic, and Oliver Niggemann. Dimensionality Reduction and Anomaly Detection for CPPS Data using Autoencoder. In *20th IEEE International Conference on Industrial Technology*, volume 2019-Febru, Melbourne, Australia, March 2019. doi:10.1109/ICIT.2019.8755116.
- 8 Hoda ElMaraghy, Laszlo Monostori, Guenther Schuh, and Waguih ElMaraghy. Evolution and future of manufacturing systems. *CIRP Annals*, 70(2):635–658, 2021. doi:10.1016/j.cirp.2021.05.008.
- 9 Astha Garg, Wenyu Zhang, Jules Samaran, Ramasamy Savitha, and Chuan-Sheng Foo. An Evaluation of Anomaly Detection and Diagnosis in Multivariate Time Series. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–10, August 2021. doi:10.1109/TNNLS.2021.3105827.
- 10 Fabian Hartung, Billy Joe Franks, Tobias Michels, Dennis Wagner, Philipp Liznerski, Steffen Reithermann, Sophie Fellenz, Fabian Jirasek, Maja Rudolph, Daniel Neider, Heike Leitte, Chen Song, Benjamin Kloepper, Stephan Mandt, Michael Bortz, Jakob Burger, Hans Hasse, and Marius Kloft. Deep anomaly detection on tennessee eastman process data. *Chemie Ingenieur Technik*, 95(9):1415–1425, 2023. doi:10.1002/cite.202200238.
- 11 Nemanja Hranisavljevic, Alexander Maier, and Oliver Niggemann. Discretization of hybrid CPPS data into timed automaton using restricted Boltzmann machines. *Engineering Applications of Artificial Intelligence*, 95:103826, 2020. doi:10.1016/j.engappai.2020.103826.
- 12 Nemanja Hranisavljevic, Tom Westermann, Swantje Plambeck, Henrik Sebastian Steude, Gesa Benndorf, and Oliver Niggemann. A Model Learning Perspective on the Complexity of Cyber-Physical Systems. In Oliver Niggemann, Jürgen Beyerer, and Alexander Diedrich, editors, *Machine Learning for Cyber-Physical Systems*, 2025. doi:10.24405/20025.
- 13 Inseok Hwang, Sungwan Kim, Youdan Kim, and Chze Eng Seah. A survey of fault detection, isolation, and reconfiguration methods. *IEEE Transactions on Control Systems Technology*, 18(3):636–653, 2010. doi:10.1109/TCST.2009.2026285.
- 14 Felix Janzen, Alexander Diedrich, and Oliver Niggemann. Modular framework for scalable analysis and modeling of time-dependent industrial sensor data using the example of the bosch production line performance dataset. In *Proceedings of the 36th CIRP Design Conference*, Tokyo, March 2026. Elsevier.
- 15 Tolga Kurtoglu, Sriram Narasimhan, Scott Poll, David Garcia, Lukas Kuhn, Johan de Kleer, and Alexander Feldman. Second international diagnostic competition (dxc'10), 2010.
- 16 Tolga Kurtoglu, Sriram Narasimhan, Scott Poll, David Garcia, Lukas Kuhn, Johan de Kleer, Arjan van Gemund, and Alexander Feldman. First international diagnosis competition-dxc'09. In *Proc. International Workshop on Principles of Diagnosis*, volume 9, pages 383–396, 2009.
- 17 Lukas Moddemann, Jonas Ehrhardt, Alexander Diedrich, and Oliver Niggemann. The HAI-CPPS benchmark: Evaluating AI capabilities across hybrid data spaces. In *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2025. doi:10.1109/ETFA65518.2025.11205680.
- 18 Laszlo Monostori, Botond Kádár, Thomas Bauernhansl, Shinsuke Kondoh, Soundar Kumara, Gunther Reinhart, Olaf Sauer, Günther Schuh, Wilfried Sihn, and K Ueda. Cyber-physical systems in manufacturing. *CIRP Annals - Manufacturing Technology*, 65:621–641, 2016. doi:10.1016/j.cirp.2016.06.005.

- 19 Oliver Niggemann, Alexander Diedrich, Christian Kühnert, Erik Pfannstiel, and Joshua Schraven. A generic DigitalTwin model for artificial intelligence applications. In *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 55–62, 2021. doi:10.1109/ICPS49255.2021.9468243.
- 20 Oliver Niggemann and Volker Lohweg. On the diagnosis of cyber-physical production systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.
- 21 Ingo Pill and Johan de Kleer. Assessing Diagnosis Algorithms: Of Sampling, Baselines, Metrics and Oracles. In Marcos Quinones-Grueiro, Gautam Biswas, and Ingo Pill, editors, *36th International Conference on Principles of Diagnosis and Resilient Systems (DX 2025)*, volume 136 of *Open Access Series in Informatics (OASICS)*, pages 5:1–5:19, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASICS.DX.2025.5.
- 22 Ingo Pill, Johan de Kleer, and Franz Wotawa. Challenges for model-based diagnosis. In *Proceedings of the 35th International Workshop on Principles of Diagnosis*, OASICS, 2024. doi:10.4230/OASICS.DX.2024.6.
- 23 Ingo Pill, Daniel Jung, et al. The dx competition 2025 and its benchmarks. In *International Conference on Principles of Diagnosis and Resilient Systems*, 2025.
- 24 Ingo Pill and Thomas Quaritsch. Rc-tree: A variant avoiding all the redundancy in reiter’s minimal hitting set algorithm. In *Proceedings of the 2015 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, ISSREW ’15, pages 78–84, USA, 2015. IEEE Computer Society. doi:10.1109/ISSREW.2015.7392050.
- 25 Ingo Pill and Franz Wotawa. *Fault Detection and Localization Using Modelica and Abductive Reasoning*, pages 45–72. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-74962-4_3.
- 26 Scott Poll, Johan de Kleer, Rui Abreu, Matthew Daigle, Alexander Feldman, David Garcia, and Andy Sweet. Third international diagnostic competition – dxc’11. In *Proc. International Workshop on Principles of Diagnosis*, pages 267–278, 2011.
- 27 Harsh Purohit, Ryo Tanabe, Kenji Ichige, Takashi Endo, Yuki Nikaido, Kaori Suefusa, and Yohei Kawaguchi. MIMII dataset: Sound dataset for malfunctioning industrial machine investigation and inspection. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop*, pages 209–213, 2019.
- 28 Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987. doi:10.1016/0004-3702(87)90062-2.
- 29 H. S. Steude, A. Diedrich, I. Pill, L. Moddemann, D. Vranješ, and O. Niggemann. Data-driven diagnosis for large cyber-physical systems with minimal prior information. In *Proceedings of the IEEE Conference on Artificial Intelligence (CAI)*. IEEE, May 2026.
- 30 Henrik Steude, Alexander Windmann, and Oliver Niggemann. Learning physical concepts in cps: A case study with a three-tank system. *IFAC-PapersOnLine*, 55(6):15–22, 2022.
- 31 Andy Sweet, Alexander Feldman, Sriram Narasimhan, Matthew Daigle, and Scott Poll. Fourth international diagnostic competition – dxc’13. In *Proc. International Workshop on Principles of Diagnosis*, pages 224–229, 2013.
- 32 Anna Sztzyber, Elodie Chanthery, and Louise Travé-Massuyès. Benchmark for fault diagnosis of water distribution network. In *The 34th International Workshop on Principles of Diagnosis (DX’23)*, 2023.
- 33 Bernd Zimmering, Oliver Niggemann, Constanze Hasterok, Erik Pfannstiel, Dario Ramming, and Julius Pfrommer. Generating Artificial Sensor Data for the Comparison of Unsupervised Machine Learning Methods. *Sensors*, 21:2397, 2021. doi:10.3390/s21072397.

Hybrid Fault Detection and Isolation for the Fuel Turbopump Subsystem of the LOX/LNG Expander-Bleed Rocket Engine LUMEN

Eldin Kurudzija 

Institute of Space Propulsion, German Aerospace Center (DLR), Köln, Germany

Marcos Quinones-Grueiro 

Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA

Kai Dresia 

Institute of Space Propulsion, German Aerospace Center (DLR), Erlenbach, Germany

Austin Coursey 

Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA

Gautam Biswas 

Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA

Jan Deeken 

Institute of Space Propulsion, German Aerospace Center (DLR), Hardthausen, Germany

Institute of Jet Propulsion and Turbomachinery, RWTH Aachen, Germany

Günther Waxenegger-Wilfing 

Institute of Space Propulsion, German Aerospace Center (DLR), Hardthausen, Germany

Institute of Computer Science, University of Würzburg, Germany

Abstract

The transition toward reusable and cost-efficient launch systems increases the need for advanced fault detection and isolation (FDI) capabilities to support the reliable and safe operation of liquid rocket engines. Development of these systems typically relies on physics-based simulation models and experimental data. LUMEN (Liquid Upper stage deMonstrator ENgine) is a modular, pump-fed LOX/LNG rocket engine with a nominal thrust of 25 kN, developed and operated by the Institute of Space Propulsion of the German Aerospace Center (DLR). The Rocket Engine Control and Diagnosis Benchmark provides a high-fidelity, experimentally validated EcosimPro/ESPSS simulation model of LUMEN for FDI system development. In this work, we use the fuel turbopump subsystem of this model as a case study and integrate data-driven components into a consistency-based diagnosis (CBD) framework to improve diagnostic performance. The results demonstrate that the hybrid architecture improves isolation accuracy, increasing it from 0.46 to 0.76 compared to the CBD baseline. In addition, we propose a sensor validation network (SVN) that uses a grey-box neural ordinary differential equation surrogate driven by valve-command inputs to decouple sensor faults from component faults. This further increases the isolation accuracy to 0.92, while retaining the ability to detect unknown component faults. These results indicate that incorporating data-driven components helps to compensate for structural non-isolability and improve diagnostic performance while preserving physical interpretability.

2012 ACM Subject Classification Applied computing → Aerospace; Computing methodologies → Anomaly detection; Computing methodologies → Neural networks; Mathematics of computing → Time series analysis

Keywords and phrases Hybrid, Fault Detection and Isolation, Liquid Rocket Engines, LUMEN

Digital Object Identifier 10.4230/OASICS.DX.2026.8

Acknowledgements ChatGPT (GPT-5.3–5.6) and Claude (Opus 4.6/4.7, Sonnet 4.6/5) were used for language editing and code development. The authors reviewed and verified all generated outputs.



© Eldin Kurudzija, Marcos Quinones-Grueiro, Kai Dresia, Austin Coursey, Gautam Biswas, Jan Deeken, and Günther Waxenegger-Wilfing;
licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 8; pp. 8:1–8:20



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Liquid-propellant rocket engines (LPREs) are complex, highly coupled systems whose safety depends on reliable monitoring during both steady-state and transient operation. The transition toward reusable and cost-efficient propulsion systems increases the need for advanced fault detection and isolation (FDI) methods to support safe testing, reduce development risk and enable reliable operation throughout the engine life cycle [10, 33, 36]. The Space Shuttle Main Engine (SSME) was the first reusable LPRE and incorporated advanced health monitoring capabilities [30]. Beyond conventional redline monitoring, it featured a simulation-based diagnostic algorithm that compared engine measurements against predictions from a nonlinear thermodynamic mass- and energy-balance model to estimate the location and magnitude of predefined anomalies [5]. Such simulation models are developed and validated with experimental data throughout the design, development and testing of rocket engines, providing an established basis for FDI development [25].

More recently, research has expanded beyond this SSME-centered literature to a broader range of LPRE applications. These methods are commonly classified into model-based and data-driven approaches [41]. Quantitative model-based methods have been extensively applied and typically rely on physics-based models for state estimation or parameter estimation [40]. Qualitative model-based approaches, particularly structural-analysis-based methods, have gained increasing attention for exploiting analytical redundancy in the model structure and available sensor configuration. However, their application to LPREs remains challenging due to the large number of candidate residual generators [29]. Moreover, these qualitative approaches characterize fault sensitivity only through residual signatures, without exploiting the distinct physical mechanisms through which sensor and component faults affect system behavior. Leveraging this information could improve isolation performance.

Data-driven FDI methods provide a complementary perspective. They can be divided into unsupervised methods, which typically learn nominal patterns and allow for detecting deviations from such patterns [11, 42], and supervised methods, which assign extracted features to predefined fault classes [31, 39]. Although these methods can learn complex fault patterns, their reliability depends on representative training data and their black-box nature limits physical interpretability. Hybrid FDI methods address these limitations by retaining the model-based structure while selectively introducing data-driven components for residual generation or decision-making [21, 37]. Although such concepts have been investigated in related domains, their systematic application to LPRE FDI remains largely unexplored.

LUMEN (Liquid Upper Stage Demonstrator Engine) is a modular, pump-fed LOX/LNG rocket-engine demonstrator developed by the Institute of Space Propulsion at the German Aerospace Center (DLR). The engine has a nominal thrust of 25 kN and operates at chamber pressures between approximately 40 bar to 80 bar and mixture ratios from 3.0 to 3.8. Hot-fire testing has demonstrated stable operation over a throttling range of approximately 38 bar to 78 bar, establishing LUMEN as a representative experimental test bed for advanced control and FDI research [9]. To support method development before experimental deployment, DLR established the LUMEN Rocket Engine Control and Diagnosis Benchmark which provides an simulation-based environment for designing and evaluating control and FDI methods [8, 32]. It includes an EcosimPro/ESPSS simulation model with a steady-state accuracy of $\pm 5\%$ that enables safe and reproducible evaluation under nominal and faulty operating conditions [23]. In this work, the fuel turbopump (FTP) subsystem of LUMEN is used as a case study. The subsystem-level focus is selected as a first step toward comprehensive engine-level FDI, since turbopump faults constitute a major class of propulsion-system anomalies [4, 14].

The contributions of this work are twofold:

1. We demonstrate how data-driven residual generators and classifiers can be integrated into a qualitative model-based FDI framework to improve diagnostic performance for LPREs.
2. We introduce a sensor validation network (SVN) and combine it with structural analysis to separate sensor faults from component faults, thereby simplifying residual selection while maintaining component-level diagnostic capability.

2 Related Work

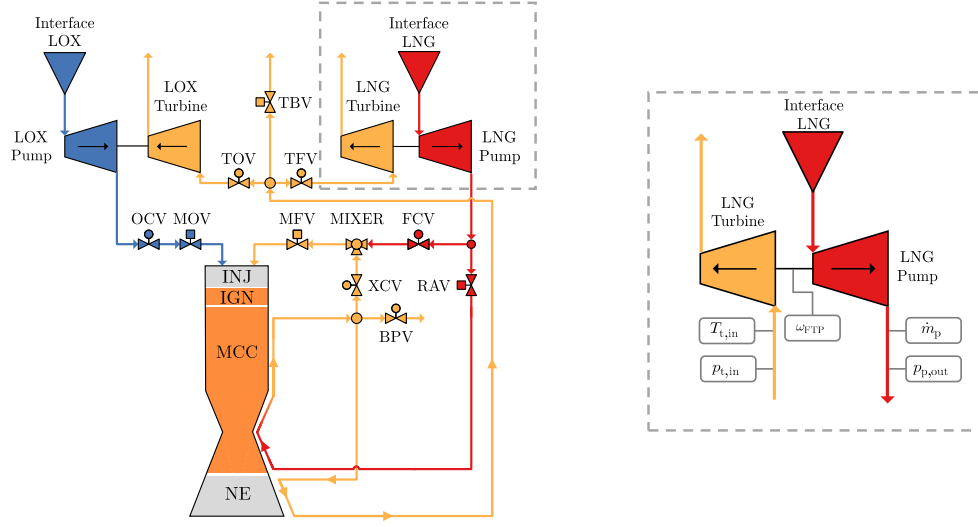
The integration of data-driven components into model-based FDI architectures has been investigated in related domains. For fault diagnosis of an internal combustion engine, neural-network-based residuals have been embedded in consistency-based diagnosis frameworks by using structural model information to define grey-box residual generators with physically interpretable inputs, outputs and state dynamics [16, 26]. Recent extensions employ neural ordinary differential equation (neural ODE) models to generate residuals for consistency-based and uncertainty-aware fault diagnosis [18, 27]. A complementary line of work combines model-based residuals with data-driven classifiers in the residual space to improve isolation performance, handle non-ideal residual signatures and support open-set diagnosis [17, 15]. Despite their demonstrated diagnostic value in these domains, such hybrid architectures have not yet been assessed for LPRE.

In LPRE applications, previous work has used Kalman filter residuals to distinguish sensor and actuator from component faults by their different effects on residual statistics [13, 24]. For diagnosis of component faults at the subsystem level, banks of Kalman filters have been employed with different parameterizations representing specific fault hypotheses [13]. Component faults have also been diagnosed using parameter-estimation-based methods, in which component faults are represented by fault factors inferred from deviations between measured engine responses and nominal model predictions [2, 24]. However, these methods for component fault diagnosis require available or reliably reconstructed mass-flow-related quantities at several engine locations, which can be restrictive for flight applications with limited instrumentation due to weight and space constraints [3].

Structural-analysis-based FDI methods incorporate the available sensor configuration directly into the structural model, yielding the detectable and isolable faults for that configuration. For complex LPRE models, however, structural analysis yields a large number of residual-generator candidates, and selection remains nontrivial given the joint constraints of fault sensitivity, causal computability and numerical conditioning [28, 29]. Separating sensor from component faults prior to residual selection removes the need for residual generators to structurally discriminate between sensor and component faults, constraining selection to component faults alone. This motivates combining fault-signature separation with structural analysis to reduce diagnostic ambiguity and identify which component faults remain diagnosable under the available sensor configuration.

3 Case Study

LUMEN, shown schematically in Figure 1, is a modular LOX/LNG expander-bleed engine with two parallel turbopumps [38]. The oxidizer turbopump (OTP) pressurizes LOX and feeds it directly to the main combustion chamber (MCC). Downstream of the LNG pump, the fuel flow splits into two paths: a smaller fraction is routed directly to the MCC and the main flow regeneratively cools the MCC in a counterflow arrangement. The heated coolant



■ **Figure 1** Flow scheme of LUMEN and the considered FTP subsystem with measurements.

is partially remixed with LNG to increase the fuel injection temperature. The remaining coolant regeneratively cools the nozzle extension (NE) before being expanded through the LOX and LNG turbines that drive the respective pumps. The turbine exhaust is then vented without combustion.

At its nominal operating point, LUMEN generates 25 kN of thrust at a combustion-chamber pressure of 60 bar and a mixture ratio of 3.4. In the latest experimental setup, the operating point is set using three electrically actuated control valves: the turbine fuel valve (TFV), the turbine oxidizer valve (TOV) and the fuel control valve (FCV). For preconditioning, ignition and safe shutdown, three on-off valves are used: the main oxidizer valve (MOV), the main fuel valve (MFV) and the regenerative cooling valve (RAV). In future test campaigns, the oxidizer control valve (OCV), mixer control valve (XCV) and bypass valve (BPV) are planned to be reimplemented to enable fault-injection capabilities for validating FDI systems.

In this work, we use the FTP subsystem as the case study. As shown in Figure 1, it comprises the LNG turbine, the mechanically coupled LNG pump and the inlet piping between the LNG interface and the pump inlet. To decouple the FTP from the rest of the engine, we use a sensor configuration consisting of the pump outlet pressure $p_{p,out}$, pump mass flow rate $\dot{m}_p$, turbopump rotational speed ω_{FTP} , turbine inlet pressure $p_{t,in}$ and turbine inlet temperature $T_{t,in}$.

3.1 Mathematical Model

Model-based FDI systems for LPREs typically use a reduced-order physics-based model, in which the engine is represented by interconnected 0-D components governed by first-principles conservation equations for mass, momentum and energy, along with other component-specific relations [1]. The resulting system of ordinary differential equations (ODEs), common in structure across LPREs, captures the engine's nonlinear dynamic behavior once its parameters are identified and calibrated against experimental data obtained during engine development and testing. In the following section, we derive the structural model of the FTP subsystem from the governing equations, following the formulation and simplifications established by Kurudzija et al. [23] through experimental evaluation and modeling of the LUMEN engine.

3.1.1 Turbine

The rotational dynamics of the FTP are described by the angular momentum balance

$$I_{\text{FTP}} \frac{d\omega}{dt} = \tau_{\text{T}} - \tau_{\text{P}}, \quad (1)$$

where ω is the rotational speed, I_{FTP} is the FTP inertia and τ_{T} and τ_{P} denote the turbine and pump torques, respectively. The turbine torque is approximated using a performance map that relates the specific torque τ_{specific} to the reduced rotational speed N_{red}

$$\tau_{\text{specific}} = \frac{\tau_{\text{T}}}{r_{\text{t}} \dot{m}_{\text{t}} a_{\text{in}}} = a + b N_{\text{red}}. \quad (2)$$

The characteristic turbine blade-tip radius is denoted by r_{t} , the turbine mass flow by $\dot{m}_{\text{t}}$, the inlet speed of sound by a_{in} and the polynomial coefficients by a and b . The inlet speed of sound is estimated as a function of temperature T and pressure p , i.e. $a_{\text{in}} = f(T, p)$. The reduced rotational speed is defined by

$$N_{\text{red}} = \frac{r_{\text{t}} \omega}{a_{\text{in}}}. \quad (3)$$

Since LUMEN uses supersonic impulse turbines, the turbine mass flow is estimated from the choked-flow conditions at the nozzle throat as

$$\dot{m}_{\text{t}} = \frac{p_{\text{t},\text{in}} A_{\text{t}}}{c^*}, \quad (4)$$

where A_{t} is the throat area and c^* is the characteristic velocity, estimated as $c^* = f(T, p)$.

3.1.2 Pump

The pump performance is described by characteristic curves expressed as functions of the outlet flow coefficient

$$\phi^+ = \frac{\dot{m}_{\text{p}}}{\rho \omega}, \quad (5)$$

with $\dot{m}_{\text{p}}$ denoting the pump mass flow. In this work, we assume the fluid density ρ to be constant. The torque coefficient λ^+ is approximated as a linear function of the outlet flow coefficient and rotational speed,

$$\lambda^+ = \frac{\tau_{\text{p}}}{\rho \omega^2} = a + b \phi^+ + c \omega, \quad (6)$$

where a , b and c are polynomial coefficients. The pressure rise across the pump is modeled with the momentum balance

$$\left(\frac{L}{A} \right) \frac{d\dot{m}_{\text{p}}}{dt} = p_{\text{in}} - p_{\text{out}} + \rho \omega^2 \psi^+. \quad (7)$$

The term L/A represents the fluid inertia, while p_{in} and p_{out} are the pump inlet and outlet pressures. The head coefficient ψ^+ is modeled as

$$\psi^+ = \frac{p_{\text{out}} - p_{\text{in}}}{\rho_{\text{in}} \omega^2} = a + b \phi^+ + c \phi^{+2} + d \omega, \quad (8)$$

where a , b , c and d are polynomial coefficients. The pressure loss in the inlet line between the LNG interface and the pump inlet is described by

$$\left(\frac{L}{A} \right) \frac{d\dot{m}_{\text{p}}}{dt} = p_{\text{if}} - p_{\text{in,p}} - k_{\text{res}} \dot{m}_{\text{p}}^2. \quad (9)$$

The parameters k_{res} and p_{if} denote the piping resistance coefficient and interface pressure, respectively. In this work, we approximate both parameters as constant model parameters.

■ **Table 1** Considered faults and ranges.

Fault	Symbol	Type	Fault Range	Unit
Pump Outlet Pressure Sensor	$f_{p_{p,out}}$	Sensor	$\pm 5, \pm 10, \pm 15$	bar
Pump Mass Flow Rate Sensor	$f_{\dot{m}_p}$	Sensor	$\pm 0.1, \pm 0.2, \pm 0.3$	kg/s
Turbopump Rotational Speed Sensor	f_{ω}	Sensor	$\pm 2, \pm 4, \pm 6$	1000/min
Turbine Inlet Pressure Sensor	$f_{p_{t,in}}$	Sensor	$\pm 3, \pm 6, \pm 9$	bar
Degraded Turbine Efficiency	f_{η}	Component	0.85, 0.90, 0.95	—
Partially opening BPV (Leakage)	f_{BPV}	Component	0.4, 0.5, 0.6	—

3.2 Fault Scenarios

Experimental fault data are typically scarce and costly to obtain for complex safety-critical systems such as LPREs. Simulation-based fault injection is therefore an established, system-agnostic approach to generate representative faulty data across a range of fault types and magnitudes [25]. We represent sensor faults as additive measurement biases, corresponding to a constant offset between the nominal sensor value and the faulty measurement, i.e.

$$y = x + f,$$

where x is the nominal sensor value and f denotes the bias magnitude. The values of the sensor biases are provided in Table 1. For simplicity, we neglect faults in the turbine inlet temperature sensor. This simplification is justified by the scope of the reduced-order FTP model, which focuses on hydraulic and rotational dynamics. The turbine inlet temperature is strongly influenced by upstream heat-transfer processes in the cooling channels, which are not resolved in the model. We introduce a component fault into the FTP subsystem by reducing the torque produced by the turbine

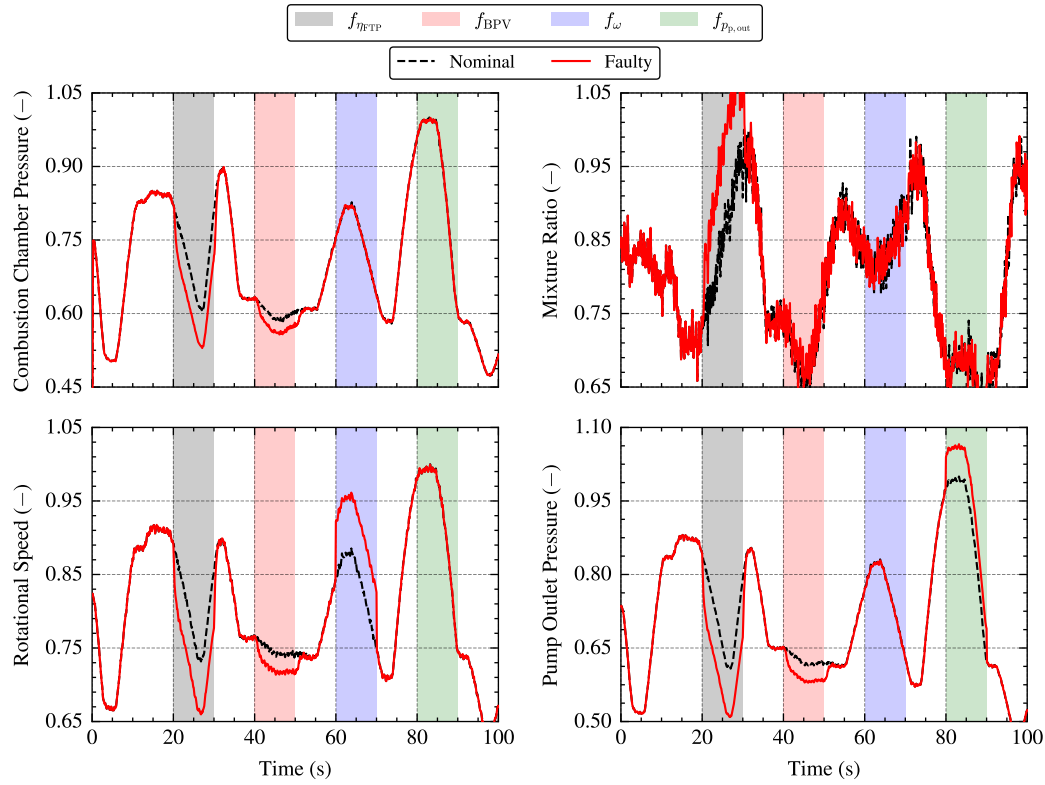
$$\tau_T = f_{\eta} \tau_{\text{specific}} r_t \dot{m}_t a_{in}, \quad f_{\eta} \in [0.85, 0.95]. \quad (10)$$

Since $f_{\eta} < 1$, this fault represents a degraded turbine efficiency. In addition, we introduce an unknown component fault by partially opening the BPV to simulate a leakage. Due to strong coupling, this fault shifts the engine's operating point. However, since the governing equations of the FTP subsystem remain internally consistent under this fault, the FDI system should be insensitive to it. From the FTP perspective, the subsystem remains consistent with nominal operation. Figure 2 shows the influence of different fault types on various sensors. Although f_{η} increases the mixture ratio, it leads to a reduction in combustion chamber pressure, pump outlet pressure and rotational speed. In contrast, f_{BPV} has only a minor effect on the mixture ratio, but also reduces combustion chamber pressure, pump outlet pressure and rotational speed. Since the engine is controlled in an open loop, a sensor fault affects only the corresponding measurement. In total, we consider six faults, as summarized in Table 1.

4 Methodology

4.1 Structural Analysis

Structural analysis is applied to the mathematical model, defined by the equation–variable incidence relations [12]. Let E denote the set of model equations and Z the set of unknown variables. The structural model is represented as a bipartite graph $G = (E, Z, A)$, with

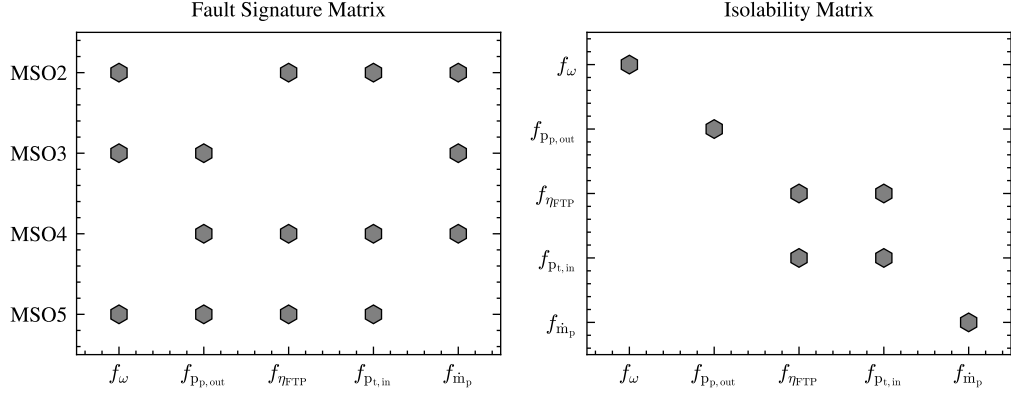


■ **Figure 2** Influence of different fault types on normalized sensor signals. Each signal is normalized by the maximum value of the nominal trajectory.

$(e_i, z_j) \in A$ whenever z_j occurs in e_i . A subset $M \subseteq E$ is structurally overdetermined if M contains more equations than unknown variables, i.e. $|M| > |Z(M)|$, where $Z(M) \subseteq Z$ denotes the unknown variables contained in M . A structurally overdetermined subset M is minimally structurally overdetermined (MSO) if no proper subset $M' \subset M$ is structurally overdetermined [22]. The Dulmage–Mendelsohn decomposition is used to extract the structurally overdetermined part of the model, from which the complete set of MSO sets is derived. These MSO sets constitute residual-generator candidates for structural detectability and isolability analysis. Structural analysis is performed using the Fault Diagnosis Toolbox [12]. For isolability assessment, the considered faults and the measurements are incorporated into the mathematical model. The resulting structural model comprises 18 unknown variables and 20 equations, yielding a redundancy degree of 2 and 5 MSO sets. A simple, minimal hitting set algorithm is then used to select the smallest set of tests that achieves the maximum attainable isolability performance [6]. Figure 3 presents the resulting fault signature and isolability matrix. With four MSO sets, all considered faults are structurally detectable. Since the degraded turbine efficiency fault f_η and the turbine inlet pressure sensor fault $f_{p_{t,in}}$ produce identical fault signatures in the selected MSO sets, they are structurally indistinguishable.

4.2 Residual Generators

Different methods exist for deriving residual generators from an MSO set. A standard approach is to select one equation of the MSO set as the redundant equation, while the remaining equations form an exactly determined system with an equal number of equations



■ **Figure 3** Results of the structural analysis.

■ **Table 2** Derived residual generators and computational causality based on the derived fault signature matrix.

MSO	Redundant Equation	Measurements	Causality	Model
2	$y_\omega - \hat{y}_\omega$	$y_{p_{t,in}}, y_{T_{t,in}}, y_{m_p}, y_\omega$	Integral	UKF
3	$y_{p_{p,out}} - \hat{y}_{p_{p,out}}$	y_{m_p}, y_ω	Algebraic	SRG
4	$y_{p_{p,out}} - \hat{y}_{p_{p,out}}$	$y_{p_{t,in}}, y_{T_{t,in}}, y_{m_p}, y_{p_{p,out}}$	Integral	UKF
5	$y_\omega - \hat{y}_\omega$	$y_{p_{t,in}}, y_{T_{t,in}}, y_{p_{p,out}}, y_\omega$	Integral	neural ODE

and unknowns. By selecting a measurement equation as the redundant equation, the residual generator compares the model prediction of the measured variable with its observed value. Under nominal conditions, the resulting residual is expected to remain close to zero, i.e. $y - \hat{y} \approx 0$. The computational graph of a residual generator is defined by the sequence in which the unknown variables are computed. Integral causality is obtained if the state variables are computed by integration, while derivative causality is obtained if the state variables are differentiated. If the MSO set contains no dynamic states, the resulting residual generator is algebraic.

In this work, we avoid applying derivative causality to the residual generators, as they require numerical differentiation and are sensitive to measurement noise [20]. Therefore, we select the redundant equation from the MSO sets to obtain residual generators with integral or algebraic causality. Residual generators with integral causality are implemented using a state observer. As the model equations are nonlinear, an unscented Kalman filter (UKF) is employed. UKF-based residual generation avoids explicit model linearization and has previously been investigated for the design of residual generators in LPRE [29]. To enable high fault sensitivity of the UKF residual generators, we choose the process noise covariance matrix to be smaller than the measurement noise covariance matrix.

To design the model-based residual generators, we approximate the mass-flow dynamics (see Eqs. 7 and 9) as quasi-static. The corresponding momentum-balance equations are, therefore, reduced by setting $dm/dt = 0$. This approximation is motivated by the characteristic time scale of the momentum-balance dynamics in the mathematical model, which is small relative to the 10 Hz sampling interval of the generated data and, therefore, not resolved. The resulting residual generator for MSO3 has algebraic causality and is implemented as a sequential residual generator (SRG). For MSO5, this approximation yields a computational

sequence that requires numerically inverting the nonlinear relationship between flow and the head coefficient (see Eq. 8), which can become ill-conditioned. To avoid these numerical issues, we use a data-driven residual generator as an alternative. For this, we implement a neural ordinary differential equation (neural ODE) residual generator, similar to the architecture in Jung and Westny [18]. The neural ODE residual generator is formulated as a nonlinear state-space model,

$$\begin{aligned}\dot{x}(t) &= \hat{g}(x(t), u(t); \theta_g), \\ \hat{y}(t) &= \hat{h}(x(t); \theta_h), \\ r(t) &= y(t) - \hat{y}(t).\end{aligned}\tag{11}$$

with dynamic state $x(t) \in \mathbb{R}^{n_x}$, inputs $u(t) \in \mathbb{R}^{n_u}$, measured output $y(t) \in \mathbb{R}^{n_y}$, predicted measurement $\hat{y}(t) \in \mathbb{R}^{n_y}$ and residual $r(t) \in \mathbb{R}^{n_y}$. The transition function $\hat{g}$ and the measurement function $\hat{h}$ are parameterized by multilayer perceptrons (MLPs) with trainable parameters θ_g and θ_h , respectively. The state dynamics are integrated over each sampling interval using a fourth-order Runge–Kutta scheme with linear interpolation between samples.

The residual generators used in this work, their redundant equation, input measurements, causality and model type are summarized in Table 2.

4.3 Sensor Validation Network

As discussed in Section 2, sensor and component faults affect the engine in different ways. A sensor fault corrupts the affected measurement, whereas a component fault shifts the engine’s operating point, resulting in correlated deviations across multiple sensors. The commanded positions of the three control valves TOV, TFV and FCV set the operating point of LUMEN. For fixed model parameters and in the absence of unmodeled disturbances, the valve-command trajectory defines the engine’s nominal behavior and, consequently, the expected sensor outputs at each time step. Based on this observation, we introduce a sensor validation network (SVN) that predicts the nominal sensor output from the control-command inputs. The SVN uses the neural ODE architecture introduced in Section 4.2. The state-transition function $\hat{g}$ propagates the latent state using the valve-command trajectory, while the output map $\hat{h}$ maps the latent state x to the predicted sensor outputs $\hat{y}$.

We perform sensor fault diagnosis by comparing the measured sensor vector with the corresponding SVN prediction. For the n_s considered sensors, the residual vector is defined as

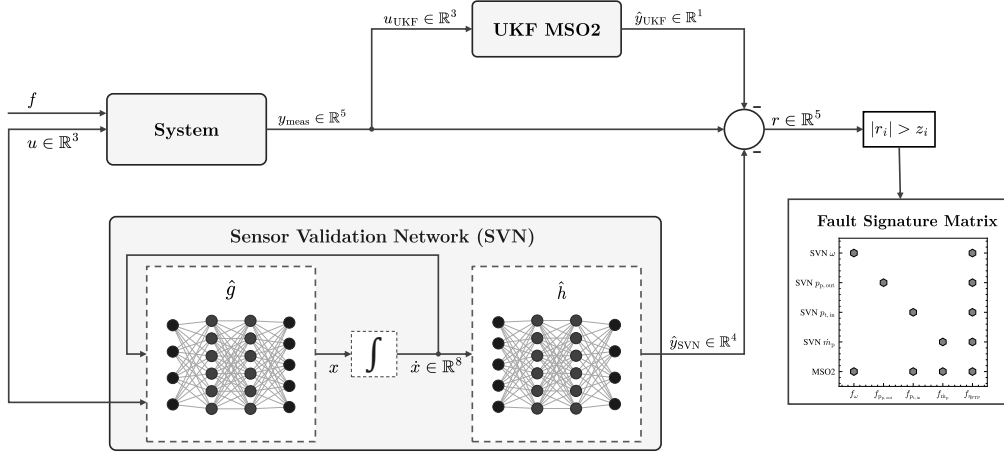
$$\mathbf{r}_{\text{SVN}}(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t), \quad \mathbf{r}_{\text{SVN}}(t) \in \mathbb{R}^{n_s}.\tag{12}$$

Since the SVN is driven by control commands rather than by measured signals, sensor faults appear as deviations between $\mathbf{y}$ and $\hat{\mathbf{y}}$.

We then combine the SVN with the structural analysis results to diagnose component faults. Specifically, we use MSO2, which is structurally sensitive to degraded turbine efficiency, together with the SVN-based sensor residuals. This yields distinct signatures for sensor faults, a degraded turbine efficiency as well as unknown component faults. Figure 4 schematically illustrates the resulting diagnostic architecture.

4.4 Fault Detection and Diagnosis

The FDI system operates in two stages on the residuals $\mathbf{r}(t) \in \mathbb{R}^{n_r}$, where n_r denotes the number of residuals derived from structural analysis. The detection stage decides whether the system is in a faulty or nominal operational state. For this, a per-residual two-sided



■ **Figure 4** Schematic overview of the SVN-based diagnosis architecture.

hypothesis test is applied to the standardized residuals, flagging residual i when

$$\frac{|r_i(t) - \mu_{0,i}|}{\sigma_{0,i}} > z_{1-\alpha/2}, \quad (13)$$

where $\mu_{0,i}$ and $\sigma_{0,i}$ denote the mean and standard deviation of residual i under nominal operation and $z_{1-\alpha/2}$ the $(1 - \alpha/2)$ -quantile of the standard normal distribution. A fault is detected when at least one residual is triggered, i.e. $d(t) = \mathbf{1}[C(t) \neq \emptyset]$. This provides a set of triggered residuals $C(t) \subseteq \{r_1, \dots, r_{n_r}\}$, which are evaluated in the next stage.

If a fault is detected, fault isolation is performed in the second stage to compute the set of diagnosis candidates. For isolation, we evaluate two different approaches. The first follows a consistency-based fault isolation logic (CBD) [7, 34]. Following the structural derivation of the residuals, the single-fault setting and a calibrated detection stage, the exoneration and no-false assumption are justified. Therefore, f_k is admitted as a diagnosis candidate iff $C(t) = S_k$, where $S_k \subseteq \{r_1, \dots, r_{n_r}\}$ denotes the set of residuals expected to trigger under fault f_k , yielding the candidate set

$$\mathcal{D}(t) = \{f_k \in \mathcal{F} \mid C(t) = S_k\}. \quad (14)$$

The limitation of this logic for the FTP subsystem is that the degraded turbine efficiency fault f_η is structurally not isolable from the turbine inlet pressure sensor fault $f_{p_{t,\text{in}}}$. To use information beyond the binary structural fault signature, we use a data-driven isolation logic based on a set of fault-specific one-class support vector machines (OC-SVMs), similar to Jung et al. [17]. OC-SVMs are trained on samples from a single class only and learn a decision boundary around this class in the feature space, classifying new observations outside the boundary as novel or anomalous [35]. For each fault $f_k \in \mathcal{F}$, an OC-SVM is trained on residual data generated under that fault, learning a closed acceptance region that encloses the corresponding residual vectors and thereby models a single fault hypothesis. A fault is selected as a diagnosis candidate if the residual vector lies in the acceptance region of the corresponding classifier, i.e.

$$\mathcal{D}(t) = \{f_k \in \mathcal{F} \mid s_k(\mathbf{r}(t)) \geq 0\}, \quad (15)$$

where $s_k(\mathbf{r})$ denotes the decision score of the classifier trained for fault f_k and $s_k(\mathbf{r}) \geq 0$ indicates that the residual vector is consistent with the corresponding fault hypothesis. If the fault-specific distributions define separable regions in the residual space, the corresponding fault modes can be isolated as distinct candidates. If these regions overlap, the affected fault hypotheses remain jointly in the candidate set $\mathcal{D}(t)$. An empty candidate set yields an *unknown* diagnosis, i.e. a fault is detected in the first stage, but no candidate is accepted in the isolation stage.

To reduce the influence of transient effects on the diagnosis, we stabilize both detection and isolation decisions by M -of- N persistence filters [15]. These filters aggregate the most recent decisions over a sliding window of length N , accepting a decision as persistent only if it occurred in at least M of these samples [19].

4.5 Evaluation Metrics

We evaluate the FDI system using four metrics, two for the detection stage and two for the end-to-end isolation. Let $\mathcal{N}$ denote the set of samples generated under nominal operation and $\mathcal{F} = \bigcup_k \mathcal{F}_k$ the set of samples generated under a known fault, where $\mathcal{F}_k$ contains the samples corresponding to fault f_k . The detection stage is evaluated based on the false alarm rate (FAR) and the true detection rate (TDR),

$$\text{FAR} = \frac{1}{|\mathcal{N}|} \sum_{t \in \mathcal{N}} \mathbf{1}[d(t) = 1], \quad \text{TDR} = \frac{1}{|\mathcal{F}|} \sum_{t \in \mathcal{F}} \mathbf{1}[d(t) = 1], \quad (16)$$

where $d(t) \in \{0, 1\}$ is the detection indicator. The end-to-end isolation performance is evaluated based on the isolation accuracy IA and its top- k relaxation IA_k ,

$$\text{IA} = \frac{1}{|\mathcal{F}|} \sum_{t \in \mathcal{F}} \mathbf{1}[\hat{c}(t) = c(t)], \quad \text{IA}_k = \frac{1}{|\mathcal{F}|} \sum_{t \in \mathcal{F}} \mathbf{1}[c(t) \in \mathcal{D}(t) \wedge |\mathcal{D}(t)| \leq k], \quad (17)$$

where $\hat{c}(t)$ is the final diagnosis label, $\mathcal{D}(t)$ the diagnosis candidate set and $c(t)$ the ground-truth fault class. In this work we use IA_2 , i.e. accepting an ambiguity group of up to two candidates as a successful diagnosis.

4.6 Data Generation

The data generation process aims to produce representative simulated trajectories comparable to those used during ground tests while providing denser coverage of the operational domain. For this, we split the process into two stages. In the first stage, steady-state data are uniformly generated within LUMEN's operational envelope using Latin Hypercube Sampling (LHS). To set the sampled operating point, two PI controllers are designed for TOV and TFV. The position of FCV is set as a function of the operating point in the open loop. In the second stage, transient trajectories are generated by connecting randomly selected steady-state operating points with ramps with randomized transition times. The resulting load point is held for a randomly sampled dwell time. To increase data diversity, the FCV position is randomly perturbed within a $\pm 5\%$ range. Each generated trajectory has a duration of 100 s. For data generation, the simulation model is initialized from a steady-state solution and simulated in open loop using sampled valve positions with a timestep of 0.1 s.

For training and validation of the data-driven isolation stage, we generate 30 trajectories for each fault class and for the nominal class in each data split. The faults are introduced at $t = 20$ s as steps and remain active for the rest of the trajectory, simplifying fault labeling and evaluation.

4.7 Implementation Details

Both the neural ODE residual generator and SVN are trained on 250 fault-free trajectories. Each trajectory is segmented into overlapping windows of approximately 50 s length with 50 % overlap. All inputs and outputs are standardized to have zero mean and unit variance, based on the training set statistics. Following Mohammadi et al. [27], we use a curriculum-learning strategy, where the trajectory-window length is increased from 20 s to approximately 50 s using a piecewise-linear schedule. Model parameters are optimized using Adam to minimize mean squared error, with an initial learning rate of 10^{-3} that decays exponentially by a factor of 0.99 per epoch. Early stopping is applied based on the validation loss.

4.7.1 State Initialization

For the neural ODE residual generator, we use a noise-aware initialization, similar to that of Mohammadi et al. [27]. During training, the initial dynamic state is sampled as $x_0 \sim \mathcal{N}(y_0, \sigma^2)$, centered at the first measurement of each trajectory window with σ the measurement-noise standard deviation. During inference, the first measurement initializes the state directly. For SVN, we use a different initialization strategy. Assuming that the simulation starts from a steady-state solution, the initial state is fixed to $x_{t0} = 0$. This assumption is justified by the results in Kurudzija et al. [23], which demonstrate that the engine dynamics can be separated into slow thermal transients and faster fluid dynamics. As the simulation model is initialized from a thermal steady-state solution (see Section 4.6), the surrogate model is trained to capture the system's fast dynamics, which reach a steady-state solution rapidly after changes in the valve commands. To avoid penalizing initialization transients, the loss is evaluated after an initial 2 s warm-up period within each trajectory window.

4.7.2 Hyperparameters

The architecture and training hyperparameters of both neural-ODE-based models are selected empirically. The selection is based on validation set performance using 50 fault-free trajectories, with the objective of balancing training time and long-horizon rollout accuracy. The selected MLP architecture consists of two hidden layers, each with 128 neurons. For SVN, the latent-state dimension is set to 8. Following Jung and Westny [18], SiLU activations are used in $\hat{g}$, while $\hat{h}$ uses ReLU activations with layer normalization after each nonlinear layer. The hyperparameters of the detection and isolation stages are selected by a grid search on the validation set consisting of 30 trajectories for each fault and the nominal class. The detection threshold is chosen to satisfy $\text{FAR} \leq 0.02$ while maximizing TDR. For the set of OC-SVMs, we use the RBF kernel and select its hyperparameters ν and γ to maximize the per-class isolation accuracy. The parameter ν upper-bounds the fraction of training samples allowed to lie outside the decision boundary, controlling the trade-off between boundary tightness and tolerance to training outliers, while γ determines how far the influence of an individual training sample extends, with larger values producing a more localized and flexible decision boundary. To ensure a consistent evaluation of the isolation stage, we apply the same persistence filter parameters to all architectures, with $M = 6$ and $N = 10$.

5 Results and Discussion

We evaluate the proposed FDI architectures on a test set of 30 trajectories per fault class and 30 nominal trajectories, using the metrics introduced in Section 4.5 to quantify detection and end-to-end isolation performance. We further assess performance on the unknown fault

■ **Table 3** Detection and isolation performance of the evaluated FDI systems. The best value for each metric is highlighted in bold.

System	FAR	TDR	IA	IA ₂
Hybrid	0.00	0.96	0.76	0.94
CBD	0.00	0.96	0.46	0.85
SVN	0.01	0.99	0.92	0.92

class and analyze the time-resolved diagnostic output along an exemplary trajectory. The following abbreviations denote the evaluated FDI systems:

- Hybrid: fault isolation using an OC-SVM per fault hypothesis, applied to MSO-derived residuals.
- CBD: fault isolation via consistency-based diagnosis, applied to MSO-derived residuals.
- SVN: fault isolation via consistency-based diagnosis, applied to residuals from the proposed sensor validation network combined with MSO2.

For the SVN architecture, the data-driven isolation stage is omitted, as the SVN residuals render the considered fault classes structurally isolable.

5.1 Detection and Isolation Performance

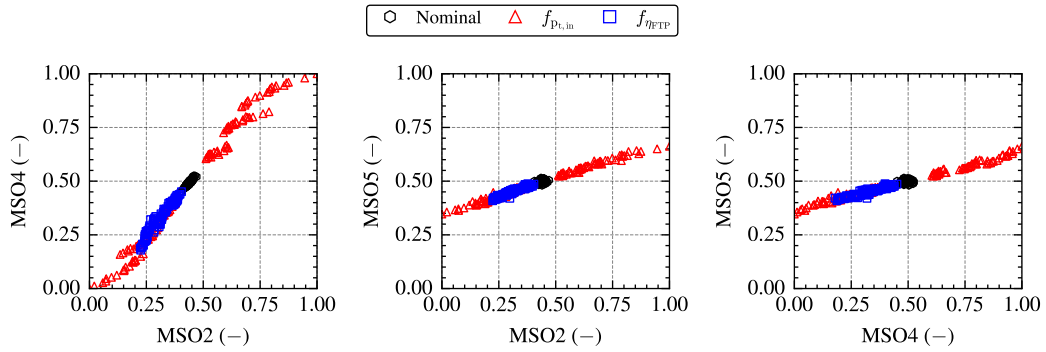
Table 3 summarizes the detection and isolation performance of the FDI systems. While the detector operating on MSO residuals achieves a perfect false alarm rate ($\text{FAR} = 0.00$), its true detection rate is slightly lower than that of the SVN-based detector, i.e. $\text{TDR} = 0.96$ compared to 0.99. SVN achieves a FAR of 0.01, indicating that both detectors perform comparably overall. The isolation performance varies substantially across the three architectures. As expected, the CBD baseline achieves the lowest isolation accuracy ($\text{IA} = 0.46$) as f_η and $f_{p_{t,\text{in}}}$ are not structurally isolable. Hybrid yields a substantial improvement to $\text{IA} = 0.76$. This indicates that the residual distributions of f_η and $f_{p_{t,\text{in}}}$ are sufficiently separated for data-driven isolation. SVN achieves the highest isolation accuracy at $\text{IA} = 0.92$. Allowing an ambiguity group of two candidates partly closes the gap for Hybrid and CBD. Hybrid achieves the highest isolation accuracy ($\text{IA}_2 = 0.95$), followed by SVN ($\text{IA}_2 = 0.92$) and CBD ($\text{IA}_2 = 0.85$). For CBD, the gap between $\text{IA} = 0.46$ and $\text{IA}_2 = 0.85$ reflects the structural indistinguishability of f_η and $f_{p_{t,\text{in}}}$. For SVN, $\text{IA} = \text{IA}_2 = 0.92$ shows that the candidate set is consistently a singleton, with more discriminative residual generation enabling unique isolation. For Hybrid, the increase from $\text{IA} = 0.76$ to $\text{IA}_2 = 0.94$ suggests that the data-driven isolator resolves the structural indistinguishability for most samples but leaves an ambiguity in a small fraction of cases, where the distributions of f_η and $f_{p_{t,\text{in}}}$ overlap in the residual subspace and both faults remain candidates.

This is illustrated in Figure 5. The residual response of $f_{p_{t,\text{in}}}$ exhibits two directions depending on the fault sign, i.e. a positive and a negative bias. On the other hand, the residual response of f_η is unidirectional. The overlap between the two fault classes in the residual space occurs specifically when $f_{p_{t,\text{in}}}$ exhibits a sensor fault with positive bias, as highlighted in Table 4. Hybrid achieves correct and unambiguous isolation whenever $f_{p_{t,\text{in}}}$ has a negative bias, but fails to do so when it has a positive bias. However, the isolation performance increases with fault magnitude in the positive-bias region, as larger magnitudes shift the residual vector further into the separable region of the residual space. Specifically, the isolation accuracy increases from $\text{IA} = 0.09$ at +3 bar to $\text{IA} = 0.95$ at +9 bar. For

■ **Table 4** Fault-magnitude-wise isolation performance (IA) for $f_{p_{t,in}}$ and f_{η} . The best value for each fault magnitude is highlighted in bold. The CBD architecture is not shown as these faults are not structurally isolable.

Model	$f_{p_{t,in}}$ (bar)						$f_{\eta}(-)$		
	+3	-3	+6	-6	+9	-9	0.95	0.90	0.85
Hybrid	0.09	0.95	0.57	0.95	0.95	0.93	0.52	0.07	0.38
SVN	0.99	0.99	0.99	0.99	0.99	0.99	0.97	0.99	0.99

f_{η} , the per-magnitude isolation accuracy exhibits a less consistent pattern, with IA = 0.07 at $f_{\eta} = 0.90$, IA = 0.52 at $f_{\eta} = 0.95$, and IA = 0.38 at $f_{\eta} = 0.85$. In contrast, SVN achieves correct and unambiguous isolation across all magnitudes and fault classes with near-perfect accuracy (IA = 0.99). This is consistent with the design of the SVN architecture and indicates that incorporating fault-discriminative information into the residual generation stage substantially improves isolation performance for structurally similar faults.

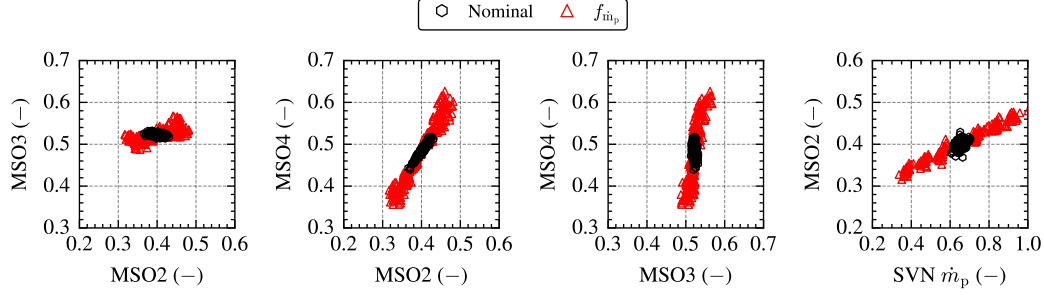


■ **Figure 5** Response of $f_{p_{t,in}}$ and f_{η} in the residual subspace. For better readability, the residuals are normalized and only every 100-th point is illustrated.

Table 5 shows the per-fault isolation performance. As discussed, the performance gap between the three configurations is most pronounced for $f_{p_{t,in}}$ and f_{η} . For f_{ω} and $f_{p_{p,out}}$, all three configurations achieve near-perfect isolation accuracy (IA $\approx$ 0.99), indicating that these faults are well-separated. For $f_{\dot{m}_p}$, the performance of the three configurations differs. Hybrid achieves the highest isolation accuracy (IA = 0.83), while CBD and SVN reach only 0.34 and 0.63, respectively. This difference can be attributed to the sensitivity of the isolation mechanism to fault magnitude, as illustrated in Figure 6. At small magnitudes, $f_{\dot{m}_p}$ does not produce a sufficiently strong response in the residuals to trigger the required crossing pattern and, therefore, fails to admit $f_{\dot{m}_p}$ as a diagnosis candidate. As a result, both architectures provide a *unknown* diagnosis. In contrast, the hybrid architecture operates directly on the residual vector $\mathbf{r}(t)$. The per-fault classifiers were trained on $f_{\dot{m}_p}$ data across all magnitudes and can therefore identify the fault even when the residual response is weak. This illustrates a key advantage of data-driven isolation: by decoupling fault isolation from the detection threshold, the per-fault classifier retains sensitivity to low-magnitude faults that would otherwise go undetected by threshold-based crossing logic.

■ **Table 5** Per-fault isolation performance. The best value for each fault class is highlighted in bold.

System	IA				
	f_ω	$f_{p_{t,in}}$	$f_{p_{p,out}}$	f_η	$f_{\dot{m}_p}$
Hybrid	0.96	0.74	0.97	0.32	0.83
CBD	0.99	0.00	0.99	0.00	0.34
SVN	0.99	0.99	0.99	0.99	0.63



■ **Figure 6** Response of $f_{\dot{m}_p}$ in the residual subspace. For better readability, the residuals are normalized and only every 100-th point is illustrated.

5.2 Unknown Fault Performance

Table 6 reports the performance of all three configurations for the unknown fault class f_{BPV} across different fault magnitudes. The fault magnitude $f_{BPV} \in [0.4, 0.6]$ denotes the relative opening of BPV. The architectures exhibit different behaviors for the unknown fault class. Since the derived MSOs for Hybrid and CBD insensitive to the unknown fault class, the detection stage should not be triggered, as the FTP is operating nominally. Both architectures effectively reject the unknown fault class during the detection stage, resulting in zero false alarms ($FAR_{unk} = 0.00$). This confirms that the MSO-based architectures do not misclassify the unknown fault class. Further, this implies that the fault is undetected within the FDI system for this subsystem, which is desirable for a distributed diagnosis system where this fault class is handled by another local diagnosis system.

In contrast, SVN is sensitive to this fault class, as it produces a correlated deviation across multiple sensor signals, captured by the SVN residuals. Since MSO2 is structurally insensitive to this fault, the isolation logic under the exoneration and no-false-alarm assumptions produces an empty candidate set, which the pipeline resolves as an *unknown* diagnosis. SVN achieves $IA_{unk} = 0.98$, confirming reliable detection of the unknown fault class. The marginal shortfall is attributable to the warm-up phase of the persistence filter, during which the first $N - 1$ samples of each fault trajectory do not yet contribute a fully populated sliding window, slightly reducing the per-sample rejection rate below unity.

5.3 Trajectory-Level Performance

To evaluate the trajectory-level performance, the exemplary test run introduced in Section 3.2 is used. The run consists of four sequential fault episodes: $f_\eta = 0.9$ is active during $t \in [20, 30]$ s, $f_{BPV} = 0.5$ during $t \in [40, 50]$ s, $f_\omega = 4000$ 1/min during $t \in [60, 70]$ s and $f_{p_{p,out}} = 10$ bar during $t \in [80, 90]$ s. As Hybrid and SVN achieve the highest isolation performance, their qualitative diagnostic output over time for the test run is shown in

■ **Table 6** Performance on the unknown fault class. The best value for each fault class is highlighted in bold.

System	Metric	$f_{\text{BPV}} = 0.5$	$f_{\text{BPV}} = 0.5$	$f_{\text{BPV}} = 0.6$
Hybrid	FAR_{unk}	0.00	0.00	0.01
CBD	FAR_{unk}	0.00	0.00	0.01
SVN	$1 - \text{IA}_{\text{unk}}$	0.02	0.02	0.02

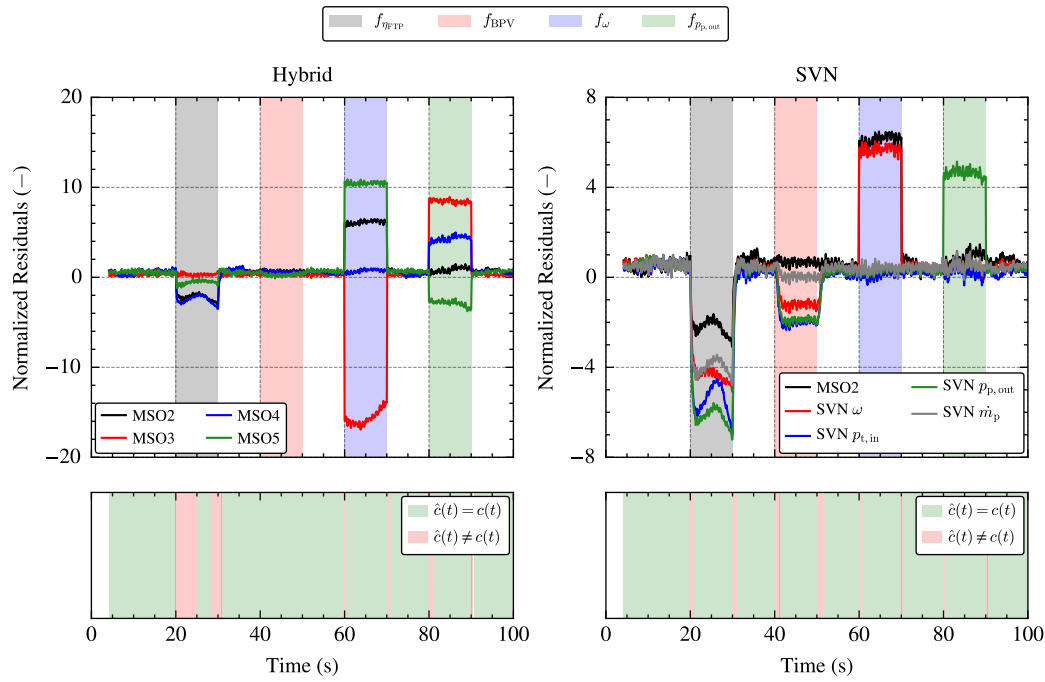
Figure 7. Within the first nominal segment $t \in [0, 20]$ s, both architectures correctly detect nominal operation. When f_η is introduced, the residuals of both architectures deviate from zero. SVN correctly diagnoses this fault after the warm-up phase of the persistence filter, while Hybrid fails to isolate it correctly due to the overlap between f_η and $f_{p_{t,\text{in}}}$ in the residual space (see Section 5.1). In the subsequent nominal segment, the return to a nominal diagnosis is briefly delayed for both architectures as a result of the persistence filter. When the unknown fault f_{BPV} is introduced, the SVN residuals deviate from zero, and the unknown fault class is correctly diagnosed after the warm-up phase of the persistence filter. Hybrid does not respond to this fault class. As a result, no warm-up phase is required, and the correct diagnosis is maintained throughout. Again, both architectures correctly classify nominal operation in the following segment. When f_ω is introduced, both architectures correctly diagnose the fault after the warm-up phase of the persistence filter. Notably, the residual response of the hybrid architecture is stronger for f_ω than for f_η , indicating a higher fault sensitivity. The following nominal segment is again correctly classified by both architectures. In the final fault segment, $f_{p_{t,\text{out}}}$ is correctly diagnosed by both architectures after the warm-up phase of the persistence filter.

Exploiting differences between sensor- and component-fault signatures, SVN correctly and unambiguously isolates all fault episodes, including structurally indistinguishable and unknown classes, where Hybrid and CBD reach their limits.

5.4 Generalizability

The scope of the proposed approach is generalization across LPREs rather than across arbitrary complex systems. Within this scope, the methodology's requirements are satisfied by construction. The governing equations of the reduced-order model follow from first-principles conservation laws and are therefore known. Validated transient simulation models are firmly embedded in the engine development process and thus available for FDI design [25]. The dominant fault mechanisms of LPREs are likewise well documented [4, 14], providing the basis for fault modeling in both the structural analysis and the generation of training data. Since the OC-SVM and SVN training data are also obtained from the validated simulation model, the data-driven components inherit the same availability as the structural model. The methodology can thus be adapted to other engine configurations by exchanging the underlying simulation model and training data, without modifying the diagnostic architecture.

Extending the approach from the FTP subsystem to the full engine follows the same workflow. The structural model is augmented with the governing equations of the remaining components, the detectability and isolability analysis is repeated for the extended fault set and sensor configuration and the data-driven components are retrained on correspondingly generated data. As structural isolability depends on the sensor configuration and fault set, the attainable diagnostic performance must be re-assessed for each target system.



■ **Figure 7** Trajectory-level performance of the hybrid architecture and SVN for multiple faults. Each residual is normalized using the fault-free first 20 s of the trajectory.

6 Conclusion and Outlook

In this work, we demonstrated that incorporating data-driven components into a model-based framework for fault detection and isolation improves the performance for the FTP subsystem of the LOX/LNG rocket engine. In the residual-generation stage, we selectively employ neural ODE residual generators for MSO sets for which direct model-based residual generation would require numerically ill-conditioned inversions. By parameterizing the system dynamics in grey-box form, additional MSO sets can be made accessible, thereby maximizing structural isolability. To address the structural indistinguishability between two of the five considered faults, we employ a data-driven isolation layer based on a set of one-class classifiers. Each classifier is trained on residual data from a single fault class, enabling the diagnosis system to account for distributional differences not captured by structural fault signatures alone. The hybrid architecture substantially improves isolation accuracy (IA) compared with the consistency-based baseline, increasing the IA from 0.46 to 0.76.

We further demonstrate that exploiting differences between sensor- and component-fault signatures yields an additional improvement in diagnostic performance, achieving an isolation accuracy of 0.92. For this purpose, we employed a grey-box neural ODE surrogate driven by control-command inputs, which serves as a sensor validation network (SVN). This fault decoupling enables local structural analysis for component-fault diagnosis, resulting in targeted diagnostic modules for selected component faults. The SVN architecture further improves the isolation accuracy to 0.92, while retaining the ability to detect unknown component faults.

Future work will extend our proposed approach to a distributed diagnosis system that covers all LUMEN subsystems. In addition, the approach will be demonstrated under closed-loop engine operation and extended to gradual fault drifts. The simulation-to-reality

gap will be addressed by incorporating slow thermal transients into the neural ODE models and adapting the surrogates to experimental data. Finally, the performance of the resulting FDI system will be validated with experimental data.

References

- 1 J. Cha. Numerical Simulation of Chemical Propulsion Systems: Survey and Fundamental Mathematical Modeling Approach. *Aerospace*, 10(10), 2023. doi:10.3390/aerospace10100839.
- 2 J. Cha and S. Ko. General Fault Factor Method for Health Monitoring of a Liquid Rocket Engine. *Journal of Spacecraft and Rockets*, 62(6):2236–2248, 2025. doi:10.2514/1.A36337.
- 3 R. L. Christenson, M. A. Nelson, and J. P. Butas. Rocket Engine Health Management: Early Definition of Critical Flight Measurements. In *39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, Huntsville, Alabama, 2003. doi:10.2514/6.2003-5251.
- 4 H. A. Cikanek. Characteristics of Space Shuttle Main Engine Failures. In *23rd Joint Propulsion Conference*, San Diego, California, June 1987. AIAA Paper 87-1939. doi:10.2514/6.1987-1939.
- 5 M. Davidson and J. Stephens. Advanced Health Management System for the Space Shuttle Main Engine. In *40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, Fort Lauderdale, FL, July 2004. AIAA 2004-3912.
- 6 J. de Kleer. Hitting Set Algorithms for Model-Based Diagnosis. In *22nd International Workshop on Principles of Diagnosis (DX'11)*, Murnau, Germany, 2011.
- 7 J. de Kleer and B. C. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32(1):97–130, 1987. doi:10.1016/0004-3702(87)90063-4.
- 8 K. Dresia, E. Kurudzija, V. Bareiß, J. Dauer, J. Deeken, and G. Waxenegger-Wilfing. Health Monitoring and Intelligent Control Benchmark for Reusable Liquid-Propellant Rocket Engines. In *Space Propulsion Conference*, Bari, Italy, 2026. 3AF. doi:10.60711/SPC2026.20260715.37863079794095352.
- 9 K. Dresia, T. Traudt, M. Börner, D. Suslov, W. Armbruster, R. dos Santos Hahn, E. Kurudzija, C. Groll, M. A. Müller, S. Klein, J. Hämisch, J. Deeken, J. Hardi, and S. Schlechtriem. Hot-Fire Testing and System Analysis of the LUMEN Liquid Upper Stage Demonstrator Engine. In *3rd Int. Conf. on Flight Vehicles, Aerothermodynamics and Re-entry (FAR)*, Arcachon, France, 2025.
- 10 European Commission. ENLIGHTEN: European iNitiative for Low Cost, Innovative & Green High Thrust Engine, October 2022. doi:10.3030/101082326.
- 11 B. Féraud, S. Le Gonidec, M. Galeotta, S. Oriol, and S. Dreyer. Anomaly Detection on Propulsive Systems by Global Approach Using Autoencoders. *IFAC-PapersOnLine*, 54(4):31–37, 2021. doi:10.1016/j.ifacol.2021.10.006.
- 12 E. Frisk, M. Krysander, and D. Jung. A Toolbox for Analysis and Design of Model Based Diagnosis Systems for Large Scale Models. *IFAC-PapersOnLine*, 50(1):3287–3293, 2017. 20th IFAC World Congress. doi:10.1016/j.ifacol.2017.08.504.
- 13 N. T. Ho. Failure Detection and Isolation for the Space Shuttle Main Engine. M.sc. thesis, Massachusetts Institute of Technology, 1998.
- 14 M. M. Hussein, D. Terakado, and O. de Weck. Comprehensive Study of the International Space Launch Industry: Programmatic Analysis and Technical Failures. *Acta Astronautica*, 234:676–689, 2025. doi:10.1016/j.actaastro.2025.03.015.
- 15 D. Jung. Data-Driven Open-Set Fault Classification of Residual Data Using Bayesian Filtering. *IEEE Transactions on Control Systems Technology*, 28(5):2045–2052, 2020. doi:10.1109/TCST.2020.2997648.
- 16 D. Jung. Automated Design of Grey-Box Recurrent Neural Networks For Fault Diagnosis using Structural Models and Causal Information. *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, 168:8–20, 23–24 June 2022. URL: <https://proceedings.mlr.press/v168/jung22a.html>.

- 17 D. Jung, K. Y. Ng, E. Frisk, and M. Krysander. Combining Model-Based Diagnosis and Data-Driven Anomaly Classifiers for Fault Isolation. *Control Engineering Practice*, 80:146–156, 2018. doi:10.1016/j.conengprac.2018.08.013.
- 18 D. Jung and T. Westny. Uncertainty-Aware Fault Diagnosis of Unknown Faults using Ensemble-Based NODE Residuals. *Mechanical Systems and Signal Processing*, 242:113599, 2026. doi:10.1016/j.ymssp.2025.113599.
- 19 C. Keliris, M. M. Polycarpou, and T. Parisini. A Distributed Fault Detection Filtering Approach for a Class of Interconnected Continuous-Time Nonlinear Systems. *IEEE Transactions on Automatic Control*, 58(8):2032–2047, 2013. doi:10.1109/TAC.2013.2253231.
- 20 H. Khorasgani and G. Biswas. Structural Fault Detection and Isolation in Hybrid Systems. *IEEE Transactions on Automation Science and Engineering*, 15(4):1585–1599, 2017. doi:10.1109/TASE.2017.2749447.
- 21 H. Khorasgani, A. Farahat, K. Ristovski, C. Gupta, and G. Biswas. A Framework for Unifying Model-Based and Data-Driven Fault Diagnosis. In *Proceedings of the Annual Conference of the PHM Society*, volume 10, September 2018. doi:10.36001/phmconf.2018.v10i1.530.
- 22 M. Krysander, J. Åslund, and M. Nyberg. An Efficient Algorithm for Finding Minimal Overconstrained Subsystems for Model-Based Diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 38(1):197–206, 2008. doi:10.1109/TSMCA.2007.909555.
- 23 E. Kurudzija, K. Dresia, V. Bareiß, G. Waxenegger-Wilfing, and J. Deeken. Transient System-Level Simulation of the LOX/LNG Expander-Bleed Rocket Engine LUMEN. SSRN. <https://ssrn.com/abstract=6806858>, 2026.
- 24 K. Lee, J. Cha, S. Ko, S.-Y. Park, and E. Jung. Fault Detection and Diagnosis Algorithms for an Open-Cycle Liquid Propellant Rocket Engine using the Kalman Filter and Fault Factor Methods. *Acta Astronautica*, 150:15–27, 2018. doi:10.1016/j.actaastro.2018.03.001.
- 25 M. A. Martin, H. H. Nguyen, W. D. Greene, and D. C. Seymout. Transient Mathematical Modeling for Liquid Rocket Engine Systems: Methods, Capabilities, and Experience. In *5th International Symposium on Liquid Space Propulsion*, Chattanooga, Tennessee, October 2003.
- 26 A. Mohammadi, M. Krysander, and D. Jung. Analysis of Grey-Box Neural Network-Based Residuals for Consistency-Based Fault Diagnosis. *IFAC-PapersOnLine*, 55(6):1–6, 2022. doi:10.1016/j.ifacol.2022.07.097.
- 27 A. Mohammadi, M. Krysander, and D. Jung. Consistency-Based Diagnosis Using Data-Driven Residuals and Limited Training Data. *Control Engineering Practice*, 159:106283, 2025. doi:10.1016/j.conengprac.2025.106283.
- 28 R. Murata, J. Marzat, H. Piet-Lahanier, S. Boujnah, and P. Belleoud. Residual Selection for Observer-Based Fault Detection and Isolation in a Multi-Engine Propulsion Cluster. In *Proceedings of the European Conference of the PHM Society*, volume 8, page 9, June 2024. doi:10.36001/phme.2024.v8i1.4056.
- 29 R. Murata, J. Marzat, H. Piet-Lahanier, S. Boujnah, and P. Belleoud. Fault Detection and Isolation Based on Structural Analysis: Application to a Multi-Engine Propulsion Cluster. *Sensors*, 25(4), 2025. doi:10.3390/s25041054.
- 30 J. W. Owen. Shuttle Propulsion System Major Events and the Final 22 Flights. In *6th JANNAF Liquid Propulsion Subcommittee Meeting, NASA Marshall Space Flight Center*, Huntsville, AL, USA, 2011.
- 31 S.-Y. Park and J. Ahn. Deep Neural Network Approach for Fault Detection and Diagnosis During Startup Transient of Liquid-Propellant Rocket Engine. *Acta Astronautica*, 177:714–730, 2020. doi:10.1016/j.actaastro.2020.08.019.
- 32 I. Pill, D. Jung, E. Kurudzija, A. Sztyber-Betley, M. Syfert, K. Dresia, G. Waxenegger-Wilfing, and J. de Kleer. The DX Competition 2025 and Its Benchmarks. *Open Access Series in Informatics (OASIs)*, 136:14:1–14:19, 2025. 36th International Conference on Principles of Diagnosis and Resilient Systems (DX 2025). doi:10.4230/OASIs.DX.2025.14.

- 33 S. Priotto and S. Le Gondiec. AI-Driven Health Monitoring Systems for Enhanced Propulsion Reliability. In *Space Propulsion Conference*, Bari, Italy, 2026. 3AF.
- 34 R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987. doi:10.1016/0004-3702(87)90062-2.
- 35 B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001. doi:10.1162/089976601750264965.
- 36 P. Simontacchia, S. Sagnier, E. Edeline, R. Blasi, A. Espinosa, J.-N. Caruana, S. Boujnah, and P. Altenhöfer. PROMETHEUS: Precursor of New Low-Cost Rocket Engine Family. In *Proceedings of the International Astronautical Congress, IAC*, 2022.
- 37 K. Tidriri, N. Chatti, S. Verron, and T. Tiplica. Bridging Data-Driven and Model-Based Approaches for Process Fault Diagnosis and Health Monitoring: A Review of Researches and Future Challenges. *Annual Reviews in Control*, 42:63–81, 2016. doi:10.1016/j.arcontrol.2016.09.008.
- 38 T. Traudt, W. Armbruster, C. Groll, R. H. Dos Santos Hahn, K. Dresia, M. Börner, S. Klein, D. Suslov, E. Kurudzija, J. Haemisch, M. A. Müller, J. C. Deeken, J. Hardi, and S. Schlechtriem. LUMEN, the Test Bed for Rocket Engine Components: Results of the Acceptance Tests and Overview on the Engine Test Preparation. In *9th Edition of the Space Propulsion Conference*, Glasgow, Scotland, 2024.
- 39 C. Wang, Y. Zhang, Z. Zhao, X. Chen, and J. Hu. Dynamic Model-Assisted Transferable Network for Liquid Rocket Engine Fault Diagnosis Using Limited Fault Samples. *Reliability Engineering & System Safety*, 243:109837, 2024. doi:10.1016/j.ress.2023.109837.
- 40 T. Wang, L. Ding, and H. Yu. Research and Development of Fault Diagnosis Methods for Liquid Rocket Engines. *Aerospace*, 9(9):481, 2022.
- 41 J. Wu. Liquid-Propellant Rocket Engines Health-Monitoring — a Survey. *Acta Astronautica*, 56(3):347–356, 2005. doi:10.1016/j.actaastro.2004.05.070.
- 42 X. Zhu, Y. Cheng, J. Wu, R. Hu, and X. Cui. Steady-State Process Fault Detection for Liquid Rocket Engines Based on Convolutional Auto-Encoder and One-Class Support Vector Machine. *IEEE Access*, 8:3144–3158, 2020. doi:10.1109/ACCESS.2019.2961742.

On the Practical Utility of Diagnoses

Ingo Pill   

Institute of Software Engineering and Artificial Intelligence, TU Graz, Austria

Johan de Kleer   

c-infinity, Mountain View, CA, USA

Abstract

Diagnoses address a universal need for explanations that arises whenever things go wrong or when we observe some unexpected behavior. The ubiquitous need for such explanations has led to an abundance of algorithmic concepts and computation variants for diagnostic processes. Comparisons have mainly focused on highlighting the virtues of individual algorithms or the required computational efforts. In this paper, we focus on evaluating the *results* of a diagnosis algorithm *in the context of an inspection and repair process*. We completely ignore the algorithm's computation concept and whether the observations would justifiably support multiple diagnoses. In particular, we focus on inspecting and discussing the *practical utility* of diagnoses as a measure of the unnecessary efforts one has to spend when fully repairing a system. We start with assessing the utility of a single diagnosis and then progress to evaluating ambiguity groups, i.e., sets of diagnoses. We propose corresponding metrics that investigate worst case and average performance, contrast them to the utility metric used in the DX Competition 2010, and discuss the metrics' results for several scenarios.

2012 ACM Subject Classification Computing methodologies → Causal reasoning and diagnostics

Keywords and phrases Model-based Diagnosis, Diagnosis, Algorithms

Digital Object Identifier 10.4230/OASICS.DX.2026.9

Acknowledgements We would like to thank the anonymous reviewers for their detailed reviews and very valuable comments.

1 Introduction

As we discussed in detail in [8], comparing and evaluating diagnosis algorithms is a much more complex task than one would expect at first glance. Natural evaluation metrics include correctness and computational efficiency. An intuitive but also essential dimension that has been under-represented in the literature is that of how well a diagnosis algorithm's results support a user in an inspection and repair process. In other words, how well they inform a user about what is actually wrong with the system so that they can effectively repair it.

In this paper, we focus exactly on this question. Instead of focusing on algorithmic traits and computational efficiency, our sole focus is on the practical effectiveness of the reported diagnoses. Intuitively, we are interested in how much a diagnosis would improve the effort we have to spend inspecting and repairing the system. Intuitively, in the worst case, we need to investigate all of the components of a system. In the best case, if we were lucky or well informed by a diagnosis algorithm, we would know *exactly* which components are faulty, so that we need to investigate and repair *only* those faulty components and can avoid working on others. In practice, we will end up somewhere between these two extreme cases, and we would like to be as close to the ideal case as possible to minimize the efforts and costs of the inspection and repair process.

The utility of a diagnosis as described in [3] was a first step in this direction and was used to evaluate the diagnosis algorithms for the DX Competition 2010. However, as we will show for several examples, this previously advocated metric is imprecise and makes estimates that do not measure what we really want. In this paper, we will thus delve deeply into the



© Ingo Pill and Johan de Kleer;

licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 9; pp. 9:1–9:20

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

question of how to capture *diagnostic utility* in the first place, and we will propose *precise* metrics for the *average* and *worst case utility* (The best case is obvious). Those metrics will cover scenarios where we want to evaluate a single diagnosis as well as sets of diagnoses. We will furthermore propose extended variants that allow us to capture inspection/repair costs that vary between individual system parts. This is relevant in practice, as in real-world applications the costs of the resulting repair process can be much more important than diagnostic precision. In particular, we might require high precision only for some components that are expensive and accept an over-approximation for cheaply replaceable components – which we can deduce also from where we place our sensors and make our measurements. Inspecting and repairing a motor block is certainly much more costly than replacing a spark plug, for example.

Our paper is organized as follows. In the next section, we introduce our notation and offer basic definitions such as the previously published metric to capture a diagnosis’ utility. In Section 3, we briefly discuss diagnosis inspired system inspection and repair processes in order to provide the necessary background for evaluating some diagnoses’ utility. Our presentation of corresponding utility metrics is divided into Sections 4 and 5 focusing on single- and multi-diagnosis scenarios. We evaluate the average and worst cases and use several examples to illustrate our arguments. In Section 6, we consider how to incorporate more complex considerations of inspection and repair costs into a utility metric. We conclude with a final discussion and an outlook on future work.

2 Notation and Basic Definitions

In this section, we introduce our notation and provide basic definitions. We also describe the basic utility metric that was proposed in [3] and used for the DX Competition 2010.

Please note that our definitions occasionally deviate from what the reader might be familiar with from the literature and papers like [2, 10, 9]. The purpose is to develop evaluation metrics that are agnostic to the algorithmic concept used to derive diagnoses. A simple example is that we do not require diagnoses to be subset-minimal. For Reiter’s diagnostic theory [10] it made sense since every super-set of a diagnosis solves the diagnosis problem as well in the context of that theory and we want to avoid unnecessary computations. In our case, however, we do not require subset-minimality in a diagnosis nor do we make any assumption how a diagnosis was computed.

Without loss of generality (see Corollary 8), we assume that a system SD consists of components $x_i \in X$, and that we have some observations OBS about the behavior of SD .¹ Following the literature [2, 10], from SD , X and OBS we construct a diagnosis problem (SD, X, OBS) . A diagnosis Δ for a diagnosis problem points us to a subset $X_C \subseteq X$ of faulty components that can explain OBS . A diagnosis assigns to each component $x_i \in X$ a mode $m \in M_i$, where M_i is the finite set of modes in which x_i can be.

The set M_i must contain the nominal mode m_{nom} and at least one fault mode. The latter can be the completely unconstrained general fault mode m_{gen} or a finite non-zero number of concrete fault modes m_k . The general fault mode implements a *Weak Fault Model (WFM)* [2, 10] that does not place restrictions on the faulty behavior of x_i . With concrete fault modes that implement a *Strong Fault Model (SFM)*, we can pose concrete constraints on the faulty behavior. Technically, there is no requirement to describe *exact* behavior (that

¹ As usual, we will also formally refer to a system model with SD and it will be clear from the context whether we refer to the system as an entity or to its model.

defines the component's output for every input combination), but only some constraint. In practice, we tend to describe exact faulty behavior, though, like when we describe a stuck-at-0 fault for a logic gate. The increased precision over a WFM approach comes at the cost of a larger search space. That is, the search space grows significantly as formalized in Def. 4. Another side effect is that we limit our diagnostic search to the implemented fault modes at the algorithmic level: If we do not model a fault, we will not be able to find it.

We consider nominal mode assignments in a diagnosis Δ to be optional, so that only those tuples (x_i, m_j) in Δ that assign a non-nominal mode $m_j \in M_i \setminus \{m_{nom}\}$ to some component x_i need to be present. All components $x_i \in X$ for which Δ does not contain an explicit assignment are defaulted to the nominal mode.

► **Definition 1** (diagnosis problem, diagnosis, subset-minimal diagnosis). *Given a system SD , a set OBS of observations about SD 's behavior, a set of components X , and for each $x_i \in X$ a set of modes M_i that x_i can take and where M_i contains x_i 's nominal mode m_{nom} and at least one fault mode. A diagnosis Δ for a diagnosis problem (SD, X, OBS) is defined as a set of mode assignments (x_j, m_k) , such that the component x_j is assigned a mode $m \in M_j$ and such that there are no two assignments (x_j, m_k) and (x_l, m_m) that refer to the same component ($x_j = x_l$). Any $x_j \in X$ for which Δ contains no explicit mode assignment $(x_j, *)$ is assigned m_{nom} by default. A diagnosis Δ is subset-minimal if and only if there is no diagnosis Δ' s.t. $\Delta' \subset \Delta$. In other words, there must be no other diagnosis Δ' whose set of explicit mode assignments is a subset of the explicit mode assignments defined by Δ .*

We would like to point out to the reader that, unlike Reiter's theory [10], our definition does not hint at a concept for computing diagnoses. We also do not assume guaranteed features about a diagnosis, such as that OBS is consistent with SD under the fault assumptions suggested by Δ (as suggested in the same theory). Similarly, we do not require that a diagnosis has a probability greater than zero, as we did in [8]. We simply consider the case where diagnoses are provided as the results of *some* diagnosis algorithm (DA).

► **Definition 2** (health state variables). *Please note that for simplicity reasons, for a diagnosis problem (SD, X, OBS) as of Def. 1, we define a set of health state variables H that contains for each $x_i \in X$ a corresponding h_i that can take as a value any mode $m_j \in M_i$.*

► **Corollary 3.** *Without loss of generality, we can also define a diagnosis problem as (SD, H, OBS) . Similarly, a diagnosis and a subset-minimal diagnosis can be defined according to Def. 1 but where we refer to H instead of X .*

► **Definition 4** (diagnosis space). *The diagnosis space Θ of a diagnosis problem as of Def. 1 is defined by the product of the components' mode sets M_i*

$$\Theta = \prod_{x_i \in X} M_i$$

If and only if we implement a weak fault model (so that there is the nominal and one unconstrained fault mode m_{gen}) or if there is a single constrained fault mode, we will also write $\Theta = 2^X$. Similarly, we will also write $\Theta = M^X$ in case we have a single universal mode set M that is applicable to all components (or health state variables).

► **Definition 5** (set of injected or actual faults, ideal diagnosis result). *Ideally, a diagnosis Δ derived by some algorithm is identical to the set $\Delta_Q \in \Theta$ of actual faults in SD . The set Δ_Q might relate to faults that were injected artificially or that occurred naturally.*

The ideal result is user-centric in that we do not refer to complete sets of justifiable diagnoses based on a theory. The goal is to identify the actual faults. Thus, inherent limitations such as limited observability that would have complete algorithms such as RC-Tree [9] or GDE [2] compute multiple justified diagnoses are not taken into account.

► **Definition 6** (faulty component set, faulty component candidate set). *For the set of injected faults Δ_Q , we define the faulty component set $X_F \subseteq X$ as the set of components x_i that Δ_Q assigns a non-nominal mode. Similarly, for a diagnosis Δ we define the faulty component candidate set $X_C \subseteq X$ as the set of components that Δ assigns a non-nominal mode.*

► **Remark 7.** Please note that with our definition of Δ_Q , we implicitly assume that the set of health state variables H in a diagnosis problem was modeled precisely enough, so that an algorithm *can* find the faults. In particular, we assume that $\Delta_Q \in \Theta$.

► **Remark 8.** We assume that a system consists of components. Natural components that have been used in the literature would be statements or functions in a program, gates or blocks in a digital system, physical components in a mechanical system, or any other type of entity or group of entities in SD . Technically, there is no restriction to any scheme for defining what a component is. We can literally have any part of SD as $x_i \in X$. A component could thus consist of two clauses in a conjunctive normal form description that belong to two individual physical components (like gates in a digital circuit). At the same time, we must take into account that, in general, there might be parts of SD (like a wire between two components) that do not belong to any component $x_i \in X$ so that they are not within the scope of any $h_i \in H$.

An important assumption that usually remains hidden in formalizations of a component-based system is that any specific part (like a sentence) of SD must not belong to more than one component $x_i \in X$. Not making this assumption would break some effects regarding parsimony/subset-minimality that are actively exploited in computation concepts [10].

In our basic definitions, we do not consider probabilities. This is in contrast to [8], where we assumed a diagnosis to report a probability for each $\Delta' \in \Theta$. As visible in the remainder of this paper, we will still discuss how to adopt probabilities if they are available.

In 2010, the organizers used the following metric to evaluate algorithms in the DX Competition. This metric has the property that $m_{utl_DXC} = 1$ when $\Delta = \Delta_Q$ (perfect diagnosis), and decreases as false positives or false negatives increase. In Section 4, compared to our metrics, we can see that it suffers from problems and reports unexpected results for certain scenarios. Sometimes, the reported value is even worse than the worst case value that we can precisely compute with our own metrics.

$$m_{utl_DXC}(\Delta, \Delta_Q, X) = 1 - \frac{|X_C \setminus X_F| * (|X_C| + 1)}{|X| * (|X_C \setminus X_F| + 1)} - \frac{|X_F \setminus X_C| * (|X \setminus X_C| + 1)}{|X| * (|X_F \setminus X_C| + 1)} \quad (1)$$

3 Diagnosis-Inspired System Inspection and Repair

When trying to assess the practical utility of diagnoses, we have to think about how we can exploit a diagnosis during inspection and repair [4]. In order to investigate this, let us first focus on a single diagnosis. This refers to cases where there is either only one diagnosis for a given diagnosis problem, where we isolated a *lead diagnosis* as a working hypothesis, where the diagnosis algorithm reported only one diagnosis, or where we compare multiple diagnoses.

In principle, a diagnosis $\Delta \in \Theta$ defines a set $X_C \subseteq X$ of components that we assume to be faulty (see Def. 6). Depending on the diagnosis space Θ , a diagnosis provides us for each $x_i \in X_C$ with either information that there is *something* wrong with x_i (m_{gen} for a WFM

computation), or detailed information about *what* is wrong with it for an SFM approach². The latter simplifies inspection & repair, but has other side effects, as we discussed in the previous section. However, let us ignore this difference for now and work with X_C . That is, since we can compute it in either case. Please note that we will elaborate on this abstraction in Sec. 6 where we discuss a detailed consideration of costs and the impact of Θ .

Simply put, we should inspect & repair all $x_i \in X_C$. That is, since a diagnosis suggests faulty components and we, ideally, have $X_C = X_F$. Assuming that no fault probabilities are available, we would do so in arbitrary order. For the inspection and repair steps, we would make the implicit/hidden assumption that we have *perfect bug understanding* and similarly that we perform a *perfect repair* [8]. That is, we will always (and immediately) detect a fault when inspecting a component, and we can and will perfectly repair it. An inspection and repair procedure will thus always succeed if the right components are worked on.

In addition, we assume that we can recognize a fully repaired system. Once all faulty components $x_i \in X_F$ have been inspected and repaired, we can thus recognize that the system is fault-free.³ A welcome effect of this assumption is that if a diagnosis Δ' is a proper super-set of the actual faults Δ_Q (s.t. $\Delta_Q \subset \Delta'$), we can stop the inspection and repair process once we inspected all x_i in X_F rather than having to inspect and repair all $x_i \in X_C$ ⁴. In this case, Δ suggests for some components that they are faulty while they are actually healthy. We coin these components $x_i \in X_C \setminus X_F$ *false negatives*.

► **Definition 9** (false negatives). *For a given diagnosis Δ as defined in Def. 1, the actual faults Δ_Q as of Def. 5, as well as X_C and X_F as of Def. 6, the set of false negatives can be derived as $X_{FN} = X_C \setminus X_F$.*

We coin the defective components that Δ fails to report *false positives*. If we suffer from such undetected defect components (s.t. $X_F \setminus X_C \neq \emptyset$), we must also investigate $x_i \in X \setminus X_C$.

► **Definition 10** (false positives). *For a given diagnosis Δ as of Def. 1, the actual faults Δ_Q as of Def. 5, as well as X_C and X_F as of Def. 6, the set of false positives can be derived as $X_{FP} = X_F \setminus X_C$.*

In many cases, a DA returns more than one diagnosis. For instance, when an algorithm is not precise enough, or when the diagnosis problem has multiple plausible solutions. The latter happens often in practice when the (limited) observations do not allow us to distinguish competing hypotheses. Indeed, when implementing a well-founded and complete analytical concept [10, 2, 9], an algorithm is forced to return multiple diagnoses in these ambiguous cases. Another source for such ambiguity is Θ . When using a weak fault model, it is well-known [10] that when we add a component to a diagnosis, the resulting Δ' can still explain the observed behavior. To illustrate the ambiguity, consider the example of two inverters in series [8]. Assume that we can observe the in- and output of the circuit and that the output is wrong. Since we have no observation about the connecting wire, there is no way to compute which inverter is faulty. Even both inverters could be faulty – in opposition to Occam's razor.

The main question now is how we would actually use a set Θ' of diagnoses. In some cases, each diagnosis might be assigned a probability, so that we rank them from the most probable diagnosis to the least likely one.⁵ If we have no probabilities, we could, e.g., take

² Such as when a logic gate suffers from a stuck-at-0 fault.

³ The conclusion is valid only for inspected scenarios and faults that *manifested* on the outputs in *OBS*.

⁴ Please note that, as described in [8], this assumption implicitly requires that healthy and inspected/repaired components must not break (again) between inspections of the individual components.

⁵ In arbitrary local order whenever there are multiple diagnoses with the same probability

into account the cardinality and/or the frequency with which the components in Δ appear in Θ' to order diagnoses. We could then walk through the diagnoses in this order, testing every new component along the way and discarding from Θ' those diagnoses that we can rule out whenever we learn whether some component is healthy or not.

There is also the option to not rank the diagnoses, but to combine them in a single set. Formally, we derive the union X'_C of the candidate spaces X_C of the individual diagnoses $\Delta_i \in \Theta'$. We would then inspect the components x_i in this *multi-diagnosis fault candidate set*. The order in which we inspect X'_C could be completely random, or, as suggested earlier, we could rank the components according to the frequency with which they appear in diagnoses.

► **Definition 11** (multi-diagnosis fault candidate set). *Given a set $\Theta' \subseteq \Theta$ of diagnoses, and computing X_{C_i} for each $\Delta_i \in \Theta'$, the multi-diagnosis fault candidate set X'_C is defined as*

$$X'_C \subseteq X = \bigcup_{\Delta_i \in \Theta'} X_{C_i}$$

Obviously, there are an abundance of processes that we could adopt to take advantage of Θ' in an inspection and repair process. As we discussed in [8], all have their individual advantages and disadvantages. In practice, we would tailor the choice to the specific application domain and the structure of the diagnosis problem.

For our evaluation, we need to draw conclusions that generalize. We thus focus on X'_C and a random inspection order. This allows us to avoid any influence of the structure of a specific diagnosis problem and a specific repair process on the average utility. In particular, it is easy to see that the ordering of diagnoses or components⁶ would depend on the structural aspects of the specific diagnosis problem which is (a) difficult to predict/estimate and (b) biased toward the sample. Inspecting/repairing the $x_i \in X'_C$ in random order allows us to circumnavigate those problems in a structured way and without introducing a new bias.

It is important to note that sometimes we would specifically look for a biased metric. In a *batch repair* process as suggested by Stern and Kalech in [13] and [14], we, e.g., leverage synergies during repair. Those arise when the costs of repairing a group of components are lower than the sum of the individual repair costs. Their concept of repairing components in batches differs significantly from our step-wise repair assumption and would certainly require an individual metric. As a final note, we would like to specifically note that we assume that we have to work with the available diagnoses only, i.e., since we are interpreting those diagnoses. Additional active testing for diagnosis/hypothesis discrimination [1] that is potentially supported by additional signal probing [5] (so that we arrive at solving a sequential diagnosis [11] problem), as well as approaches that integrate diagnosis and repair like [16] are thus not directly applicable and not in our scope.⁷

4 The Practical Utility of a Single Diagnosis

Consider the utility of a single diagnosis and let us first focus on a WFM such that $\Theta = 2^X$. Ideally, the DA would return the actual faults Δ_Q as the diagnosis result (see Def. 5). Intuitively, Δ_Q also represents the *minimal set of components that we need to inspect and repair* to arrive at a fully repaired system. Consequently, we cannot circumvent spending inspection and repair efforts for $x_i \in \Delta_Q$ and we in turn do not spend any *avoidable* efforts

⁶ As noted before, we could inspect the $x_i \in X'_C$ according to the number of times they appear in $\Delta_j \in \Theta'$.

⁷ The underlying concepts would allow us to compute educated orders for repairing x_i from X_C or X'_C .

if $\Delta = \Delta_Q$. Intuitively, our metric should thus return the maximum utility value for this scenario. In our work, we target a metric $m_{util}(\Delta, \Delta_Q, X)$ that provides a value in the interval $[0, 1]$, so that we should have $m_{util}(\Delta, \Delta_Q, X) = 1$ if and only if $\Delta = \Delta_Q$. The minimum value of 0 indicates that we spend the maximum of those efforts that we could theoretically avoid. Practically speaking, it corresponds to the worst case in which we have to investigate *all* components $x_i \in X$, and especially all $x_i \in X \setminus X_F$ since we cannot avoid the $x_i \in X_F$. Having to inspect all components seems a bit extreme, and one might thus be tempted to make the misled assumption that we would never see such a case in practice. But, as we will show later, we can suffer from a diagnosis with utility 0 even for small examples. For the range between the two extremes, we aim for a linear decrease in $m_{util}(\Delta, \Delta_Q, X)$. This entails that the utility decreases by $\frac{1}{|X \setminus X_F|}$ for any additional (unnecessarily) investigated component since we can compute the precise upper bound on the number of avoidable components as X_F 's complement $X_A = |X \setminus X_F|$.

► **Definition 12** (necessary/unavoidable and unnecessary/avoidable inspection space). *The entire inspection space is defined via the set of components X (or alternatively H). The necessary or unavoidable inspection space $X_U \subseteq X$ is the set of faulty components X_F as of Def. 6. The unnecessary or avoidable inspection space X_A is defined as its complement $X_A = X \setminus X_U = X \setminus X_F$.*

► **Corollary 13.** *If the avoidable inspection space X_A is empty, then the worst and average utility of a diagnosis is 1 by default.*

Proof. (sketch) If we have $X_A = \emptyset$, then $X_F = X$ due to $X_A = X \setminus X_F$. In this case, we have to inspect and repair all $x_i \in X$, no matter which x_i a diagnosis suggests to be faulty. ◀

► **Assumption 14.** *Due to Corollary 13, we have that the utility of a diagnosis is 1 if $X_A = \emptyset$. For all the metric formulae that we provide, we thus assume $X_A \neq \emptyset$. The reason is twofold. First, we don't need any computation in this case. And second, we would otherwise need to define the special case of $X_A = \emptyset$ for most equations which we will explain for Eq. 2.*

Intuitively speaking, the envisaged metrics capture the fraction of the *avoidable inspection space* X_A that we can *actually* avoid when we take into account a diagnosis Δ .⁸ So, let us formally define the metrics for the *worst case* and *average* utility of a diagnosis, starting with a consideration of the worst case. Recall from Sec. 3 how we would exploit some diagnosis Δ . We first investigate all $x_i \in X_C$. Only if this does not result in a working system (s.t. $\Delta_Q \not\subseteq \Delta$), would we investigate components in the complement of X_C . *False positives* (see Def. 10), i.e., components $x_i \in X_F$ that were reported to work OK so that $x_i \notin X_C$, and *false negatives* (see Def. 9), i.e., components x_i that are OK ($x_i \notin X_F$) but are reported as defective by Δ (s.t. $x_i \in X_C$) should have an impact on the metric. In fact, in their union they define the actual additional efforts that we need to spend on our inspection/repair process. For instance, if and only if there are no false positives s.t. $\Delta_Q \setminus \Delta = \emptyset$, we can define a diagnosis Δ 's worst-case utility via the following equation:

$$m_{util_{wc}}(\Delta, \Delta_Q, X) = 1 - \frac{|X_C \setminus X_F|}{|X_A|} \quad (2)$$

Taking into account Eq. 2, we can easily see the motivation for the Assumption 14. That is, $X_A = \emptyset$ would lead to a division by 0 so that we would need to define a specific case. Let us now interpret the formula and recall that we basically deduct from the maximum value

⁸ By the above obvious conjecture, we cannot avoid looking at the $x_i \in \Delta_Q$ since they are faulty.

1 the fraction of the avoidable inspection space X_A that we actually have to inspect. In a worst case scenario, we would first inspect the false negatives, which means that we have to investigate *all* components $x_i \in X_C$. Since there are no false positives, we stop the inspection after the last $x_i \in X_C$ since we recognize that the system is fault free. If we also suffer from false positives, we also need to check components not in X_C since then $X_F \setminus X_C \neq \emptyset$. In a worst case scenario, we inspect the false positives last which leads us to the following formula:

$$m_{utl_{wc}}(\Delta, \Delta_Q, X) = 1 - \frac{|X_C \setminus X_F|}{|X_A|} - \frac{|X \setminus (X_F \cup X_C)|}{|X_A|} \quad (3)$$

The interested reader will have deduced that we actually have $m_{utl_{wc}} = 0$ then. Formally, this follows from the observation that $|X_C \setminus X_F| + |X \setminus (X_F \cup X_C)| = |X \setminus X_C| = |X_A|$. Nevertheless, we can observe that the two fractions each capture the individual fractions of the unnecessary efforts associated with false negatives and positives. The first captures the efforts associated with false negatives, and the second those connected to false positives.

Now that we have a clear picture about the worst case utility, let us switch our focus to investigating the *average case utility*. Like for the worst case evaluation, let us first assume that there are no false positives. The main difference versus the worst case scenario is that we do not inspect the false negatives first. Rather, we inspect $x_i \in X_C$ in a completely random order – as we discussed in Section 3. When X_C now contains y -many faulty components, and z -many false negatives,⁹ the question is how many components from X_C would we inspect on average until all $x_i \in X_F$ have been covered.

Order statistics and in particular the scenario of a hypergeometric distribution¹⁰ provide us with an answer. Consider a simple example where we draw balls from a bowl. Assume that there are n balls (and that we do not replace balls). k -many balls are red and the others green for $0 \leq k \leq n$. Using the following equation, we can estimate the average number of draws that we need to recover all the red balls. The intuition behind the formula is that we estimate the position of the last red ball in a random ordering.

$$E[\text{draws/position}] = \frac{k * (n + 1)}{k + 1} \quad (4)$$

Obviously, this estimate is equal to the number of draws that we need on average to retrieve all the red balls. Recall that, in principle, we are interested not in the total number of picks, but rather in the unnecessary ones that we could have avoided to make. In other words, the picks that produced a green ball instead of a red one. To derive this number, we simply subtract k from the estimate, since we need these k picks to recover all the red balls.

$$E[\text{avoidable draws}] = E[\text{draws/position}] - k = \frac{k * (n + 1)}{k + 1} - k = \frac{k * (n - k)}{k + 1} \quad (5)$$

When transferring Eq. 5 to our diagnosis-oriented inspection/repair process, k is the number of faulty components s.t. $k = y = |X_F|$, n the number of components in X_C s.t. $n = y + z = |X_C|$, and $z = n - k = |X_C| - |X_F| = |X_C \setminus X_F|$.¹¹ For our metric, we also need to divide the number of avoidable inspections/repairs by X_A . This ensures that we subtract the *fraction* of X_A that we could not actually avoid. Please note that we use the subscript X_C to E to indicate that we take into account only false negatives at this time.

$$E[\text{avoidable inspections}]_{X_C} = \frac{|X_F| * |X_C \setminus X_F|}{(|X_F| + 1)}$$

⁹ Since we assume there are no false positives, $y = |X_F|$ and we have $z = |X_C \setminus X_F|$, and $y + z = |X_C|$.

¹⁰ <https://mathworld.wolfram.com/HypergeometricDistribution.html>

¹¹ Note that $X_F \subseteq X_C$ due to the absence of false positives.

$$m_{utl_{avg}}(\Delta, \Delta_Q, X) = 1 - \frac{E[\text{avoidable inspections}]_{X_C}}{|X_A|} = 1 - \frac{|X_F| * |X_C \setminus X_F|}{|X_A| * (|X_F| + 1)} \quad (6)$$

Now that we know how to handle false negatives, let us work towards characterizing the effects false positives have on the utility. An important observation that we need to make is that even false negatives must be handled differently *whenever* there are false positives. That is, whenever there are false positives, we cannot receive feedback that the system is repaired when investigating components $x_i \in X_C$. Consequently, we must inspect the entire X_C and cannot avoid the inspection of *any* false negative. With $\frac{|X_C \setminus X_F|}{|X_A|}$, the impact of false negatives on $m_{utl_{avg}}(\Delta, \Delta_Q, X)$ is thus the same as for the worst case utility (see Eq. 3). Since there are false positives, we must also inspect at least some $x_i \in X \setminus X_C$. To calculate an estimate, how many $x_i \in X \setminus X_C$ we need to inspect and repair on average, we again adapt Eq. 4. This time, the set of components “to draw from” is of size $n = |X \setminus X_C|$, and $k = |X_F \setminus X_C|$ is the number of false positives that we need to find:

$$E[\text{avoidable inspections}]_{X \setminus X_C} = \frac{|X_F \setminus X_C| * (|X \setminus X_C| - |X_F \setminus X_C|)}{|X_F \setminus X_C| + 1}$$

$$m_{utl_{avg}}(\Delta, \Delta_Q, X) = 1 - \frac{|X_C \setminus X_F|}{|X_A|} - \frac{|X_F \setminus X_C| * (|X \setminus X_C| - |X_F \setminus X_C|)}{|X_A| * (|X_F \setminus X_C| + 1)} \quad (7)$$

So, let us evaluate our metrics in the context of a few examples, and let us compare the results to those for the 2010 DX Competition metric captured in Eq. 1 in Section 2.

1. **Example 1:** $|X| = 10, \Delta_Q = \Delta = \{x_1, x_5, x_6, x_8\} : |X_C| = 4, |X_F| = 4, |X_C \setminus X_F| = 0, |X_F \setminus X_C| = 0, |X_A| = |X \setminus X_F| = 6$. In this scenario, we have neither false negatives nor false positives. Intuitively, the utility should be 1 since we make no unnecessary inspections and repairs.

$$m_{utl_DXC} = 1 - \frac{0*5}{10*1} - \frac{0*7}{10*1} = 1$$

$$m_{utl_{wc}} = 1 - \frac{0}{6} = 1, m_{utl_{avg}} = 1 - \frac{4*0}{6*5} = 1$$

2. **Example 2:** $|X| = 10, \Delta_Q = \{x_1, x_5, x_6, x_8\}, \Delta = \{x_1, x_5, x_6, x_8, x_9\} : |X_C| = 5, |X_F| = 4, |X_C \setminus X_F| = 1, |X_F \setminus X_C| = 0, |X_A| = |X \setminus X_F| = 6$. In this scenario, there is one false negative but no false positive. Intuitively, the utility should thus be lower than 1 since we (potentially) need to investigate one additional component. For the worst case estimate, we can be more precise in that it should be lower by $1/|X_A| = 1/6$. The average utility should be higher than that. For this example, we can easily observe that the metric used for the DX competition did not catch the average case very well. In particular, the reported value is even *lower* than our precise worst case utility.

$$m_{utl_DXC} = 1 - \frac{1*6}{10*2} - \frac{0*6}{10*1} = 0.7$$

$$m_{utl_{wc}} = 1 - \frac{1}{6} = 0.8\dot{3}, m_{utl_{avg}} = 1 - \frac{4*1}{6*5} = 0.8\dot{6}$$

3. **Example 3:** $|X| = 10, \Delta_Q = \{x_1, x_5, x_6, x_8\}, \Delta = \{x_1, x_2, x_5, x_6, x_8, x_9\} : |X_C| = 6, |X_F| = 4, |X_C \setminus X_F| = 2, |X_F \setminus X_C| = 0, |X_A| = |X \setminus X_F| = 6$. In this scenario, there are two false negatives but no false positives. Compared to Example 2, the values for all metrics should be lower since we have two false negatives instead of one. We can again observe that the DXC metric reports a lower average utility than our precise worst case utility estimate.

$$m_{utl_DXC} = 1 - \frac{2*7}{10*3} - \frac{0*5}{10*1} = 0.5\dot{3}$$

$$m_{utl_{wc}} = 1 - \frac{2}{6} = 0.6\dot{6}, m_{utl_{avg}} = 1 - \frac{4*2}{6*5} = 0.7\dot{3}$$

4. **Example 4:** $|X| = 10, \Delta_Q = \{x_1, x_5\}, \Delta = \{x_1, x_5, x_6, x_8\} : |X_C| = 4, |X_F| = 2, |X_C \setminus X_F| = 2, |X_F \setminus X_C| = 0, |X_A| = |X \setminus X_F| = 8$. Here we have two false negatives but no false positives. We thus need to investigate two additional components – as in Example 3. The score is still supposed to be different, since the amount of injected faults is different. That is, since this has an impact on the fraction of the components that we could avoid investigating. The DXC average utility metric suffers from the same issue as for the previous two examples and reports a utility that is below the worst case utility.

$$m_{utl_DXC} = 1 - \frac{2*5}{10*3} - \frac{0*7}{10*1} = 0.6$$

$$m_{utl_{wc}} = 1 - \frac{2}{8} = 0.75, m_{utl_{avg}} = 1 - \frac{2*2}{8*3} = 0.8\dot{3}$$

5. **Example 5:** $|X| = 10, \Delta_Q = \{x_1, x_5, x_6, x_8\}, \Delta = \{x_1, x_5, x_8, x_9\} : |X_C| = 4, |X_F| = 4, |X_C \setminus X_F| = 1, |X_F \setminus X_C| = 1, |X_A| = |X \setminus X_F| = 6$. In this scenario, there is one false negative (x_9) and one false positive (x_6). The worst case utility should be 0 since there is a false positive. For the computation of $m_{utl_{wc}}$, note that $|X_F \cup X_C| = 5$. In contrast to the previous three examples, the DXC average utility metric reports a value that is quite close to our average utility $m_{utl_{avg}}$.

$$m_{utl_DXC} = 1 - \frac{1*5}{10*2} - \frac{1*7}{10*2} = 0.4$$

$$m_{utl_{wc}} = 1 - \frac{1}{6} - \frac{5}{6} = 0, m_{utl_{avg}} = 1 - \frac{1}{6} - \frac{1*(6-1)}{6*2} = 0.41\dot{6}$$

6. **Example 6:** $|X| = 10, \Delta_Q = \{x_1, x_5, x_6, x_8\}, \Delta = \{x_1, x_5, x_8\} : |X_C| = 3, |X_F| = 4, |X_C \setminus X_F| = 0, |X_F \setminus X_C| = 1, |X_A| = |X \setminus X_F| = 6$. In this scenario, there is no false negative but one false positive. Due to the false positive, the worst case utility should be 0. For the computation of $m_{utl_{wc}}$ note that we have $X_C \cup X_F = X_F$ for this case. Intuitively, the average utility should be 0.5, i.e., since there is no false negative, and we will find the single false positive in the middle of inspecting $X \setminus X_C$ on average. Our average case metric fulfills this intuition, while the DXC one reports a larger value.

$$m_{utl_DXC} = 1 - \frac{0*4}{10*1} - \frac{1*8}{10*2} = 0.6$$

$$m_{utl_{wc}} = 1 - \frac{0}{6} - \frac{6}{6} = 0, m_{utl_{avg}} = 1 - \frac{0}{6} - \frac{1*(7-1)}{6*2} = 0.5$$

7. **Example 7:** $|X| = 10, \Delta_Q = \{x_1, x_4, x_5, x_6, x_7, x_8, x_9, x_A\}, \Delta = \{\{x_1, x_2, x_3, x_A\} : |X_C| = 4, |X_F| = 8, |X_C \setminus X_F| = 2, |X_F \setminus X_C| = 6, |X_A| = |X \setminus X_F| = 2$. This scenario is a quite interesting corner case. We have 8 faulty components, but only two (x_1, x_A) are part of the reported diagnosis Δ . Δ contains 2 false negatives (x_2, x_3) which are actually the only two components that are not faulty. Consequently, we have $X_F \cup X_C = X$ for our worst case evaluation. Despite the complicated scenario, the worst case utility should be 0 since we have false positives. In this specific scenario, also the avg. utility should be 0 since we indeed have to investigate *all* components. Our metric matches these expectations, while the DXC metric is slightly off.

$$m_{utl_DXC} = 1 - \frac{2*5}{10*3} - \frac{6*7}{10*7} = 0.0\dot{6}$$

$$m_{utl_{wc}} = 1 - \frac{2}{2} - \frac{0}{2} = 0, m_{utl_{avg}} = 1 - \frac{2}{2} - \frac{4*(6-6)}{2*7} = 0$$

As we can see in the results, our metrics report values that match the intuitions and expectations we have about a metric that captures the utility of a diagnosis. In some cases also the DXC metric is close, but in others it is clearly off. Sometimes it reports a value that is even below that of the worst case utility, a situation that must not arise for a metric that is supposed to describe the average case.

There are a number of hidden assumptions that our formulae have been based on, such as that there is a health state variable for each component¹². In practice, this might not be the case, so more general equations can be defined by considering H (the set of health state variables) instead of the component set X . For example, Eq. 7 could be reformulated to

$$m_{util_{avg}}(\Delta, \Delta_Q, H) = 1 - \frac{|H_C \setminus H_F|}{|H_A|} - \frac{|H_F \setminus H_C| * (|H \setminus H_C| - |H_F \setminus H_C|)}{|H_A| * (|H_F \setminus H_C| + 1)} \quad (8)$$

We furthermore assumed that the cost of inspecting any component is unity. In reality, the costs for inspecting and repairing components differ between individual components. Consequently, we might want to consider a cost function, which we discuss in Section 6.

A most prominent assumption was that we focused first on a weak fault model with the diagnosis space $\Theta = 2^X$. The imminent question now is whether the move to a strong fault model with a larger Θ would require us to derive new formulae, and our answer is *yes and no*. That is, the answer depends on how we define the costs of inspecting a component. If we approximate the costs with unity no matter the fault model, and if we assume perfect bug understanding, we can indeed use all formulae proposed so far also for the SFM case. Only if we do not assume unity, we need to use different formulae. Since this assumes a cost function, let us continue the discussion in Section 6.

5 Considering Ambiguity Groups

In many if not most practical cases, a DA will return more than one diagnosis – as we analyzed in Sec. 3. Like we argued in the same section, there are multiple options to exploit the set Θ' of reported diagnoses. An important and inherent challenge is that the structure of Θ' depends on the structure of the system, the observations, and Δ_Q . For illustration purposes, assume that we pick a certain element for inspection. Depending on whether the component is actually faulty, a certain set of diagnoses in Θ' will become invalid, so that a working set of hypotheses or X'_C would need to be re-computed. Depending on the metrics used to select a component, we might need to update also the ranking of the individual diagnoses and components that we follow in the inspection and repair process. For a precise average utility metric, we would need to estimate the effects of this process in a meaningful way which is a quite complex challenge. For this and other reasons raised in Sec. 3, we chose to focus on the multi-diagnosis fault candidate set X'_C as of Def. 11.

Like for single diagnoses, we initially assume $\Theta = 2^X$ and start our discussion with the worst case utility. It is easy to see that we can easily transfer the worst case utility equations for the single diagnosis case to cover multiple diagnoses by considering X'_C instead of X_C . If we assume that there is no false positive (see Def. 10), i.e., $X_F \setminus X'_C = \emptyset$, we arrive at:

$$m_{util_{wc}}(\Theta', \Delta_Q, X) = 1 - \frac{|X'_C \setminus X_F|}{|X_A|} \quad (9)$$

Like for a single diagnosis, we basically deduct from the maximum utility value the fraction of the avoidable inspection space X_A that we actually have to inspect. If there are false positives, we need to also look at components in X'_C 's complement since $X_F \setminus X'_C \neq \emptyset$. In the worst case, we inspect the false positives last, so that we arrive at:

$$m_{util_{wc}}(\Theta', \Delta_Q, X) = 1 - \frac{|X'_C \setminus X_F|}{|X_A|} - \frac{|X \setminus (X_F \cup X'_C)|}{|X_A|} \quad (10)$$

¹²In fact, we argued primarily with X rather than H .

Please note that Cor. 13 transfers also to X'_C and Assumption 14 is thus valid also when considering ambiguity groups. The average utility situation is more complex. Since being utterly precise is not an option due to the issues discussed in Sec. 3, we investigated estimations and present two variants. For $m_{utl_{avg_est}}(\Theta', \Delta_Q, X)$, we compute X'_C and transfer the formulae we proposed for the single diagnosis case. If there are no false positives ($X'_C \setminus X_F = \emptyset$), we thus have:

$$m_{utl_{avg_est}}(\Theta', \Delta_Q, X) = 1 - \frac{|X_F| * |X'_C \setminus X_F|}{|X_A| * (|X_F| + 1)} \quad (11)$$

If there are also false positives $X'_C \setminus X_F \neq \emptyset$, we would use

$$m_{utl_{avg_est}}(\Theta', \Delta_Q, X) = 1 - \frac{|X'_C \setminus X_F|}{|X_A|} - \frac{|X_F \setminus X'_C| * (|X \setminus X'_C| - |X_F \setminus X'_C|)}{|X_A| * (|X_F \setminus X'_C| + 1)} \quad (12)$$

For both equations, we compute X'_C and randomly process components from X'_C and thus connect to order statistics. A welcome side effect of these formulae is that the obtained values compare nicely to those for the single diagnosis case. That is, when $|\Theta'| = 1$ and $X'_C = X_C$ the results are indeed the same as if we had used the single diagnosis formula.

We investigated also an alternative $m_{utl_{avg_est'}}(\Theta', \Delta_Q, \rho, X)$ that does not focus on computing X'_C . To compute it, we, in principle, average over the average utility of all $\Delta_i \in \Theta'$. If we have probabilities for the individual diagnoses, we can also incorporate weights. To that end, we consider a function $\rho(\Delta)$ that gives us the probability of some $\Delta \in \Theta'$. With $1 = \sum_{\Delta_i \in \Theta'} \rho(\Delta_i)$ and $\rho(\Delta_i) = \frac{1}{|\Theta'|}$ in case the DA does not report probabilities, we have

$$m_{utl_{avg_est'}}(\Theta', \Delta_Q, \rho, X) = \sum_{\Delta_i \in \Theta'} \rho(\Delta_i) * m_{utl_{avg}}(\Delta_i, \Delta_Q, X) \quad (13)$$

Let us now test the metrics for some examples that we derived from those used in Sec. 4.

- **Example 1':** $|X| = 10, \Delta_Q = \{x_1, x_5, x_6, x_8\}, \Theta' = \{\{x_1, x_5, x_6\}, \{x_5, x_8, x_1\}, \{x_6, x_8\}\} : |X_F| = 4, |X'_C| = 4, |X'_C \setminus X_F| = 0, |X_F \setminus X'_C| = 0, |X_A| = |X \setminus X_F| = 6$. In this variant of Example 1, the DA returns 3 diagnoses (Θ'), and none of these diagnoses represents Δ_Q . However, their union X'_C is equal to Δ_Q , so we have neither false negatives nor false positives. Intuitively, the worst case and average utility for our X'_C -oriented approach should therefore be 1 since the DA identified *all* faulty components and *only* faulty components. The fact that no diagnosis is equal to Δ_Q , is actually irrelevant to the repair and inspection process. In particular, even if we did not follow the process described in Section 3 but consider the diagnoses in Θ' in any imaginable way and order, would we inspect and repair faulty components only.

$$m_{utl_{wc}} = 1 - \frac{0}{6} = 1, m_{utl_{avg_est}} = 1 - \frac{4*0}{6*5} = 1$$

For $m_{utl_{avg_est_2}}$, we would compute the average over 3 bad single diagnosis utilities where Δ_i refers to the i -th diagnosis in Θ . For Δ_1 , $|X_C| = 3, |X_C \setminus X_F| = 0, |X_F \setminus X_C| = 1$. For Δ_2 , we have $|X_C| = 3, |X_C \setminus X_F| = 0, |X_F \setminus X_C| = 1$, and for Δ_3 , $|X_C| = 2, |X_C \setminus X_F| = 0, |X_F \setminus X_C| = 2$. So we do not have false negatives, but false positives for each Δ_i .

$$m_{utl_{avg}}(\Delta_1) = 1 - \frac{0}{6} - \frac{1*(7-1)}{6*2} = 0.5,$$

$$m_{utl_{avg}}(\Delta_2) = 1 - \frac{0}{6} - \frac{1*(7-1)}{6*2} = 0.5,$$

$$m_{utl_{avg}}(\Delta_3) = 1 - \frac{0}{6} - \frac{2*(8-2)}{6*3} = 0.3$$

$$m_{utl_{avg_est'}}(\Theta') = \frac{1}{3} * (0.5 + 0.5 + 0.3) = 0.4$$

From this single example, we can easily see that $m_{utl_{avg_est'}}$ has serious problems in assessing the utility of Θ' . In fact, it is a good indicator of how good individual diagnoses are on average, but it does not tell us the utility of Θ' .

- **Example 2’:** $|X| = 10, \Delta_Q = \{x_1, x_5, x_6, x_8\}, \Theta' = \{\{x_1, x_5, x_6\}, \{x_5, x_8, x_1, x_9\}, \{x_6, x_8\}\} : |X'_C| = 5, |X_F| = 4, |X'_C \setminus X_F| = 1, |X_F \setminus X'_C| = 0, |X_A| = |X \setminus X_F| = 6$. In this scenario, there is one false negative but no false positive for X'_C . Intuitively, the utility should thus be slightly lower than 1 since we (potentially) need to investigate one additional component. For the worst case estimate, we can be more precise in that it should be lower by $1/X_A = 1/6$.

$$m_{utl_{wc}} = 1 - \frac{1}{6} = 0.8\dot{3}, m_{utl_{avg_est}} = 1 - \frac{4*1}{6*5} = 0.8\dot{6}$$

Now, let us look at how $m_{utl_{avg_est'}}$ works for this example. For Δ_1 , $|X_C| = 3, |X_C \setminus X_F| = 0, |X_F \setminus X_C| = 1$. For Δ_2 , we have $|X_C| = 4, |X_C \setminus X_F| = 1, |X_F \setminus X_C| = 1$, and for Δ_3 , $|X_C| = 2, |X_C \setminus X_F| = 0, |X_F \setminus X_C| = 2$. So we have false positives for each Δ_i , and for Δ_2 there is also one false negative.

$$m_{utl_{avg}}(\Delta_1) = 1 - \frac{0}{6} - \frac{1*(7-1)}{6*2} = 0.5$$

$$m_{utl_{avg}}(\Delta_2) = 1 - \frac{1}{6} - \frac{1*(6-1)}{6*2} = 0.41\dot{6}$$

$$m_{utl_{avg}}(\Delta_3) = 1 - \frac{0}{6} - \frac{2*(8-2)}{6*3} = 0.\dot{3}$$

$$m_{utl_{avg_est'}}(\Theta') = \frac{1}{3} * (0.5 + 0.41\dot{6} + 0.\dot{3}) = 0.41\dot{6}$$

Compared to the previous example, the false positive for Δ_2 further decreases $m_{utl_{avg_est'}}$ – as expected and as should be. We again observe that this metric is more indicative of the distance between Δ_i and Δ_Q in terms of efforts, rather than describing the utility of Θ' . That is, for this example, there is only one false negative for which we have to waste our efforts when following our inspection and repair approach for Θ' . Since $m_{utl_{avg_est'}}$ does not implement the desired intuitions, we skip it for the remaining examples.

- **Example 3’:** $|X| = 10, \Delta_Q = \{x_1, x_5, x_6, x_8\}, \Theta' = \{\{x_1, x_5, x_6\}, \{x_5, x_8, x_1, x_9\}, \{x_5, x_6, x_8, x_2\}\} : |X'_C| = 6, |X_F| = 4, |X'_C \setminus X_F| = 2, |X_F \setminus X'_C| = 0, |X_A| = |X \setminus X_F| = 6$. This is a multi-diagnosis variant of Example 3 with the same values reported by the respective metrics. That is, X'_C is equal to X_C derived for Example 3.

$$m_{utl_{wc}} = 1 - \frac{2}{6} = 0.\dot{6}, m_{utl_{avg_est}} = 1 - \frac{4*2}{6*5} = 0.7\dot{3}$$

- **Example 4’:** $|X| = 10, \Delta_Q = \{x_1, x_5\}, \Theta' = \{\{x_1, x_5, x_6\}, \{x_5, x_8, x_1\}, \{x_5, x_6, x_8\}\} : |X'_C| = 4, |X_F| = 2, |X'_C \setminus X_F| = 2, |X_F \setminus X'_C| = 0, |X_A| = |X \setminus X_F| = 8$. In this scenario, X'_C contains two false negatives, but no false positives. Like for Example 3’, we need to investigate two additional components. The score is still supposed to be different than for Example 3’, since the amount of injected faults differs between the examples.

$$m_{utl_{wc}} = 1 - \frac{2}{8} = 0.75, m_{utl_{avg_est}} = 1 - \frac{2*2}{8*3} = 0.8\dot{3}$$

- **Example 5’:** $|X| = 10, \Delta_Q = \{x_1, x_5, x_6, x_8\}, \Theta' = \{\{x_1, x_5, x_9\}, \{x_5, x_8\}, \{x_1, x_8\}\} : |X'_C| = 4, |X_F| = 4, |X'_C \setminus X_F| = 1, |X_F \setminus X'_C| = 1, |X_A| = |X \setminus X_F| = 6$. In this scenario, there are one false negative (x_9) and one false positive (x_6) in X'_C . The worst case utility should be 0 since there is a false positive. For the computation of $m_{utl_{wc}}$, note that $|X_F \cup X'_C| = 5$.

$$m_{utl_{wc}} = 1 - \frac{1}{6} - \frac{5}{6} = 0, m_{utl_{avg_est}} = 1 - \frac{1}{6} - \frac{1*(6-1)}{6*2} = 0.41\dot{6}$$

- **Example 6’:** $|X| = 10, \Delta_Q = \{x_1, x_5, x_6, x_8\}, \Theta' = \{\{x_1, x_5\}, \{x_5, x_8\}, \{x_1, x_8\}\} : |X'_C| = 3, |X_F| = 4, |X'_C \setminus X_F| = 0, |X_F \setminus X'_C| = 1, |X_A| = |X \setminus X_F| = 6$. In this scenario, there is no false negative but one false positive. Due to the false positive, the worst case utility should be 0. For the computation of $m_{utl_{wc}}$ note that we have $X_C \cup X_F = X_F$ for this case. Intuitively, the average utility should be 0.5 as there is no false negative, and, on average, we will find the remaining false positive (x_6) in the middle of inspecting $X \setminus X_C$. $m_{utl_{wc}} = 1 - \frac{0}{6} - \frac{6}{6} = 0, m_{utl_{avg_est}} = 1 - \frac{0}{6} - \frac{1*(7-1)}{6*2} = 0.5$

- **Example 7’:** $|X| = 10, \Delta_Q = \{x_1, x_4, x_5, x_6, x_7, x_8, x_9, x_A\}, \Theta' = \{\{x_A, x_2\}, \{x_3\}, \{x_1, x_A\}\} : |X'_C| = 4, |X_F| = 8, |X'_C \setminus X_F| = 2, |X_F \setminus X'_C| = 6, |X_A| = |X \setminus X_F| = 2$. This scenario is a quite interesting corner case. We have 8 faulty components, but only two

are part of X'_C . X'_C contains 2 false negatives that are actually the only two components (x_2, x_3) that are not faulty. Consequently, we have $X_F \cup X'_C = X$ for our worst case evaluation. Despite the complicated scenario, the worst case utility should be 0 since we have false positives. In this specific scenario, also the avg. utility should be 0 since we indeed have to investigate all components due to the specific structure of the diagnosis and the diagnosis problem.

$$m_{util_{wc}} = 1 - \frac{2}{2} - \frac{0}{2} = 0, m_{util_{avg_est}} = 1 - \frac{2}{2} - \frac{6*(6-6)}{2*7} = 0$$

While $m_{util_{avg_est'}}$ might seem intuitive at first glance, we can easily observe that its results are counterintuitive. The particular problem is that we might average over potentially bad single diagnosis utilities which causes us to miss that the information contained in Θ' in its entirety can lead to a *perfect* inspection and repair process. We still chose to report it in order to highlight this observation. $m_{util_{avg_est'}}$ on the other hand, leverages the synergies between the individual $\Delta_i \in \Theta'$ via its computation of X'_C . As we saw, even for non-ideal Δ_i s such that none of them is equal to Δ_Q , we can achieve a perfect inspection and repair process with a utility score of 1. That is, as long as we have no false negatives¹³ in any $\Delta_i \in \Theta'$, nor false positives¹⁴ in X'_C , the utility of Θ' can and *should* indeed be 1.

► **Proposition 15.** *Given a diagnosis problem (SD, X, OBS) with diagnosis space $\Theta = 2^X$, the injected faults Δ_Q , and a multi-diagnosis solution $\Theta' \subseteq \Theta$ provided by some diagnosis algorithm. The worst case and average case utility for Θ' can have an ideal value of 1 even if Θ' does not contain Δ_Q as a diagnosis.*

Proof. The proof follows directly from the empirical evidence for Example 1', its discussion, and the definition of our inspection and repair process in Section 3. ◀

Like for the single diagnosis scenario, there is the question of whether an SFM diagnosis space Θ would require us to use different formulae to describe the utility of Θ' . And again we must answer this question taking the repair costs into account. However, the situation is slightly different since the individual $\Delta_i \in \Theta'$ might suggest different fault modes for the same component. So, if we approximate the costs of inspecting and repairing a component in X'_C with unity, no matter the fault modes suggested by the individual Δ_i , and if we assume perfect bug understanding, we can indeed use all formulae proposed so far also for the SFM case. Only if we do not assume unity, do we need to use different formulae and replace X'_C with a more complex data structure. Due to the inherent connection to cost functions, let us continue the discussion in the next section.

6 Taking Individual Costs Into Account

A prominent assumption made for the previously proposed metric is that inspection and repair costs should be equal for all components. This is certainly not the case in practice, as we can observe when comparing the inspection and repair costs for a motor block with those for a spark plug. Consequently, for real-world applications, we might be interested in metrics that take into account variations in those costs.

To define the corresponding variants of our metric, assume that there is a cost function $c(x_i) : X \rightarrow \mathbb{R}^+$ that reports for each component x_i (or health state variable if preferred) a positive real value indicating the costs of investigating and repairing x_i .

¹³ Healthy components that are contained in one $\Delta_i \in \Theta'$ (see Def. 9).

¹⁴ Faulty components that are part of no $\Delta_i \in \Theta'$ so that they are not part of X'_C (see Def. 10).

► **Definition 16.** Given a set of components X , a cost function $c(x \in X)$ is defined as function $c() : X \rightarrow \mathbb{R}^+$ that maps each component $x_i \in X$ to a positive real value $r \in \mathbb{R}^+$ that describes the costs of inspecting and repairing x_i .

Please note that splitting the costs for inspecting and repairing a component is indeed possible. Sometimes one would even be able to provide more exact results as we mentioned in Sec. 3. A simple example is when we have only two components and inspect the first one. Depending on the result, we will indeed need to repair only one of the components. This split adds some additional complexity to the formulae and argumentation line¹⁵, though, so that we chose to consider unified costs in this manuscript. For considering costs instead of the *number* of false positives and false negatives, we make the following three updates:

1. We do not consider the cardinality of a diagnosis or the number of components in certain sets but the sum of the associated costs.
2. The *unavoidable costs* of the *unavoidable efforts* are given by the sum of the costs of inspecting and repairing all $x_i \in X_F$.
3. The *avoidable costs* of the *avoidable efforts* are given by the sum of the costs of inspecting and repairing all $x_i \in X \setminus X_F$.

For a simpler notation, let us define a cost function $C(X')$ that computes for a given component subset $X' \subseteq X$ the costs for inspecting and repairing all $x_i \in X'$:

► **Definition 17.** For a given subset X' of X , and a cost function $c()$ following Def. 16, let us define a cost function $C(X') = \sum_{x_i \in X'} c(x_i)$.

Replacing the size $|X'|$ of a component subset with costs $C(X')$ allows us to generate variants of our previous equations that work with costs. In some of the earlier equations, we see a +1 for some terms. It had the meaning of one additional component. When considering costs, we thus replace this +1 with the adding average costs over all elements. The motivation is simple. In the original formula, we add 1 to the number of components in a specific subset, and now we add the average costs for one component to the sum of the costs.¹⁶ Implementing this translation concept, we end up with the equations listed in the next subsection.

In Subsection 6.2, we will continue our discussion of whether using a strong fault model with multiple fault modes would require us to update our metrics.

6.1 The WFM Case

Based on the proposed transfer concept, we can derive the following metrics. Let us start with the single diagnosis metrics from Section 4. From Equations 2,3 (no false positives/with false positives) we can derive the worst case utility as (remember that $X_A = X \setminus X_F$):

$$mC_{utl_{wc}}(\Delta, \Delta_Q, X, C()) = 1 - \frac{C(X_C \setminus X_F)}{C(X_A)} \quad (14)$$

$$mC_{utl_{wc}}(\Delta, \Delta_Q, X, C()) = 1 - \frac{C(X_C \setminus X_F)}{C(X_A)} - \frac{C(X \setminus (X_F \cup X_C))}{C(X_A)} \quad (15)$$

¹⁵ In order to be precise, we would then need to estimate costs when we know by deduction (and not inspection) that a component is actually wrong.

¹⁶ In order to take the exact term into account more precisely, we could also use, e.g., the average over the complement of the subset considered in the term.

For the average utility, we get from Equations 6,7 (no false positives/with false positives) the following formulae, where $c_{avg} = \frac{1}{|X|} \sum_{x_i \in X} c(x_i)$ represents the average costs of inspecting and repairing some component.

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C()) = 1 - \frac{C(X_F) * C(X_C \setminus X_F)}{C(X_A) * (C(X_F) + c_{avg})} \quad (16)$$

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C()) = 1 - \frac{C(X_C \setminus X_F)}{C(X_A)} - \frac{C(X_F \setminus X_C) * (C(X \setminus X_C) - C(X_F \setminus X_C))}{C(X_A) * (C(X_F \setminus X_C) + c_{avg})} \quad (17)$$

Note that we can refer also to H instead of X like in Eq. 7. We only need to overload $C()$ to describe the costs for the SD parts in the scope of H rather than for some component:

$$mC_{utl_{avg}}(\Delta, \Delta_Q, H, C()) = 1 - \frac{C(H_C \setminus H_F)}{C(H_A)} - \frac{C(H_F \setminus H_C) * (C(H \setminus H_C) - C(H_F \setminus H_C))}{C(H_A) * (C(H_F \setminus H_C) + c_{avg})} \quad (18)$$

Proceeding to the multi-diagnosis utility metrics from Section 5, the formulae for the worst case utility can be deduced from Eqs. 9,10 (no false positives/with false positives):

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C()) = 1 - \frac{C(X'_C \setminus X_F)}{C(X_A)} \quad (19)$$

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C()) = 1 - \frac{C(X'_C \setminus X_F)}{C(X_A)} - \frac{C(X \setminus (X_F \cup X'_C))}{C(X_A)} \quad (20)$$

The cost-oriented variants for Eqs. 11, 12 (no false positives/with false positives) that encode our first average utility estimation variant are defined as follows:

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C()) = 1 - \frac{C(X_F) * C(X'_C \setminus X_F)}{C(X_A) * (C(X_F) + c_{avg})} \quad (21)$$

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C()) = 1 - \frac{C(X'_C \setminus X_F)}{C(X_A)} - \frac{C(X_F \setminus X'_C) * (C(X \setminus X'_C) - C(X_F \setminus X'_C))}{C(X_A) * (C(X_F \setminus X'_C) + c_{avg})} \quad (22)$$

Since the experimental evaluation in Sec. 5 showed that $m_{utl_{avg_est}}$ does not explain the utility of some Θ' very well, we do not provide a cost-related version. Now, let us revisit Examples 1-3,6-7 in order to illustrate the cost-based metrics. Note that cost functions are described as lists such that the costs are ordered according to the component ID.

- **Example 1/1':** $|X| = 10, \Delta_Q = X_F = \{x_1, x_5, x_6, x_8\}, |X_F| = 4, |X_A| = |X \setminus X_F| = 6$, two cost functions $C_1() = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$, $C_2() = [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]$ s.t. $c_{avg_1} = 1$ and $c_{avg_2} = 3$.

$$\Delta = X_C = \Delta_Q : |X_C| = 4, |X_C \setminus X_F| = 0, |X_F \setminus X_C| = 0$$

$$mC_{utl_{wc}}(\Delta, \Delta_Q, X, C_1()) = 1 - \frac{0}{1+1+1+1+1+1} = 1 \text{ (via Eq. 14)}$$

$$mC_{utl_{wc}}(\Delta, \Delta_Q, X, C_2()) = 1 - \frac{0}{2+3+4+2+4+5} = 1 \text{ (via Eq. 14)}$$

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C_1()) = 1 - \frac{(1+1+1+1)*0}{(1+1+1+1+1+1)*((1+1+1+1)+1)} = 1 \text{ (via Eq. 16)}$$

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C_2()) = 1 - \frac{(1+5+1+3)*0}{(2+3+4+2+4+5)*((1+5+1+3)+3)} = 1 \text{ (via Eq. 16)}$$

$$\Theta' = \{\{x_1, x_5, x_6\}, \{x_5, x_8, x_1\}, \{x_6, x_8\}\} : X'_C = \Delta_Q, |X'_C| = 4, |X'_C \setminus X_F| = 0, |X_F \setminus X'_C| = 0.$$

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C_1()) = 1 - \frac{0}{1+1+1+1+1+1} = 1 \text{ (via Eq. 19)}$$

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C_2()) = 1 - \frac{0}{2+3+4+2+4+5} = 1 \text{ (via Eq. 19)}$$

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C_1()) = 1 - \frac{(1+1+1+1)*0}{(1+1+1+1+1+1)*((1+1+1+1)+1)} = 1 \text{ (via Eq. 21)}$$

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C_2()) = 1 - \frac{(1+5+1+3)*0}{(2+3+4+2+4+5)*((1+5+1+3)+3)} = 1 \text{ (via Eq. 21)}$$

Intuitively, since we have no false positives nor false negatives, the utility should be 1 regardless of the utilized cost function, whether we consider the worst or average case, and whether we have one or more diagnoses.

- **Example 2/2':** $|X| = 10, \Delta_Q = X_F = \{x_1, x_5, x_6, x_8\}, |X_F| = 4, |X_A| = |X \setminus X_F| = 6, C_1() = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], C_2() = [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]$ s.t $c_{avg_1} = 1$ and $c_{avg_2} = 3$.

$$\Delta = \{x_1, x_5, x_6, x_8, x_9\} : |X_C| = 5, |X_C \setminus X_F| = 1, |X_F \setminus X_C| = 0$$

$$mC_{utl_{wc}}(\Delta, \Delta_Q, X, C_1()) = 1 - \frac{1}{6} = 0.8\dot{3} \text{ (via Eq. 14)}$$

$$mC_{utl_{wc}}(\Delta, \Delta_Q, X, C_2()) = 1 - \frac{4}{2+3+4+2+4+5} = 0.8 \text{ (via Eq. 14)}$$

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C_1()) = 1 - \frac{(1+1+1+1)*1}{(1+1+1+1+1+1)*((1+1+1+1)+1)} = 0.8\dot{6} \text{ (via Eq. 16)}$$

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C_2()) = 1 - \frac{(1+5+1+3)*4}{(2+3+4+2+4+5)*((1+5+1+3)+3)} = 0.846 \text{ (via Eq. 16)}$$

$$\Theta' = \{\{x_1, x_5, x_6\}, \{x_5, x_8, x_1, x_9\}, \{x_6, x_8\}\} : X'_C = \{x_1, x_5, x_6, x_8, x_9\}, |X'_C| = 5, |X'_C \setminus X_F| = 1, |X_F \setminus X'_C| = 0.$$

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C_1()) = 1 - \frac{1}{6} = 0.8\dot{3} \text{ (via Eq. 19)}$$

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C_2()) = 1 - \frac{4}{2+3+4+2+4+5} = 0.8 \text{ (via Eq. 19)}$$

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C_1()) = 1 - \frac{(1+1+1+1)*1}{(1+1+1+1+1+1)*((1+1+1+1)+1)} = 0.8\dot{6} \text{ (via Eq. 21)}$$

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C_2()) = 1 - \frac{(1+5+1+3)*4}{(2+3+4+2+4+5)*((1+5+1+3)+3)} = 0.846 \text{ (via Eq. 21)}$$

As we can observe, if we assume unity for the costs we get the same results as when not using a cost-related metric. It is easy to see that this observation is true also if the cost function has a uniform distribution with a higher cost value like 2. Between C1 and C2, the expected penalty for C2 results from the fact that the false negative x_9 has higher (4) than average ($c_{avg_2} = 3$) costs.

- **Example 3/3':** $|X| = 10, \Delta_Q = X_F = \{x_1, x_5, x_6, x_8\}, |X_F| = 4, |X_A| = |X \setminus X_F| = 6, C_1() = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], C_2() = [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]$ s.t $c_{avg_1} = 1$ and $c_{avg_2} = 3$.

$$\Delta = X_C = \{x_1, x_5, x_6, x_8, x_3, x_9\} : |X_C| = 6, |X_C \setminus X_F| = 2, |X_F \setminus X_C| = 0$$

$$mC_{utl_{wc}}(\Delta, \Delta_Q, X, C_1()) = 1 - \frac{2}{6} = 0.\dot{6} \text{ (via Eq. 14)}$$

$$mC_{utl_{wc}}(\Delta, \Delta_Q, X, C_2()) = 1 - \frac{3+4}{2+3+4+2+4+5} = 0.65 \text{ (via Eq. 14)}$$

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C_1()) = 1 - \frac{(1+1+1+1)*2}{(1+1+1+1+1+1)*((1+1+1+1)+1)} = 0.7\dot{3} \text{ (via Eq. 16)}$$

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C_2()) = 1 - \frac{(1+5+1+3)*(3+4)}{(2+3+4+2+4+5)*((1+5+1+3)+3)} \approx 0.731 \text{ (via Eq. 16)}$$

$$\Theta' = \{\{x_1, x_5, x_6\}, \{x_5, x_8, x_1, x_9\}, \{x_5, x_6, x_8, x_3\}\} : X'_C = \{x_1, x_5, x_6, x_8, x_3, x_9\}, |X'_C| = 6, |X'_C \setminus X_F| = 2, |X_F \setminus X'_C| = 0.$$

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C_1()) = 1 - \frac{2}{6} = 0.\dot{6} \text{ (via Eq. 19)}$$

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C_2()) = 1 - \frac{7}{20} = 0.65 \text{ (via Eq. 19)}$$

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C_1()) = 1 - \frac{4*2}{6*(4+1)} = 0.7\dot{3} \text{ (via Eq. 21)}$$

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C_2()) = 1 - \frac{10*7}{20*(10+3)} \approx 0.731 \text{ (via Eq. 21)}$$

- **Example 6/6':** $|X| = 10, \Delta_Q = X_F = \{x_1, x_5, x_6, x_8\}, |X_F| = 4, |X_A| = |X \setminus X_F| = 6, C_1() = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], C_2() = [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]$ s.t $c_{avg_1} = 1$ and $c_{avg_2} = 3$.

$$\Delta = X_C = \{x_1, x_5, x_8\} : |X_C| = 3, |X_C \setminus X_F| = 0, |X_F \setminus X_C| = 1$$

$$mC_{utl_{wc}}(\Delta', \Delta_Q, X, C_1()) = 1 - \frac{0}{6} - \frac{6}{6} = 0 \text{ (via Eq. 15)}$$

$$mC_{utl_{wc}}(\Delta', \Delta_Q, X, C_2()) = 1 - \frac{0}{2+3+4+2+4+5} - \frac{20}{20} = 0 \text{ (via Eq. 15)}$$

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C_1()) = 1 - \frac{0}{6} - \frac{1*(7-1)}{6*(1+1)} = 0.5 \text{ (via Eq 17)}$$

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C_2()) = 1 - \frac{0}{2+3+4+2+4+5} - \frac{1*(21-1)}{20*(1+3)} = 0.75 \text{ (via Eq 17)}$$

$$\Theta' = \{\{x_1, x_5\}, \{x_5, x_8\}, \{x_1, x_8\}\} : X'_C = \{x_1, x_5, x_8\}, |X'_C| = 3, |X_F| = 4, |X'_C \setminus X_F| = 0, |X_F \setminus X'_C| = 1, |X_A| = |X \setminus X_F| = 6$$

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C_1()) = 1 - \frac{0}{6} - \frac{6}{6} = 0 \text{ (via Eq. 20)}$$

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C_2()) = 1 - \frac{0}{2+3+4+2+4+5} - \frac{20}{20} = 0 \text{ (via Eq. 20)}$$

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C_1()) = 1 - \frac{0}{6} - \frac{1*(7-1)}{6*(1+1)} \text{ (via Eq. 22)}$$

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C_2()) = 1 - \frac{0}{20} - \frac{1*(21-1)}{20*(1+3)} = 0.75 \text{ (via Eq. 22)}$$

Compared to the previous examples, we have to switch formulae since we have a false positive. We remind the reader that we have $X \setminus (X_F \cup X'_C) = X \setminus (X_F \cup X_C) = X_A$ for this example. Due to the false positive, the worst case metrics report a utility of 0, no matter the cost function.

- **Example 7/7':** $|X| = 10, \Delta_Q = X_F = \{x_1, x_4, x_5, x_6, x_7, x_8, x_9, x_A\}, |X_F| = 8, |X_A| = |X \setminus X_F| = 2, C_1() = [2, 2, 2, 2, 2, 2, 2, 2, 2, 2], C_2() = [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]$ s.t $c_{avg_1} = 2$ and $c_{avg_2} = 3$.

$$\Delta = \{x_1, x_2, x_3, x_A\} : |X_C| = 4, |X_C \setminus X_F| = 2, |X_F \setminus X_C| = 6$$

$$mC_{utl_{wc}}(\Delta', \Delta_Q, X, C_1()) = 1 - \frac{2+2}{2+2} - \frac{0}{2+2} = 0 \text{ (via Eq. 15)}$$

$$mC_{utl_{wc}}(\Delta', \Delta_Q, X, C_2()) = 1 - \frac{2+3}{2+3} - \frac{0}{2+3} = 0 \text{ (via Eq. 15)}$$

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C_1()) = 1 - \frac{4}{4} - \frac{(2+2+2+2+2)*(12-12)}{4*(12+2)} = 0 \text{ (via Eq 17)}$$

$$mC_{utl_{avg}}(\Delta, \Delta_Q, X, C_2()) = 1 - \frac{5}{5} - \frac{(4+5+1+2+3+4)*((4+5+1+2+3+4)-(4+5+1+2+3+4))}{(2+3)*((4+5+1+2+3+4)+3)} = 0$$

$$\Theta' = \{\{x_A, x_2\}, \{x_3\}, \{x_1, x_A\}\} : X'_C = \{x_1, x_2, x_3, x_A\}, |X'_C| = 4, |X'_C \setminus X_F| = 2, |X_F \setminus X'_C| = 6$$

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C_1()) = 1 - \frac{4}{4} - \frac{0}{4} = 0 \text{ (via Eq. 20)}$$

$$mC_{utl_{wc}}(\Theta', \Delta_Q, X, C_2()) = 1 - \frac{5}{5} - \frac{0}{5} = 0 \text{ (via Eq. 20)}$$

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C_1()) = 1 - \frac{4}{4} - \frac{4*0}{4*14} \text{ (via Eq. 22)}$$

$$mC_{utl_{avg_est}}(\Theta', \Delta_Q, X, C_2()) = 1 - \frac{5}{5} - \frac{19*0}{5*22} = 0 \text{ (via Eq. 22)}$$

As desired, the utility is 0 for this example, no matter the cost function and whether we consider the worst or average case.

6.2 Do Strong Fault Models Change the Game?

A question that we have been continuously revisiting in our discussion is that of whether the use of strong fault models should change the way we define our metrics. In the context of cost functions, our answer depends on the cost function's complexity. In particular, we can observe that, in contrast to a WFM situation, Θ' might contain different faulty mode assignments for one and the same component. So, $\Delta_1 \in \Theta'$ could suggest fault mode m_5 for component x_2 , while $\Delta_4 \in \Theta'$ suggests m_6 instead. To that end, we could derive an intuitive alternative to X'_C in the form of a set X_C^* that contains all the individual mode assignments from Θ' . However, this would make sense only if we have cost estimates that depend on the suggested and actual fault modes. Enabling the corresponding precise and structured

estimates would require us to differentiate between inspection and repair costs.¹⁷ Defining and analyzing such a complex cost function is a task that we have to leave for future work. As a coarser estimate, we suggest to keep using cost functions as they were defined for the WFM case. Our argument is simply that our assumptions of perfect fault-understanding and repair mean that we will be able to (a) understand the fault and (b) repair it with the estimated costs. Consequently, we then need to look at each component just once and can use X'_C (and our formulae) instead of X^*_C also for the SFM case.

7 Discussion and Conclusions

Motivated by the fact that the practical assessment of diagnoses has been an under-represented research area, we delved into the question of how to measure the utility of one or more diagnoses. As we outlined in Section 3, we considered the context of a general inspection and repair process without any interleaved active testing or diagnosis steps. We established the required notation and definitions as well as developed precise metrics for assessing the worst case and average impact of a diagnosis algorithm's results on the process. In order to support complex real-world scenarios, we developed metric variants that allow us to take into account individual inspection/repair cost-functions.

We illustrated the advantages of our metrics for a set of examples and contrasted their performance with that of a metric proposed for the 2010 DX Competition [3]. As we showed, the DX Competition metric sometimes reports values that are even below our worst case evaluations. In our evaluation, we found that our metrics are much more precise. The are also intuitive and adhere to expected traits.

We discussed several directions for future work. Of specific interest are complex cost functions that split inspection and repair costs, and thus could offer advantages in the context of SFM scenarios. In order to support a general assessment, we defined metrics that generalize well enough, but complex repair processes (like batch repair) and elaborate concepts for ranking components/diagnoses could profit from precisely tailored metrics that consider the specifics of these processes. Such an investigation will need to be accompanied by experiments that investigate whether these dedicated variants show real benefits in practice. Only the isolation of such benefits could justify the additional effort needed to, e.g., define and maintain complex cost functions.

An important aspect of future experiments will be to compare the diagnostic information returned by complex analytic diagnosis approaches such as GDE [2] or RC-Tree [9] with those of computationally less complex ones, such as spectrum-based fault localization [6] and its recent extensions [12]. Data-driven solutions [15, 7] will be of specific interest in these experiments. Assessing the results with our utility metric will allow us to depict a better picture of the trade-off between computational complexity in the diagnostic process and the resulting efficiency of the inspection and repair process.

References

- 1 R. Alur, C. Courcoubetis, and M. Yannakakis. Distinguishing tests for nondeterministic and probabilistic machines. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pages 363–372, 1995. doi:10.1145/225058.225161.

¹⁷ As a bonus, this would allow us to effectively consider special scenarios like when we can deduce that the sole remaining component is faulty.

- 2 J. de Kleer and B. C. Williams. Reasoning about Multiple Faults. In *5th National Conf. on Artificial Intelligence Volume 1: Science*, pages 132–139, 1986.
- 3 A. Feldman, T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, J. de Kleer, L. Kuhn, and A. van Gemund. Empirical evaluation of diagnostic algorithm performance using a generic framework. *International Journal of Prognostics and Health Management*, pages 1–28, 2010.
- 4 G. Friedrich, G. Gottlob, and W. Nejdl. Formalizing the repair process. In *10th European Conference on Artificial Intelligence*, pages 709–713, 1992.
- 5 A. Hou. A theory of measurement in diagnosis from first principles. *Artificial Intelligence*, 65(2):281–328, February 1994. doi:10.1016/0004-3702(94)90019-1.
- 6 J. Jones and M. Harrold. Empirical evaluation of the tarantula automatic fault-localization technique. In *20th IEEE/ACM International Conference on Automated Software Engineering, ASE 2005*, pages 273–282, November 2005. doi:10.1145/1101908.1101949.
- 7 A. Lundgren and D. Jung. Data-driven fault diagnosis analysis and open-set classification of time-series data. *Control Engineering Practice*, 121:105006, 2022.
- 8 I. Pill and J. de Kleer. Assessing Diagnosis Algorithms: Of Sampling, Baselines, Metrics and Oracles. In *36th Int. Conf. on Principles of Diagnosis and Resilient Systems (DX 2025)*, volume 136 of *Open Access Series in Informatics (OASICS)*, pages 5:1–5:19, 2025. doi:10.4230/OASICS.DX.2025.5.
- 9 I. Pill and T. Quaritsch. RC-Tree: A variant avoiding all the redundancy in Reiter’s minimal hitting set algorithm. In *IEEE Int. Symp. on Software Reliability Engineering Workshops (ISSREW)*, pages 78–84, 2015.
- 10 R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987. doi:10.1016/0004-3702(87)90062-2.
- 11 P. Rodler. Sequential model-based diagnosis by systematic search. *Artificial Intelligence*, 323:103988, 2023. doi:10.1016/j.artint.2023.103988.
- 12 J. Schleich and F. Wotawa. Combining Dynamic Slicing and Spectrum-Based Fault Localization - A First Experimental Evaluation. In *36th Int. Conf. on Principles of Diagnosis and Resilient Systems (DX)*, volume 136 of *Open Access Series in Informatics (OASICS)*, pages 3:1–3:18, 2025. doi:10.4230/OASICS.DX.2025.3.
- 13 R. Stern and M. Kalech. Repair planning with batch repair. In *25th Int. Workshop on Principles of Diagnosis*, 2014. URL: http://dx-2014.ist.tugraz.at/papers/DX14_Wed_AM_S1_paper4.pdf.
- 14 R. Stern, M. Kalech, and H. Shinitzky. Implementing troubleshooting with batch repair. *CEUR Workshop Proceedings*, 1507:113–118, 2015. 26th Int. Workshop on Principles of Diagnosis. URL: <https://ceur-ws.org/Vol-1507/dx15paper15.pdf>.
- 15 H. S. Steude, A. Diedrich, I. Pill, L. Moddemann, D. Vranješ, and O. Niggemann. Data-driven diagnosis for large cyber-physical-systems with minimal prior information. In *2026 IEEE Conference on Artificial Intelligence (CAI)*, pages 325–332, 2026.
- 16 S. Thiébaux, M. Cordier, O. Jehl, and J. Krivine. Supply restoration in power distribution systems: a case study in integrating model-based diagnosis and repair planning. In *12th Int. Conf. on Uncertainty in Artificial Intelligence*, pages 525–532, 1996.

Planning Domain Repair Using Preferred Plans and Background Knowledge

Thomas Eckstein ✉

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Gerald Steinbauer-Wagner ✉ 🏠 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Abstract

Planning domain repair assists developers with the tedious task of finding and fixing errors in planning domains. We present Domain Repair for PDDL (DrPDDL), a novel automated debugger supporting lifted STRIPS domains with typing. As input, it takes a faulty domain, problem instances with corresponding preferred plans – plans the user desires but which may be invalid in the current domain – and optional State Integrity Axioms (SIAs) that must hold across all reachable states. Unlike prior tools, DrPDDL can also repair domains when the preferred plans are already valid. The debugger uses a pipeline of consecutive algorithms. It first relaxes the domain to fix failing action preconditions and SIA violations, and subsequently constricts it as much as possible while ensuring preferred plan validity. DrPDDL can add or remove positive and negative effects, as well as positive preconditions. We evaluated our debugger by automatically inserting random errors into International Planning Competition domains, generating thousands of test instances. The results demonstrate that DrPDDL successfully detects and repairs a high percentage of these errors.

2012 ACM Subject Classification Computing methodologies → Artificial intelligence; Computing methodologies → Planning and scheduling

Keywords and phrases Planning, Planning Domain Definition Language (PDDL), Debugging, Preferred Plans, Background Knowledge

Digital Object Identifier 10.4230/OASICS.DX.2026.10

1 Introduction

AI Planning is a field of research dedicated to exploring ways to solve problems commonly occurring in robotics, intelligent agents, and logistics that involve finding a sequence of actions. The Planning Domain Definition Language (PDDL) [5] is a popular language for modeling such problems so that a planner can solve them. The appeal behind planning is that the user can model a domain in a declarative way by stating a set of fluents along with actions and their effects. Planning adds another abstraction layer that eliminates the need for deriving a concrete algorithm for a problem, so that the developer doesn't need to care whether to use a greedy or exhaustive search strategy, for example.

However, since many developers aren't very familiar with planning or declarative programming, they may make mistakes that are hard to find when creating planning domains. We think that the many peculiarities of planning hinder its acceptance and use by a broader community of developers.

Therefore, we want to help developers and students with crafting planning domains by designing a debugger for PDDL. This way, developers can rely on a tool that checks the correctness of a planning domain while also giving suggestions for fixes, relieving the frustration that arises when testing a domain that just doesn't work as intended. So far, existing debuggers for PDDL, such as those designed by [4, 7, 11], can only handle problem instances where no plan can be found at all. These debuggers assume that the intended behavior of the domain is that a given set of problem instances should be solvable. However,



© Thomas Eckstein and Gerald Steinbauer-Wagner;
licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 10; pp. 10:1–10:22

OpenAccess Series in Informatics

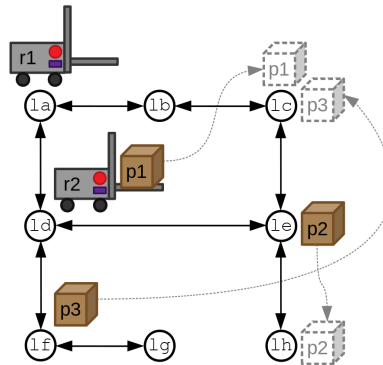


OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

10:2 Planning Domain Repair Using Preferred Plans and Background Knowledge

current PDDL debuggers are not able to detect and repair errors when the intended behavior is more complex. It often occurs that a solution is found, but the plan is incorrect because the domain doesn't behave as the developer intended.

For example, we consider a simple packet delivery domain. Multiple locations are connected via a graph, and one or more robots can move between them, as illustrated in Figure 1. Each robot can pick up and carry at most one packet at a time and drop it at another location. The goal is for all packets to be delivered to their destination.



■ **Figure 1** A possible configuration of the delivery robot domain with 8 locations, 2 robots, and 3 packets. Robot **rp** carries packet **p2**. The destinations of the packets stated in the goal are also shown.

In our example, the developer forgot the negative effect (`(not (isat ?r ?from))`) that removes the robot's old location in the `move` action. The developer tests the domain using a problem instance where the robot starts at location **1a** and needs to bring a packet from **1b** back to **1a**. The planner gives a solution that is obviously wrong since the robot must move back to **1a** after picking up the packet:

```
1 (move r1 1a 1b)
2 (pickup r1 p1 1b)
3 (drop r1 p1 1a)
```

However, the developer prefers to obtain the following plan:

```
1 (move r1 1a 1b)
2 (pickup r1 p1 1b)
3 (move r1 1b 1a)
4 (drop r1 p1 1a)
```

The domain developer also has the background knowledge that a robot can only be in one location at a time. He wishes to automatically verify whether this property holds for the domain, and otherwise, a fix should be suggested so that the property holds.

This example leads us to the idea of building a debugger for PDDL domains that fixes the domain by not only making problem instances solvable, but also considers additional knowledge of the developer about the domain. The developer should have more freedom in specifying the intended behavior of the domain, allowing a wider range of errors to be found.

In order to assist PDDL developers with writing correct planning domains, our goal is to design and implement a novel debugger for PDDL domains. Instead of an interactive tool like many debuggers found in IDEs, we want a tool that requires little user intervention and finds errors in a domain automatically, while at the same time attempting to repair the domain. In contrast to existing PDDL debuggers, it should be able to detect a wider range of errors, such as missing preconditions, missing negative effects, or unnecessary positive

effects. The debugger should take the domain as input, as well as multiple pairs of problem instances and preferred plans. Additionally, background knowledge about the domain can be provided. The debugger should change the domain so that all given plans are valid and the background knowledge is respected by the domain. One major challenge to overcome is the complexity of planning domains, because the number of possible effects, or preconditions, grows rapidly with each new parameter. When an error is encountered, multiple different adjustments can be made, followed by another set of adjustments, leading to an exponential cascade of solutions. Another obstacle is the large set of features present in modern PDDL.

The main contributions of the paper are a clear notation for PDDL debugging using preferences and background knowledge, a methodology for debugging PDDL if such information is given, and an implementation of the approach as a tool ¹.

The remainder of the paper is organized so that we first discuss related work in the validation and debugging of PDDL domains. The next section introduces the notations, considered errors, and basic methodology. This is followed by a detailed description of the developed approach. In the next section, an experimental evaluation with common planning domains is presented, while in the final section we draw some conclusions and outline future work.

2 Related Research

The problem of debugging, validating, and repairing planning domains is an active topic of research with many contributions in recent years. In this section, we present various approaches made towards providing assistance to PDDL developers.

VAL (Plan Validator), originally developed by [9] to verify planners, checks if a given plan solves a specific planning problem. It simulates the initial state and evaluates the plan step-by-step, checking preconditions, applying effects, and verifying the final goal. If a precondition fails, it marks the plan invalid and reports a detailed violation explanation. Although VAL can be repurposed for domain debugging by assuming the plan is correct and the domain is faulty, it lacks automation; a human developer must still manually determine and apply the necessary domain repairs.

Another domain repair tool, presented by [4], addresses planning problems that fail due to over-constrained preconditions. Using a domain and problem instances as input, it iteratively modifies preconditions to relax the domain until a solution is found. Driven by the Delta Debugging algorithm, the tool alters precondition components and invokes a planner to find a local minimum of removed constraints. This black-box approach requires neither human intervention nor preferred plans, allowing it to support advanced PDDL features like temporal planning. However, it is strictly limited to fixing a single error type: additional preconditions.

Domain repair for incomplete positive effects was addressed by [8]. When a planning problem has no solution, their algorithm adds positive effects to relax the domain and make the goal reachable. They encode this repair process as a planning problem itself, using auxiliary actions and fluents that dynamically modify actions. The planner returns both a valid plan and the required domain modifications. To counter the planner's natural bias toward excessive repair (adding extraneous effects to shorten plans), the authors penalize each repair made. A user interface was later introduced in [7].

The most advanced debugger we are aware of was presented by [10] and [11]. In their first paper, they mapped domain repair for grounded STRIPS domains to a diagnosis problem solved via a Minimal Hitting Sets algorithm. The debugger relaxes the domain to validate

¹ The implementation is publicly available under <https://git.ist.tugraz.at/ais/pddl-debugger>

an incorrect preferred plan by removing preconditions, adding positive effects, or removing negative effects. In their second paper, they extended the tool to support lifted STRIPS domains, multiple problem-plan pairs, and negative preconditions (allowing the removal of negative preconditions/positive effects or the addition of negative effects). However, this debugger still fails to address scenarios where the preferred plan is already valid, but the planner returns a different, incorrect plan as the solution.

An exploratory study by Gragera and Pozanco investigated the utility of Large Language Models (LLMs) like ChatGPT for debugging planning domains. In each test case, they prompted the LLM with an unsolvable planning problem – consisting of a domain and a problem instance – alongside a brief task description. Their findings indicated that, given their current capabilities, LLMs fail to provide meaningful assistance to planning domain developers.

Another notable tool created to assist PDDL developers is PDDL Studio [15]. It is a complete IDE for PDDL that is capable of syntax highlighting and marking compile errors. Furthermore, it provides a graphical interface for various planners. A similar collection of tools is available as an add-on for the popular text editor Visual Studio Code [3]. It also integrates features from planutils, a collection of tools for planning created by Muise [14].

An exploratory study by Sleath et al. systematically cataloged PDDL domain errors to evaluate PDDL parsers, creating a repository of both syntactical and semantical mistakes [17]. Syntactic errors include basic formatting slips like missing closing parentheses, while semantic errors cover logical flaws such as attempting to access an undefined variable. However, this repository fails to capture higher-level design errors, such as missing effects or redundant preconditions.

Regarding integrity constraints for situation calculus, the work presented in [16] has laid the groundwork for proving that properties hold for all situations reachable from the initial situation using induction as a proof technique.

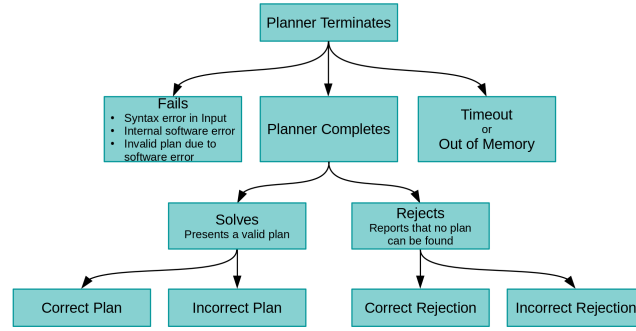
3 Methodology

3.1 Basic Notations

Figure 2 depicts the possible results of running a planner on a planning problem. The planner may **fail** in providing meaningful results due to internal software errors, syntax errors in the domain model, or algorithmic errors. The planner may not be able to provide meaningful results due to **limited computational resources** or domain complexity. The planner may terminate normally (**completes**) and present a valid result regarding the given planning problem. The planner completes and presents a valid plan for the problem (**solves**). The planner completes and states that no valid plan exists for the given planning problem (**rejects**).

A plan is **valid** if it fulfills all requirements given by the domain and problem description, meaning that at each step, the respective preconditions hold and the goals are met in the end. Conversely, a plan is **invalid** if a precondition is violated or the goal is not met. A plan is **correct** if it is the solution to a concrete planning problem and can be executed in the corresponding world showing the intended behavior. An **incorrect** plan is either infeasible or does not achieve the goals intended in the corresponding world. For example, a robot may try to pick up an object that is not in place. Please note that a valid plan isn't necessarily correct, while a correct plan isn't necessarily valid given a domain model.

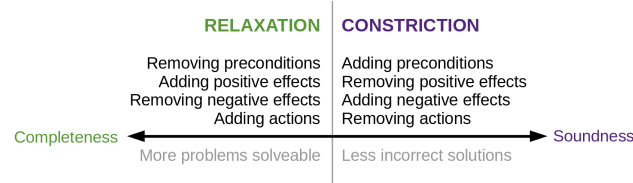
Figure 3 illustrates how relaxing or constricting a domain shifts its boundaries, allowing us to formally define soundness and completeness in PDDL modeling. A domain is **sound** if all valid plans are correct, and it is **unsound** if an invalid plan is permitted. **Constricting** a



■ **Figure 2** The potential outcomes of running a planner on a planning problem.

domain by adding preconditions, removing positive effects, or adding negative effects restricts the set of valid plans and increases soundness. Conversely, a domain is **complete** if all correct plans are valid, and it is **incomplete** if a correct plan is rejected. **Relaxing** a domain by removing preconditions, adding positive effects, or removing negative effects expands the valid plan space and increases completeness.

A domain is correct only when it is simultaneously sound and complete. If it is incorrect, modeling errors cause it to report incorrect plans or false rejections, failing to accurately reflect the real-world behavior or the modeler’s intent.



■ **Figure 3** Relation of relaxation and constriction of a domain to changes in the domain.

3.2 Error Catalog and Fixes

We identified 3 error types related to modeling: (1) syntax errors in the domain model and the problem instance, (2) errors in the problem instance, and (3) errors in the domain model. We assume that the first type is handled by the planner. The second type is beyond the scope of this work to use preferred plans and background knowledge for domain debugging. Table 1 provides a list of modeling errors we aim to automatically debug, along with a detailed description and the potential consequences.

We define a **fix** as an atomic modification to a domain. We focus on altering only the preconditions and effects of domains. There are other types of fixes, such as altering parameters of actions, adding or removing fluents, and creating or deleting actions. This is beyond the scope of this work. Applying a fix fix to a domain d gives a new domain d' : $d' = apply_fix(d, fix)$. We also allow multiple fixes to be applied to a domain: $d' = apply_fix(apply_fix(d, fix_1), fix_2) = apply_fix(d, \{fix_1, fix_2\})$.

The application of a fix to a domain is idempotent, meaning that the domain doesn’t change if the same fix is applied one more time: $apply_fix(d, fix) = apply_fix(apply_fix(d, fix), fix)$.

Table 2 lists the types of fixes considered and the notation used to denote them.

■ **Table 1** Catalog of considered modeling errors and their consequence on the planning results: domain relaxation (RE), domain constriction (CO), jeopardize domain correctness (CR).

error type	error description	consequence
precondition missing	a literal is missing in the precondition	RE
additional precondition	an additional incorrect precondition is present	CO
positive effect missing	a positive literal is missing in the precondition	CO
additional positive effect	an additional incorrect positive literal is present	RE
additional negative effect	an additional incorrect negative literal is present	CO
negative effect missing	a negative literal is missing in the precondition	RE
action parameter missing	a action parameter is missing and most likely a precon.	RE
fluent missing	a fluent is missing and modeling of the related impact	CC
action missing	an action is missing and modeling of the related behavior	CC
parameter mismatch precon.	wrong parameter used in precondition	CC
parameter mismatch effect	wrong parameter used in effect	CC
incorrect effect polarity	mismatch between positive and negative effect	CC

■ **Table 2** The types of fixes considered, along with the notation we introduce to denote them.

Fix	Meaning
$\langle a.C + c \rangle$	Adding precondition c to the action a .
$\langle a.C - c \rangle$	Removing precondition c from the action a .
$\langle a.E^+ + e \rangle$	Adding positive effect e to the action a .
$\langle a.E^+ - e \rangle$	Removing positive effect e from the action a .
$\langle a.E^- + e \rangle$	Adding negative effect e to the action a .
$\langle a.E^- - e \rangle$	Removing negative effect e from the action a .

3.3 Background Knowledge

We express background knowledge as a set of state integrity axioms (SIA) that hold if a situation is consistent in the modeled world. If an action is taken in a consistent situation leading to an inconsistent situation, then an error must have been made in modeling that action. Background knowledge is intrinsic to the planning problem itself, but it isn't necessary to incorporate it in the planning domain directly, as discussed by Mueller [13]. For instance, it is reasonable that a robot can only be in one location at once in the delivery robot domain.

We state that a domain model D is incorrect if there exists a valid plan that leads to a state where an SIA doesn't hold. The SIA must hold for all valid plans. This means that for all valid states, all possible actions must lead to another state for which the SIA holds, which is similar to concepts presented by Reiter [16].

$$\begin{aligned}
& \text{correct}(D) \rightarrow [\forall I, G, \pi : \text{valid}(D, I, G, \pi) \\
& \rightarrow \forall a_i \in \pi, s_i : s_i = \gamma(s_{i-1}, a_{i-1}) \wedge \text{SIA}(s_i)]
\end{aligned} \tag{1}$$

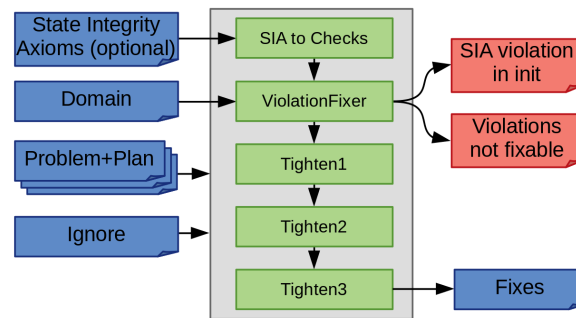
An SIA can be expressed using a syntax similar to logic expressions in PDDL of the form `(operator (fluent1 term1_1 term1_n) (fluentN termN_1 termN_N))`. For instance, the SIA `(always (f t1 t2))` encodes the constraint that at least one atomic formula of f needs to be part of the state S : $\forall t_1 \in O : \exists t_2 \in O : \langle f, t_1, t_2 \rangle \in S$. An example is `(always (isat r1 ?))` encodes that robot `r1` needs to be at least at one location at all times, where `?` encodes a wildcard. Consistency checks can be executed efficiently using the techniques proposed by Giacomo and Mancini [2]. Besides `always`, the debugger supports `not`, `binary`, `implies`, `nand`, `xor`, and `and` in SIA.

Hypotheses for the violation are reported as a set of tuples $\langle positive, prev, modified \rangle$ where *positive* is true if the atom is present, but shouldn't be, and vice versa, *prev* holds the atom if established already before the evaluated action, and *modified* holds the atom if established by the evaluated action. For instance, if in the action *move* the positive effect for the target plocation is missing, the following hypotheses are given: $\{\langle positive : \perp, modified : (at\ r1\ la) \rangle, \langle positive : \perp, prev : (at\ r1\ ?) \rangle\}$.

3.4 PDDL Debugger

DrPDDL takes a PDDL domain, a set of State Integrity Axioms, a list of at least one pair of a problem instance and preferred plan as well as a set of fixes to ignore as input. It first checks whether any action preconditions or SIAs are violated when evaluating the plans. It reports these violations and attempts to fix them by adding or removing preconditions and effects. Under the assumption that any behavior outside of the behavior represented in the preferred plans is unwanted behavior, DrPDDL constricts the domain as much as possible while still ensuring that the plans are valid and no SIAs are violated. Finally, it gives a list of fixes as output, as shown in Algorithm 1.

Currently, our debugger is limited to PDDL domains, making use of only the *strips* and *typing* requirements. It also cannot add new actions, remove actions, or add new parameters to actions, since new parameters would require the provided plans to be revised during the process of debugging them. It is also limited to debugging planning domains only. It cannot debug problem instances by explaining how the initial situation needs to be altered for the goal to become achievable.



■ **Figure 4** A flowchart illustrating the input, output, and components of the proposed debugger.

Internally, the debugger makes use of various algorithms, as shown in Figure 4. At first, it loads and parses the input. The SIAs are translated into a list of checks. A check is a short procedure that is executed after each step in the plans and checks for a specific change regarding an SIA. If a violation is found, hypotheses about the cause of the violation are reported.

The VIOLATIONFIXER is responsible for changing a domain so that it doesn't violate any preconditions of the plans or SIAs during the evaluation of those plans. While fixing violations of preconditions leads to a domain relaxation, fixing a violation of an SIA may lead to either a relaxation or a constriction. It uses a breadth-first search strategy where the plans are validated, and when a violation is encountered, one or more fixes are proposed. It succeeds if a set of fixes is found so that all plans are valid, and it fails if all branches have been searched without success.

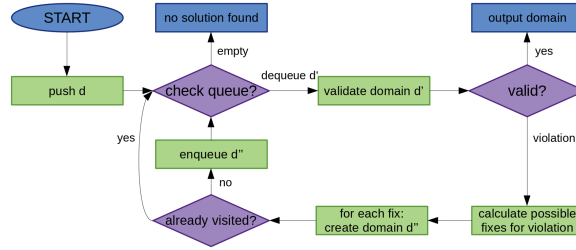
■ **Algorithm 1** Main function of our debugger that calls all core algorithms, incrementally refining the domain.

```

1: function MAIN(d, plans, SIAs, ignore)
2:   checks  $\leftarrow$  SIATOCHECKS(SIAs)
3:   d  $\leftarrow$  VIOLATIONFIXER(d, plans, checks, ignore)
4:   d  $\leftarrow$  TIGHTEN1(d, plans, checks, ignore)
5:   d  $\leftarrow$  TIGHTEN2(d, plans, checks, ignore)
6:   d  $\leftarrow$  TIGHTEN3(d, plans, checks, ignore)
7:   return d
8: end function

```

VIOLATIONFIXER (Algorithm 2) takes a domain *d*, problem-plan pairs *plans*, a set of checks *checks*, and a set of ignored fixes *ignore* as parameters. It initializes a queue with the initial domain *d* and creates a set for visited domains. The algorithm loops until the queue is empty, extracting a domain *d'* from the front. If *d'* was already visited, it is ignored. Otherwise, the plans are validated using *d'*. If the plans are valid, the algorithm terminates and returns *d'*. Otherwise, it calculates possible fixes for the violation, applies them to *d'*, and pushes the resulting domains back to the queue. This breadth-first search strategy inherently favors domains requiring fewer fixes. If the queue becomes empty because no further fixes can be generated, the algorithm fails. Figure 5 outlines this control flow.



■ **Figure 5** A flowchart showing the steps of the VIOLATIONFIXER algorithm.

The method for calculating fixes is determined by the specific type of violation encountered. When a violation occurs within a check, the algorithm calls the function CALCFIXES1 for each component of that violation. This function generates distinct sets of fixes, each of which is applied to the domain before being added to the queue. Alternatively, if the violation is found within a goal or a precondition, the fixes are determined by the function CALCFIXES2. Furthermore, if the issue stems specifically from a precondition, an additional domain where that precondition has been entirely removed is also generated and enqueued.

In order to derive fixes from these components, the following rules are applied, as reflected in Algorithm 3 in the same order. If an atom is already present, a negative effect of removing it is introduced. For example, if the SIA (xor A B) is violated because A was already set when B is applied, a possible fix is adding the effect (not A). If an atom were added to the violating action, the positive effects of creating it are removed. In the (xor A B) example, another fix is removing the effect (B) so only A remains. If both cases above are true, the Cartesian product of the fixes is returned, applying both a negative effect addition and a positive effect removal simultaneously. If B and A' are set while A is present, the fix set is $\{ \langle a.E^- + \langle \text{not } A \rangle \rangle, \langle a.E^+ - \langle A \rangle \rangle \}$. If an atom is already absent, either a positive effect creating it or a precondition testing for it is added. For example, if (xor A B) is violated when B is removed because A is absent, a fix is adding the positive effect (A). If an atom was removed in the violating action, either the negative effect removing it is deleted, or a positive

■ **Algorithm 2** VIOLATIONFIXER. It takes a domain, a list for problem-plan pairs, a list of checks and a list of fixes to ignore as parameters and returns a list of fixes necessary to uphold all action preconditions and SIAs.

```

1: function VIOLATIONFIXER(d, plans, checks, ignore)
2:   queue  $\leftarrow \langle \rangle$ 
3:   enqueue(d)
4:   searched  $\leftarrow \emptyset$ 
5:   while d' = dequeue() do
6:     if d'  $\in$  searched then
7:       continue
8:     end if
9:     searched  $\leftarrow$  searched  $\cup \{d'\}$ 
10:    result  $\leftarrow$  VALIDATOR(d', plans, checks)
11:    if result = VALID then
12:      return d'
13:    end if
14:    if result occurred in check then
15:      for all c in result.violation.components do
16:        for all f in CALCFIXES1(c, result.s) do
17:          enqueue(apply_fix(d', f \ ignore))
18:        end for
19:      end for
20:    else
21:      r  $\leftarrow$  result.precondition(result.s)
22:      for all f in CALCFIXES2(d', result.s, r) do
23:        enqueue(apply_fix(d', f \ ignore))
24:      end for
25:      if result occurred in precondition then
26:        enqueue(apply_fix(d',  $\{\langle \text{result.s.a.C} -$ 
27:          result.pc  $\rangle\} \setminus \text{ignore}$ ))
28:      end if
29:    end if
30:  end while
31:  return NULL
32: end function

```

effect adding an alternative atom is introduced (e.g., removing (not B) or adding (B')). If both absent/removed cases are true, the Cartesian product of those fixes is returned. If (or A B) is violated because A is absent when the action removes B, and a “blind” negative effect also removes A, the fix set is $\{\langle a.E^- - \text{'(not A)'} \rangle, \langle a.E^+ + \text{'(A)'} \rangle\}$.

When an action precondition or planning goal is violated, the function in Algorithm 3 seeks fixes to satisfy it by traversing the plan in reverse, starting immediately before the violation. At each step, the algorithm attempts to introduce a positive effect that creates the required atom. However, if it encounters an action containing a negative effect that explicitly removes that atom, an alternative fix is generated to delete that negative effect (regardless of whether the atom was present prior to that action). Once this occurs, the reverse loop terminates and returns the accumulated set of fixes.

The Tighten1 algorithm, outlined in Algorithm 5, constricts a domain by removing as many positive effects as possible without compromising its validity. The simplest approach would be to eliminate all positive effects whose created atoms are never subsequently queried by a precondition. However, multiple positive effects can generate the exact same atom for a downstream precondition, meaning redundant effects can be safely removed. As illustrated in Figure 6 and Figure 7 left, this challenge can be transformed into a Minimal Hitting Set problem. Here, each evaluated precondition contributes a set containing all positive effects capable of creating its required atom. By solving for the minimal hitting set, the algorithm identifies the absolute minimum collection of effects necessary to satisfy all preconditions. Any positive effects outside this set are then removed from the domain.

■ **Algorithm 3** Function `CALCFIXES1` used by the `VIOLATIONFIXER` algorithm. It takes a component v of a violation and a step s and returns a set of all fixes that can be used to prevent the violation.

```

1: function CALCFIXES1( $v, s$ )
2:   if  $v.positive$  then
3:     if  $v.prev \neq \text{NIL} \wedge v.mo = \text{NIL}$  then
4:        $F = \{\{ \langle v.a.E^- + x \rangle \} | x \in \text{EC}(v.a, v.p)\}$ 
5:       return  $F$ 
6:     else if  $v.prev = \text{NULL} \wedge v.m \neq \text{NULL}$  then
7:        $F = \{\{ \langle v.a.E^+ - e \rangle \} | e \in v.a.E^+ \wedge$ 
8:          $e(s) = v.m\}$ 
9:       return  $F$ 
10:    else if  $v.prev \neq \text{NULL} \wedge v.m \neq \text{NULL}$  then
11:       $F1 \leftarrow \{\{ \langle v.a.E^- + x \rangle \} | x \in \text{EC}(v.a, v.p)\}$ 
12:       $F2 \leftarrow \{\{ \langle v.a.E^+ - e \rangle \} | e \in v.a.E^+ \wedge$ 
13:         $e(s) = v.m\}$ 
14:      return  $F1 \times F2$ 
15:    end if
16:  else
17:    if  $v.prev \neq \text{NIL} \wedge v.m = \text{NIL}$  then
18:       $F1 \leftarrow \{\{ \langle v.a.E^+ + x \rangle \} | x \in \text{EC}(v.a, v.p)\}$ 
19:       $F2 \leftarrow \{\{ \langle v.a.C + x \rangle \} | x \in \text{EC}(v.a, v.p)\}$ 
20:      return  $F1 \cup F2$ 
21:    else if  $v.prev = \text{NIL} \wedge v.modified \neq \text{NIL}$  then
22:       $F1 \leftarrow \{\{ \langle v.a.E^- - e \rangle \} | e \in v.a.E^- \wedge$ 
23:         $e(s) = v.m\}$ 
24:       $F2 \leftarrow \{\{ \langle v.a.E^+ + x \rangle \} | x \in \text{EF}(v.a, v.m)\}$ 
25:      return  $F1 \cup F2$ 
26:    else if  $v.prev \neq \text{NIL} \wedge v.m \neq \text{NIL}$  then
27:       $F1 \leftarrow \{\{ \langle v.a.E^+ + x \rangle \} | x \in \text{EF}(v.a, v.p)\}$ 
28:       $F2 \leftarrow \{\{ \langle v.a.C + x \rangle \} | x \in \text{EF}(v.a, v.p)\}$ 
29:       $F3 \leftarrow \{\{ \langle v.a.E^- - e \rangle \} | e \in v.a.E^- \wedge$ 
30:         $e(s) = v.m\}$ 
31:      return  $(F1 \cup F2) \times F3$ 
32:    end if
33:  end if
34: end function

```

■ **Algorithm 4** Function `CALCFIXES2` used by the `VIOLATIONFIXER` algorithm. It takes a domain, step and atom queried by a precondition as argument and returns a list of fixes in order for the precondition to hold.

```

1: function CALCFIXES2( $d, s, r$ )
2:    $F \leftarrow \emptyset$ 
3:   while  $s.num > 0$  do
4:      $s \leftarrow s - 1$ 
5:     if  $\exists e \in s.a.E^- : e(s) = r$  then
6:        $F \leftarrow F \cup \{\{ \langle s.a.E^- - e \rangle \}\}$ 
7:       break
8:     end if
9:     for all  $x$  in  $\text{EE}(r, s.args)$  do
10:       $fixes \leftarrow fixes \cup \{\{ \langle s.a.E^+ + x \rangle \}\}$ 
11:    end for
12:  end while
13:  return  $F$ 
14: end function

```

Using a Minimal Hitting Set algorithm, it is also possible to mark effects that must be included in the resulting hitting set, as shown in Figure 7 right. We use this feature to mark all effects contained in the ignore set as well as all effects in the initial situations, since they should not be removed. In the end, the resulting effects are removed one by one, with a call to the Validator algorithm being made each time. If the removal of one effect results in an invalid domain, the removal will be reversed.

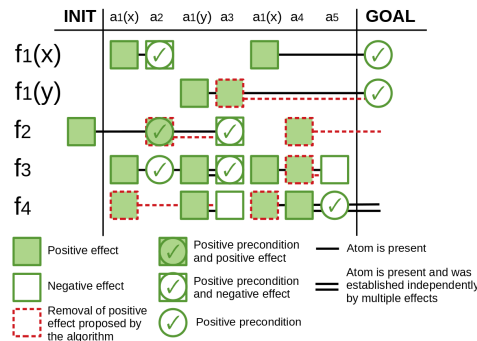
The function `ExtendPlan` simplifies the evaluation of the initial situation and goal. It constructs a new action `init` from the initial situation of a given problem and prepends it to the beginning of the plan. This action has no parameters or preconditions, and the atoms of the initial situation make up its positive effects. Next, it constructs another action `goal` from the problem's goal and appends it to the end of the plan. It has no parameters or effects, but uses the planning goal as its precondition. Finally, the function returns the new plan, an approach also used in plan-space planning [6].

■ **Algorithm 5** `TIGHTEN1`. It takes a domain, a list of problem-plan pairs, a list of checks compiled from the SIAs and a set of changes to be ignored and it then removes as many positive effects as possible from the domain.

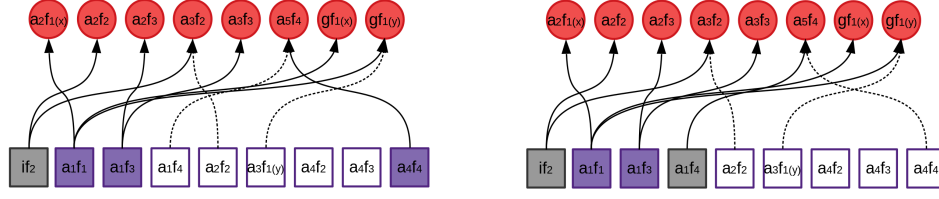
```

1: function TIGHTEN1( $d$ ,  $plans$ ,  $checks$ ,  $ignore$ )
2:    $sets \leftarrow \emptyset$ 
3:   for all  $\langle problem, plan \rangle$  in  $plans$  do
4:     for all  $e$  in  $problem.init$  do
5:        $ignore \leftarrow ignore \cup \{e\}$ 
6:     end for
7:      $plan \leftarrow \text{EXTENDPLAN}(plan, problem)$ 
8:      $S \leftarrow \emptyset$ 
9:     for all  $s$  in  $plan$  do
10:      for all  $c$  in  $s.a.C$  do
11:         $r \leftarrow c(s)$ 
12:         $sets \leftarrow sets \cup S_r$ 
13:      end for
14:      for all  $e$  in  $s.a.E$  do
15:         $r \leftarrow e(s)$ 
16:        if  $e$  is positive then
17:           $S_r \leftarrow S_r \cup \{e\}$ 
18:        else
19:           $S_r \leftarrow \emptyset$ 
20:        end if
21:      end for
22:    end for
23:  end for
24:   $removeable \leftarrow d.E^+ \setminus MHS(sets, ignore)$ 
25:  for all  $e$  in  $removeable$  do
26:     $d' \leftarrow \text{apply\_fix}(d, \langle e.a.E^+ - e \rangle)$ 
27:    if VALIDATOR( $d'$ ,  $plans$ ,  $checks$ ) then
28:       $d \leftarrow d'$ 
29:      yield  $\langle e.a.E^+ - e \rangle$ 
30:    end if
31:  end for
32: end function

```

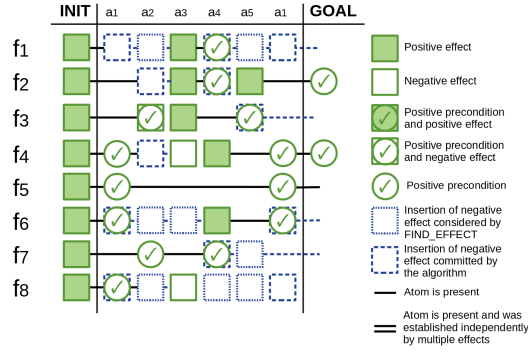


■ **Figure 6** Example scenario demonstrating the `TIGHTEN1` algorithm.



■ **Figure 7** Left: The graph constructed from the scenario illustrated in Figure 6 as well as a possible set of effects that can be removed, marked with white background. Effects in the initial situation are by default blacklisted. Right: Another possible solution for the scenario in Figure 6 with the effect a_1f_4 also blacklisted this time. Another possible solution for the example scenario with one additional effect blacklisted.

The TIGHTEN2 algorithm is a greedy procedure that adds as many negative effects to a domain as possible to constrict it. To preserve plan validity, effects are introduced only where existing atoms are present but not subsequently required by a precondition. The algorithm uses a heuristic based on the proximity of removable atoms to the latest preconditions to select the best candidate effect. Once added, the VALIDATOR checks for State Integrity Axiom (SIA) violations. If a violation is found, the effect is removed and placed in an ignore set. As defined in Algorithm 6, this process repeats until no further effects can be found. Figure 8 illustrates this behavior for a hypothetical domain and plan.



■ **Figure 8** A hypothetical domain and plan showing the set of negative effects the TIGHTEN2 algorithm adds.

In each iteration, the most suitable negative effect to be added is calculated using the function in Algorithm 7. This function utilizes mutable records passed by reference. First, the EXPRENUMERATOR calculates the entire set of addable negative effects for all actions and fluents. For each effect, a record is created containing the effect itself, a score initialized to zero, and a validity field initialized to $\top$. Finally, these records are stored in a list and bucketized by their corresponding action.

Thereafter, the algorithm iterates over each problem–plan pair and each step of the plan, including the initial situation and goal state. For every atom in the current situation, it stores the creation step, the step in which it was last used as a precondition, and a list of encountered candidate negative effects. At each step, the action’s preconditions are evaluated. When an atom is queried, its last-use index is updated, and any previously encountered candidate negative effects for that atom are marked invalid, since they would remove the atom and invalidate the plan. The action’s effects are then applied. Positive effects create new atom records with the current step as their creation and last-use index, while negative effects

remove the corresponding atom records. Finally, candidate negative effects of the action are considered. If the resulting atom exists in the current situation and was created before the current step, the candidate is added to the atom's effects. Its score is also increased according to a heuristic based on the distance to the atom's last use, with higher scores assigned to atoms removed shortly after their last use. Finally, the candidate with the highest score is picked out of the set of all valid candidates with scores higher than 0. If no such candidate exists, NULL is returned.

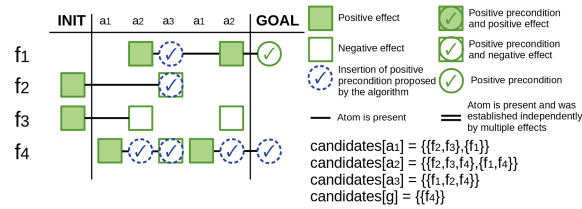
■ **Algorithm 6** TIGHTEN2 algorithm. It takes a domain, a list of problem-plan pairs, a list of checks compiled from the SIAs and a set of changes to be ignored and adds as many negative effects as possible to the domain.

```

1: function TIGHTEN2( $d, plans, checks, ignore$ )
2:   while  $e \leftarrow \text{FINDEFFECT}(d, plans, checks, ignore)$  do
3:      $d' \leftarrow \text{apply\_fix}(d, \langle e.a.E^- + e \rangle)$ 
4:     if  $\text{VALIDATOR}(d', plans, checks)$  then
5:        $d \leftarrow d'$ 
6:       yield  $\langle e.a.E^- + e \rangle$ 
7:     else
8:        $ignore \leftarrow ignore \cup \{e\}$ 
9:     end if
10:  end while
11: end function

```

The TIGHTEN3 algorithm (Algorithm 8) adds as many preconditions as possible while preserving the validity of the given plans. It processes all plans and computes, for each action occurrence, the set of preconditions that could be added based on the current situation. These candidate sets are grouped by action. For each action, all collected sets are intersected, yielding the preconditions that hold whenever the action is executed. A precondition is reported unless it is already present in the action, belongs to the goal, or is contained in the *ignore* set. Figure 9 shows how the TIGHTEN3 algorithm adds preconditions in a hypothetical scenario as well as the resulting data structure.



■ **Figure 9** An example plan and domain showing the set of preconditions the TIGHTEN3 algorithm can add. The resulting data structure is also shown. For each step, the set of all possible additional preconditions is calculated that is then bucketized by action. The result is obtained using the intersection of the sets of preconditions within each bucket. Example scenario demonstrating the Tighten4 algorithm.

4 Evaluation

4.1 Metrics

To evaluate the effectiveness of our proposed PDDL debugger, we first define appropriate metrics. We evaluate how many errors our debugger can detect and repair in a faulty domain. Each testcase consists of a domain containing a set of errors M , for which the debugger

■ **Algorithm 7** The `FIND_EFFECT` function is used by the `TIGHTEN2` algorithm. It calculates all sensible negative effects that can be added to a domain and returns the best-suited fix.

```

1: function FIND_EFFECT(d, plans, checks, ignore)
2:   candidates ← ∅
3:   for all a in d do
4:     for all f in d.F do
5:       for all x in EXPRENUMERATOR(a, f) do
6:         candidatesa ← candidatesa ∪ {⟨e : x, score : 0, valid : ⊤⟩}
7:       end for
8:     end for
9:   end for
10:  for all ⟨problem, plan⟩ in plans do
11:    plan ← EXTENDPLAN(plan, problem)
12:    S ← ∅
13:    for all s in plan do
14:      for all c in s.a.C do
15:        r ← c(s)
16:        Sr.last_use ← s.num
17:        for all e in Sr.encountered do
18:          e.valid ← ⊥
19:        end for
20:      end for
21:      for all e in s.a.E do
22:        r ← e(s)
23:        if e is positive then
24:          Sr ← ⟨origin : s.num, last_use : s.num, encountered : ∅⟩
25:        else
26:          remove Sr
27:        end if
28:      end for
29:      for all candidate in candidatesa do
30:        r ← candidate.e(s)
31:        if r ∈ S ∧ Sr.origin < s.num then
32:          add candidate to Sr.encountered
33:          candidate.score ← candidate.score + 1/(s.num - Sr.last_use + 1)2
34:        end if
35:      end for
36:    end for
37:  end for
38:  remove from candidates where ¬candidate.valid ∨ candidate.score = 0
39:  if candidates = ∅ then
40:    return NULL
41:  else
42:    return  $\underset{candidate \in candidates}{argmax} \quad candidate.score$ 
43:  end if
44: end function

```

returns a set of fixes F . The correctly identified errors are the true positives $M \cap F$. Since the number of errors differs between testcases, we measure performance using the recovery rate rec defined in Definition 1. False positives are given by $F \setminus M$. We report the false positive rate from Definition 2, which measures the fraction of reported incorrect fixes. Additionally, we record the runtime t required to process each testcase.

► **Definition 1.** *The recovery rate is defined as the rate of correctly identified errors among the introduced errors.*

$$rec = \frac{|M \cap F|}{|M|}$$

► **Definition 2.** *The false positive rate is defined as the rate of false positives among all reported errors. If no errors are reported, fp is 0.*

$$fp = \frac{|F \setminus M|}{\max(|F|, 1)}$$

■ **Algorithm 8** TIGHTEN3. It takes a domain, a list of problem-plan pairs, a list of checks compiled from the SIAs and a set of changes to be ignored as input and adds as many preconditions as possible to the domain.

```

1: function TIGHTEN3(d, plans, checks, ignore)
2:   buckets  $\leftarrow \emptyset$ 
3:   for all  $\langle \text{problem}, \text{plan} \rangle$  in plans do
4:     plan  $\leftarrow \text{EXTENDPLAN}(\text{plan}, \text{problem})$ 
5:     S  $\leftarrow \emptyset$ 
6:     for all s in plan do
7:       for all r in S do
8:         fixes  $\leftarrow \text{FIXENUMERATOR}(r, s.\text{args})$ 
9:         bucketss,a  $\leftarrow \text{buckets}_{s,a} \cup \{\text{fixes}\}$ 
10:      end for
11:      apply s to S
12:    end for
13:  end for
14:  for all  $\langle a, \text{candidates} \rangle$  in buckets do
15:    matches  $\leftarrow \bigcap_{\text{candidate}} \text{candidate}$ 
16:    for all c in matches do
17:      if  $c \notin a.C \wedge c \notin \text{ignore}$  then
18:        yield  $\langle a.C + c \rangle$ 
19:      end if
20:    end for
21:  end for
22: end function

```

When running a large number of similar testcases, we can just calculate the average recovery rate rec_μ as well as the average false positive rate fp_μ . Similarly, we calculate the average runtime t_μ and its standard deviation t_σ .

To evaluate the core design decisions, we investigate the impact of background knowledge on result quality, determine which error types are easiest to detect and repair, assess the benefit of preemptively blacklisting false positives, analyze how the recovery rate varies with the number of errors present, and compare the results for domains that are valid and invalid with respect to the preferred plans. The last question targets scenarios where a preferred plan is valid, but the planner returns an incorrect solution. To test these, our debugger's interface allows adjusting the following meta-parameters: the sequence of algorithms used, whether to use all, some, or no SIAs, and which fixes to exclude from the search space. To answer these questions, we considered the following debugger configurations:

- **normal**: All plans available, all algorithms used, all SIA used, no fixes excluded.
- **no tighten**: All plans available, only VIOLATIONFIXER used, all SIA used, no fixes excluded.
- **no SIA**: All plans available, all algorithms used, no SIAs used, no fixes excluded.
- **with ignored**: First, the debugger is executed the same way as in **normal**. Then, the set of false positives is calculated, and the debugger is run again with that set being excluded.
- **single**: Only one plan available at a time, all algorithms used, all SIA used, no fixes excluded. This results in the number of runs being multiplied by the number of plans available for that domain.

4.2 Setup

We rely on realistic domains written by various developers. The archive of the International Planning Competition (IPC) [12] offers a large set of domains of various complexities. It also contains dozens of problem instances for each domain. The more challenging aspect is to introduce errors into those domains that are likely to occur when the domain is written by a human. We filter out all domains with PDDL features not supported by DrPDDL, meaning that we only keep domains that only have both **strips** and **typing** as requirements.

Subsequently, we use the Optic planner [1] to generate plans for these domains. We discard any domain where fewer than 10 problem instances yielded valid plans due to timeouts, further narrowing down the set of suitable domains. Then we sort out domains that are very similar in their structure and purpose, such as **driverlog-strips-hand-coded** and **driverlog-numeric-hard-automatic**, narrowing down to 10 domains listed in Table 3. We manually specify the background knowledge for each domain. To make sure that background knowledge is correct, we validate each domain using the previously generated plans.

■ **Table 3** The domains used along with metrics regarding their complexity. It shows the name, the number of actions $|A|$, number of fluents $|F|$, total number of preconditions $|C|$, the total number of positive and negative effects $|E^+|$ and $|E^-|$ respectively, the number of steps in all plans $|P|$ as well as the number of SIAs $|SIA|$.

Domain Name	$ F $	$ A $	$ C $	$ E^+ $	$ E^- $	$ P $	$ SIA $
2000-blocks-strips-typed	5	4	9	9	9	336	8
2000-freecell-strips-typed	11	10	61	26	30	243	7
2000-logistics-strips-typed	3	6	12	6	6	360	6
2002-driverlog-strips-automatic	6	6	14	7	7	303	6
2002-satellite-strips-automatic	8	5	15	5	4	398	3
2002-zenotravel-strips-automatic	4	5	14	7	7	189	6
2006-pipesworld-propositional	15	6	57	26	26	346	12
2006-rovers-propositional	25	9	45	17	13	313	12
2006-storage-propositional	8	5	18	10	10	158	8
2006-tpp-propositional	7	4	17	7	7	566	3

To test a range of error types in each domain, we decided to automatically add errors, allowing for the generation of hundreds of faulty mutants from one domain.

Regarding the types of errors, we will use the error catalog presented in Section 3 as a reference. In our quantitative evaluation, we will only consider errors our debugger is suited for, which are: **pre rem** - missing precondition, **pre add** - additional precondition, **pre rep** - error in a precondition, **pos rem** - missing positive effect, **pos add** - additional positive effect, **pos rep** - error in a positive effect, **neg rem** - missing negative effect, **neg add** - additional negative effect, **neg rep** - error in a negative effect, and **polarity** - wrong polarity of an effect.

As for missing actions, fluents, or action parameters, we decide to exclude those error types from our quantitative evaluation and will briefly discuss them in a qualitative evaluation.

We start by creating a list of all reasonable errors grouped by type for each domain. Additionally, we determine the set of dynamic fluents for each domain. We call a fluent dynamic if it changes after the initial situation, meaning that there's at least one effect for that fluent.

Error types **pre rem**, **pos rem**, **neg rem**, and **polarity** are easy to generate because it involves iterating over all preconditions and effects and then removing that element or changing its polarity.

Then, we inject errors of the types **pre rep**, **pos rep**, and **neg rep** by iterating over each parameter and checking if it can be replaced by another of the same type. For example, we can replace a precondition (`edge ?from ?to`) in the action `move` with (`edge ?to ?to`), (`edge ?to ?from`), or (`edge ?from ?from`) since the parameters `?from` and `?to` share the same type.

For error types with element additions, namely **pre add**, **pos add**, and **neg add**, we generate them by iterating over each action in the domain. For each fluent, we make use of the `EXPRENUMERATOR` to generate syntactically valid expressions based on the parameters of the action. Additionally, it is reasonable to add an effect for a fluent only if that fluent is dynamic. For example, we exclude errors such as the effect (`not (edge ?from ?to)`) since this fluent doesn't change in the original domain.

Having obtained a list of reasonable errors, we generate faulty domains by randomly picking a set of errors of a given type from that list and inserting them into the domain.

We run the following sets of testcases for each IPC domain. We generated 100 domains with 2 errors for each mode specified above. For the mode **single**, we run the debugger for each problem instance. Since we use 10 problem instances per domain, the number of runs is 1000 instead of 100 for that mode. For each error type, we generate 100 domains with one error and test it in the **normal** mode. For all n between 1 and 15, generate 100 domains with n errors and test it in the **normal** mode.

We ensure that the generated faulty domain is valid in each testcase. The testcases are executed on a machine with an Intel i7-4770k CPU and 16GiB memory running on Ubuntu 18.04 and using OpenJDK17. In order to save time, the test cases are run in parallel on 6 threads.

4.3 Results

Recovery rates vary substantially across domains (Table 4), ranging from 52% for `tpp` to 96% for logistics, despite their similar sizes (Table 3). We observe no clear relationship between domain size and recovery rate. Simpler domains such as blocks and logistics achieve lower false positive rates fp_μ than more complex domains like pipesworld and rovers, likely due to their smaller space of possible fixes. Runtime (t_μ) also varies by domain, with `zenotravel` being the fastest and `tpp` the slowest, although most testcases finish within one second. The relatively large t_σ indicates that runtime depends largely on the structure of individual faulty domains.

■ **Table 4** Average recovery rate rec_μ , average false positive rate fp_μ , average runtime t_μ , and its standard deviation t_σ . The results are averaged over testcases with 2 errors, all error types in normal execution mode.

Domain	n	$rec_\mu[\%]$	$fp_\mu[\%]$	$t_\mu[ms]$	$t_\sigma[ms]$
blocks	100	76	12	34	38
freecell	100	73	85	266	56
logistics	100	98	2	30	16
driverlog	100	86	48	32	26
satellite	100	59	59	43	23
zenotravel	100	65	52	14	7.96
pipesworld	100	80	98	148	46
rovers	100	74	90	226	49
storage	100	83	77	18	14
tpp	100	55	81	71	29

Table 5 shows that recovery rates are higher for invalid than for valid faulty domains (80% vs. 60%). We assume errors are easier to identify when they cause a concrete plan failure. Notably, when the TIGHTEN algorithms are disabled and only VIOLATIONFIXER is used, the debugger still achieves a 28% recovery rate on valid domains. Since no precondition violations occur in this setting, these errors are detected solely through the SIAs, while maintaining a low false positive rate of 6%.

■ **Table 5** The results for all domains grouped by execution mode and faulty domain validity. The first row of each execution mode shows all corresponding testcases. The second and third rows shows only testcases where the faulty domain is valid or invalid. The results refer to testcases with 2 mixed errors. The table shows the average recovery rate rec_μ , average false positive rate fp_μ , the average runtime t_μ , and its standard deviation t_σ .

Mode	Valid?	n	$rec_\mu[\%]$	$fp_\mu[\%]$	$t_\mu[ms]$	$t_\sigma[ms]$
normal	$\top, \perp$	1000	75	60	88	93
	$\top$	263	60	71	90	91
	$\perp$	737	80	57	88	94
no tighten	$\top, \perp$	1000	53	9	21	26
	$\top$	263	28	6	15	13
	$\perp$	737	62	11	23	29
no sia	$\top, \perp$	1000	70	63	56	60
	$\top$	263	55	74	60	63
	$\perp$	737	75	59	54	60
with ignored	$\top, \perp$	1000	75	27	89	284
	$\top$	263	59	35	75	154
	$\perp$	737	80	24	94	317
single	$\top, \perp$	10000	64	66	20	54
	$\top$	2630	54	76	19	87
	$\perp$	7370	68	62	20	35

As shown in Table 5, providing all plans instead of only a single plan improves the recovery rate by 11 percentage points (75% vs. 64%). The effect is moderate because most problem instances in the test set already use all actions in the domain. In testcases where only the VIOLATIONFIXER is run and the TIGHTEN algorithms are skipped, the recovery rate is just 53%, but the false positives rate is only 9%. This is due to the fact that the VIOLATIONFIXER only makes changes when a violation is encountered, while the TIGHTEN algorithms constrict the domain as much as possible.

Regarding the usefulness of SIAs in general, rec_μ is 5 percentage points lower when no SIAs are given versus testcases with all SIAs, as shown in Table 5. fp_μ is also only slightly influenced, while the usage of SIAs increases the runtime of the testcases by 57%. As it turns out, blacklisting false positives obtained in the first run doesn't improve the recovery rate in the second round on average, as evident from the results in Table 5.

Table 6 compares recovery rates across error types and distinguishes between valid and invalid faulty domains. Added preconditions cannot be detected in valid domains (0% recovery), but are removed in 88% of cases when they cause violations. Missing preconditions always yield valid domains and are recovered in 84% of cases, while modified preconditions are repaired in 80% overall and 90% when violations occur. Added positive effects are repaired in 87% of cases. Missing positive effects are recovered in 89% of invalid domains but never when the domain remains valid. Likewise, modified positive effects are recovered much more often in invalid than valid domains (94% vs. 41%). Additional negative effects are the hardest to repair, with a recovery rate of 41%. However, recovery differs sharply between valid and

■ **Table 6** Results separated by error type. In each testcase, one error of the corresponding type is introduced into the domain. All testcases are run with normal execution mode. Average recovery rate rec_μ , average false positive rate fp_μ , the average runtime t_μ , and its standard deviation t_σ are shown. n reflects the number of distinct errors and thus faulty domains that could be generated for the corresponding error type.

Error	Valid?	n	$rec_\mu[\%]$	$fp_\mu[\%]$	$t_\mu[ms]$	$t_\sigma[ms]$
pre add	$\top, \perp$	510	78	87	112	86
	$\top$	56	0	100	100	89
	$\perp$	454	88	85	114	86
pre rem	$\top, \perp$	262	84	83	112	88
	$\top$	262	84	83	112	88
	$\perp$	0	0	0	0	0
pre rep	$\top, \perp$	357	80	86	116	84
	$\top$	53	22	96	93	90
	$\perp$	304	90	84	120	82
pos add	$\top, \perp$	297	87	89	192	576
	$\top$	297	87	89	192	576
	$\perp$	0	0	0	0	0
pos rem	$\top, \perp$	120	84	78	101	84
	$\top$	7	0	100	116	92
	$\perp$	113	89	77	100	84
pos rep	$\top, \perp$	55	89	77	137	88
	$\top$	6	41	85	137	84
	$\perp$	49	94	76	137	88
neg add	$\top, \perp$	331	41	89	152	186
	$\top$	186	3	92	142	81
	$\perp$	145	91	86	165	265
neg rem	$\top, \perp$	119	83	80	106	87
	$\top$	119	83	80	106	87
	$\perp$	0	0	0	0	0
neg rep	$\top, \perp$	69	48	82	140	90
	$\top$	62	42	86	148	87
	$\perp$	7	100	49	71	89
polarity	$\top, \perp$	239	70	76	151	645
	$\top$	126	84	76	193	882
	$\perp$	113	55	76	105	89

invalid domains (3% vs. 91%). Missing negative effects are recovered in 83% of cases, while modified negative effects are repaired in 42% of valid domains and all invalid ones. Effect polarity errors show the opposite trend, achieving a higher recovery rate in valid than in invalid domains (84% vs. 55%). Runtime t_μ is highest for additional precondition errors due to the exponential complexity of the minimal hitting set computation. The large t_σ is largely explained by the varying complexities of the evaluated domains.

The most surprising result is the limited impact of the number of errors on recovery rate (Table 7). While domains with a single error achieve a recovery rate of 76%, domains with up to seven errors still average 71%. Only beyond that point does performance decline substantially, reaching 35% for domains with 15 errors. As expected, the average runtime t_μ increases with the number of errors, whereas the false positive rate fp_μ decreases. These results suggest that the debugger can reliably reconstruct missing components as long as the domain's basic structure remains intact, but its effectiveness deteriorates once too many elements are missing.

■ **Table 7** The results for all domains grouped by the number of errors introduced in each testcase. The testcases are run under normal execution mode with mixed error types being picked. The table shows the average recovery rate rec_μ , average false positive rate fp_μ , the average runtime t_μ as well as its standard deviation t_σ . In the first row, n is less than 1000, because only 883 distinct errors can be generated using our strategy.

Error Count	n	$rec_\mu[\%]$	$fp_\mu[\%]$	$t_\mu[ms]$	$t_\sigma[ms]$
1	883	76	71	81	343
2	1000	75	60	88	93
3	1000	75	55	147	717
4	1000	74	52	231	910
5	1000	73	51	307	1059
6	1000	72	47	555	1552
7	1000	71	44	933	2148
8	1000	68	41	1355	2604
9	1000	64	38	1931	3116
10	1000	60	34	2497	3593
11	1000	55	29	3139	3888
12	1000	51	26	3754	4111
13	1000	43	22	4645	4321
14	1000	40	18	5099	4349
15	1000	35	15	5674	4353

5 Limitations

While the quantitative evaluation demonstrated that DrPDDL can reliably repair the considered error types, it also revealed some limitations. One limitation is scalability. Algorithms such as VIOLATIONFIXER have exponential runtime and may become computationally expensive for larger domains, even though this was not an issue in our evaluation. Another limitation is that DrPDDL reports only a single repair. Although users can exclude suggested fixes and rerun the debugger, alternative solutions produced by components such as the minimal hitting set algorithm are discarded, and the fixed sequence of algorithms further restricts the set of possible repairs. To illustrate these limitations, we now examine error types that were not considered in the implementation.

Removing the `pickup` action from the delivery robot domain causes a precondition violation for `(holding ?r ?p)` during `drop` evaluation. The debugger removes this precondition, but a subsequent SIA violation and failed goal achievement lead to contradictions, yielding no solution. In contrast, removing the fluent `empty` relaxes the domain. Since no preferred-plan violations occur, the debugger cannot detect the error. Missing fluents generally exhibit this behavior. Initial-state errors covered by SIAs can be detected. For example, adding `(isat r1 lb)` triggers an SIA violation. Errors not covered by SIAs, such as removing a traversed `(edge la lb)`, cannot be identified directly; instead, the debugger removes related preconditions, indicating the affected fluent. Impossible goals, incorrect SIAs, and errors in preferred plans all lead to contradictions that prevent a consistent repair. For instance, requiring a packet to be in two places simultaneously or removing a `(pickup r1 p2 le)` step causes violations that ultimately leave the problem unsolvable.

6 Conclusion and Future Work

The goal of this work was to develop an automated debugger for PDDL domains. Given a faulty domain, a set of preferred plans, and background knowledge in the form of State Integrity Axioms (SIAs), the debugger repairs the domain such that all plans remain valid and all SIAs are satisfied.

We first introduced the necessary planning concepts, formalized planning domains, classified common domain errors, and defined a restricted language for expressing SIAs. Based on this foundation, we designed a repair framework consisting of four algorithms: VIOLATIONFIXER, which resolves plan and SIA violations; TIGHTEN1, which removes redundant positive effects; TIGHTEN2, which adds valid negative effects; and TIGHTEN3, which adds maximal positive preconditions.

The implementation was evaluated on thousands of faulty instances generated from 10 IPC domains. Recovery rates were analyzed with respect to domain characteristics, error count, error type, available plans, and SIAs. The debugger repaired 75% of domains containing a single error and still achieved 69% recovery with 10 concurrent errors. Using multiple preferred plans improved recovery by about 10 percentage points, while SIAs provided a smaller but measurable benefit. Positive-effect errors were the easiest to repair, whereas negative-effect errors proved the most challenging.

Overall, the results demonstrate that preferred plans and SIAs can be effectively combined to automatically repair planning domains containing multiple concurrent errors.

We have laid the groundwork for an advanced PDDL domain debugger, leaving several avenues for future extension. Adding parameters expands the search space and requires updating preferred plans with new arguments. Supporting full propositional logic in SIA improves expressiveness but increases validation complexity. Incorporating features like quantifiers, fluents, and durative actions scales complexity beyond lifted STRIPS. Finding violations using only the domain, problem, and SIAs would eliminate the need for preferred plans.

References

- 1 J. Benton, Amanda Coles, and Andrew Coles. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling*, pages 2–10. AAAI Press, 2012.
- 2 Giuseppe De Giacomo and Toni Mancini. Scaling up reasoning about actions using relational database technology. In *Proceedings of the 19th National Conference on Artificial Intelligence*, pages 245–250. AAAI Press, 2004.
- 3 Jan Dolejsi. Vs code planning domain description language support, 2022. URL: <https://marketplace.visualstudio.com/items?itemName=jan-dolejsi.pddl>.
- 4 Thomas Eckstein, Marco De Bortoli, and Gerald Steinbauer-Wagner. Fixing pddl domains using delta debugging. In *34th International Workshop on Principles of Diagnosis*. AAAI Press, 2023.
- 5 Malik Ghallab, Adele Howe, Craig Knockblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl the planning domain definition language. Technical report, Yale Center for Computational Vision and Control, 1998.
- 6 Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning. Theory & practice*. Elsevier, 2004.
- 7 Alba Gragera, Raquel Fuentetaja, Angel Garcia-Olaya, and Fernando Fernandez. Pddl domain repair: Fixing domains with incomplete action effects. In *International Conference on Automated Planning and Scheduling*. AAAI Press, 2023.
- 8 Alba Gragera, Raquel Fuentetaja, Angel Garcia-Olaya, and Fernando Fernandez. A planning approach to repair domains with incomplete action effects. In *International Conference on Automated Planning and Scheduling*. AAAI Press, 2023.
- 9 Richard Howey and Derek Long. Val’s progress: the automatic validation tool for pddl2.1 used in the international planning competition. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling*, pages 28–37. AAAI Press, 2003.

- 10 Songtuan Lin, Alban Grastien, and Pascal Bercher. Planning domain repair as a diagnosis problem. *DX 2022*, 2022.
- 11 Songtuan Lin, Alban Grastien, and Pascal Bercher. Towards automated modeling assistance: An efficient approach for repairing flawed planning domains. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37:12022–12031, 2023. doi:10.1609/AAAI.V37I10.26418.
- 12 Patrick Lühne. pddl-instances, 2017. URL: <https://github.com/potassco/pddl-instances>.
- 13 Erik Mueller. *Commonsense Reasoning - An Event Calculus Based Approach*. Elsevier, 2014.
- 14 Christian Muise. planutils, 2020. URL: <https://github.com/AI-Planning/planutils>.
- 15 Tomas Plch, Miroslav Chomut, Cyril Brom, and Roman Barták. Inspect, edit and debug pddl documents: Simply and efficiently with pddl studio. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling*. AAAI Press, 2012.
- 16 Raymond Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64(2):337–351, 1993. doi:10.1016/0004-3702(93)90109-0.
- 17 Kayleigh Sleath and Pascal Bercher. Detecting ai planning modelling mistakes - potential errors and benchmark domains. In *Proceedings of the 20th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2023)*. Springer, 2023. doi:10.1007/978-981-99-7022-3_41.

QSIM-Guided Weak Supervision for Fault Detection in Cyber-Physical Systems

Ankita Das¹ 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Roxane Koitz-Hristov 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Franz Wotawa² 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Abstract

This paper presents an initial investigation into the combination of qualitative simulation and machine learning for fault detection in cyber-physical systems. In recent years, the detection and diagnosis of faults in cyber-physical systems has become increasingly dependent on machine learning. However, supervised machine learning approaches rely on large labelled datasets, which are rarely available during the initial deployment phase. In systems such as smart buildings, faults occur rarely, and even the nominal behaviour of a system can vary across structurally identical installations. We therefore propose a framework that addresses this issue by generating weak supervision labels using qualitative reasoning. Given a qualitative model that captures the system's nominal behaviour, we use a conformance check to determine whether the system traces are likely to be faulty or not. These binary pseudo-labels are then used to train a downstream classifier. Unlike statistical anomaly detection or self-supervised techniques, the generated supervision signal is based on physical consistency rather than deviations from a learned baseline. We evaluate our approach on four benchmarks relating to electrical, fluid, mechanical and thermal dynamics, comparing our methodology with unsupervised anomaly detection and supervised learning with limited labels. Our initial experiments suggest that, while qualitative conformance checks can provide usable training signals when labelled data is unavailable, their effectiveness depends on the discriminative structure of the qualitative model.

2012 ACM Subject Classification Computing methodologies → Causal reasoning and diagnostics; Computing methodologies → Spatial and physical reasoning; Computing methodologies → Modeling methodologies

Keywords and phrases Qualitative Simulation, Fault Detection, Outlier Detection

Digital Object Identifier 10.4230/OASICS.DX.2026.11

Funding The work presented in this paper has been supported by the FFG project Artificial Intelligence for Smart Diagnosis in Building Automation (ALFA) under grant 914932.

Ankita Das: Supported by the Christian-Doppler Forschungsgesellschaft (CDG) project AI-based Diagnosis for Energy Transition and the Circular Economy (AID4ETCE).

Acknowledgements The following AI-based tools were used for language editing and polishing: ChatGPT 5.5 and Claude 4.7.

1 Introduction

Cyber-physical systems (CPS) tightly couple physical processes and computing components through the use of sensors, actuators, and feedback loops to enable smart monitoring and control in buildings, industrial automation, or transportation. Fault detection and diagnosis

¹ Authors are listed in alphabetical order.

² Corresponding author.



© Ankita Das, Roxane Koitz-Hristov, and Franz Wotawa;
licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 11; pp. 11:1–11:20

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

(FDD) of CPS is becoming increasingly data-driven with the goal of improving the operational efficiency, reliability, and safety of CPS [15]. Machine learning (ML) and deep learning approaches have shown promising results for anomaly detection [27, 21] and predictive maintenance [26]. However, these methods typically require large operational datasets and reliable fault annotations to sufficiently capture the system behavior and generalize across operating conditions.

In practical CPS settings, this requirement is often difficult to satisfy. In many CPS domains, fault data is rare, unavailable at the beginning of the system’s life cycle, or expensive to acquire [7]. Let us consider smart buildings, a domain that suffers strongly from the cold-start problem, i.e., the absence of sufficient representative training data during early deployment phases. Newly commissioned buildings do not provide an extensive amount of operation data and faults occur rarely. This leads to little to no training data being available. Moreover, even nominal behavior may vary substantially across structurally identical systems due to installation specific-parameters, environmental conditions, or occupant behavior. Thus, ML models trained on one building generalize poorly to others [20, 31].

Several research areas address this issue by reducing the data and annotation requirements of learning-based diagnosis, e.g., transfer learning [13, 10], self-supervised learning [14, 32], and anomaly detection [4]. Yet, these approaches typically depend on sufficiently representative operational data, assume the availability of an initial fault-free operational phase, or infer abnormal behavior from deviations from a learned statistical baseline rather than from violations of physical consistency. Weak supervision is a complementary line of research that addresses label scarcity by replacing expert-provided annotations with programmatic labeling functions encoding heuristics, rules, or external knowledge sources [24, 33]. The quality of weak supervision, however, depends on the quality of these labeling sources. For CPS, the most reliable source is the system’s: a fault is, by definition, a behavior that the underlying physical model cannot produce when it functions correctly. Thus, symbolic reasoning from first principle, e.g., realized through qualitative reasoning, presents itself as a natural supervision source for diagnosis.

Qualitative reasoning provides a way to formalize physically plausible system behavior and structure without the need for precise numeric values [8, 11, 18, 28]. In contrast to quantitative approaches, such as digital twins or physics-informed neural networks, qualitative models describe the system behavior through qualitative variables that encompass qualitative magnitudes (e.g., positive, negative, or zero) and qualitative directions of change (e.g., increasing, decreasing, or steady) [11]. In our previous work [5, 6], we have shown that qualitative conformance checking, i.e., comparing observed traces against the set of behaviors permitted by a qualitative simulation model, can detect faults in CPS without requiring precise parameterization.

While qualitative conformance checking provides interpretable and physically grounded diagnosis results, purely symbolic reasoning approaches may become computationally expensive for continuous online monitoring and large-scale deployment. In contrast, ML-based classifiers provide efficient runtime inference and scalable deployment, but face the aforementioned cold-start problem. This motivates us to combine both paradigms by using qualitative reasoning as a symbolic supervision mechanism for training ML fault detectors.

In this paper, we propose a framework that exploits qualitative conformance outcomes between a qualitative model and observed CPS traces to generate weak supervision labels for fault detection. In essence our approach can be divided into two stages. First, we exploit a QSIM model of the nominal system behavior to detect deviations in system observation traces by checking the trace’s conformance with the admissible system behavior generated

by the qualitative simulation. In case the system behavior does not conform to any of the qualitative simulation traces, we label the trace as *non-conforming*, i.e., faulty; otherwise the trace is labeled *conforming*, i.e., nominal. Second, we use those pseudo-labels to train a binary ML classifier. To determine the feasibility and effectiveness of our approach, we evaluate our framework on multiple benchmarks and a real-world system under cold-start conditions and compare our method against off-the-shelf anomaly detection methods.

The remainder of this paper is structured as follows. In Section 2, we discuss related work. Subsequently, we give background information on qualitative simulation and our conformance checking approach in Section 3. Section 4 describes how we exploit qualitative simulation and conformance checking to obtain weak labels, while Section 5 and Section 6 present our initial empirical study and its results, respectively. Lastly, we conclude the paper and provide directions for future work in Section 7.

2 Related Work

A substantial body of research addresses FDD in CPS in low-label environments. For instance, transfer learning approaches reuse models across installations or operating conditions to compensate for limited data availability. Genkin and McArthur [13] combine transfer learning with reinforcement learning to transfer optimal control policies from an existing building to a newly commissioned structure. While reducing the warm-up period and overall mean variance, their approach requires a transfer building with a wide range of sensors to work effectively. Dou et al. [10] exploit simulation data from a source building to pre-train neural network models that are then fine-tuned on the measurements of the target installation with promising results.

Self-supervised learning (SSL) is another widely used strategy for low-label environments. SSL exploits unlabeled data and a pretext task to learn structure and relationships in the data to generate supervisory signals without the need for human annotations [14]. Senanayaka et al. [25] combine one-class support vector machine (SVM) for health estimation with supervised convolutional neural network (CNN) classification for online fault diagnosis in electric powertrains. First, the one-class SVM is trained on available nominal data to define the machine health status; second, a CNN classifier is trained on those health classes. Although explicit labeling is no longer a prerequisite, such methods still require a sufficiently representative operational data and may not be able to generalize across heterogeneous system configurations [32].

Anomaly and outlier detection methods, such as one-class SVMs, are also commonly used when labeled fault data is unavailable. Those methods train exclusively on nominal system behavior, avoiding the need for fault data altogether. While requiring less data than traditional supervised learning methods, these methods assume an initial fault-free start-up phase and rely on statistical methods rather than physical inconsistencies to determine abnormal behavior [4].

A complementary line of research addresses low-label environments by changing the source of supervision rather than the data requirements. Weak supervision is one such technique that reduces dependence on human-provided annotations through programmatic labeling functions that encode heuristics, rules, or other external knowledge sources to produce training labels [33, 24]. Frameworks, such as Snorkel [24], demonstrate that programmatically generated labels can be used successfully for training models in domains where manual annotations are infeasible or expensive. Snorkel combines multiple labeling functions through a generative model to produce probabilistic training labels at scale. Weakly supervised learning has increasingly been explored in predictive maintenance and CPS settings to address incomplete, inexact, or inaccurate supervision [22].

In contrast to existing weak supervision and SSL approaches, our work derives supervisory signals not from statistical anomalies but from symbolic physical reasoning. Building on our previous work on qualitative simulation and conformance checking [5, 6], we use qualitative conformance outcomes as weak labels for downstream supervised learning. Similar to prior pseudo-labeling approaches, as the one presented by Senanayaka et al. [25], our framework employs an intermediate diagnosis mechanism to generate labels for a supervised learning classifier. Our proposed approach generates labels from violations of physically admissible system behavior encoded in a qualitative model.

3 Qualitative Simulation (QSIM)

Qualitative Simulation (QSIM) provides an abstract representation of dynamical system behavior by replacing precise numerical values with symbolic categories and qualitative relations [11, 9, 19, 18]. Instead of modeling variables through exact measurements, QSIM represents them using qualitative values such as *low*, *medium*, *high*, *on*, and *off*, together with constraints that capture structural and physical dependencies between variables. This abstraction is particularly useful for CPS, where the qualitative structure of the system is often known even when exact parameter values or detailed numerical models vary.

A key advantage of qualitative simulation is that it can describe families of behaviors rather than a single numerical trajectory. For example, changes in resistance, capacitance, inflow rate, damping, or thermal resistance may alter the exact numerical trajectory of a system, while preserving its qualitative structure. QSIM exploits this property by reasoning over qualitative magnitudes, landmarks, intervals, and qualitative trends. As a result, the same qualitative model can often be reused across different parameterizations of a benchmark, provided that the qualitative relationships between variables remain unchanged.

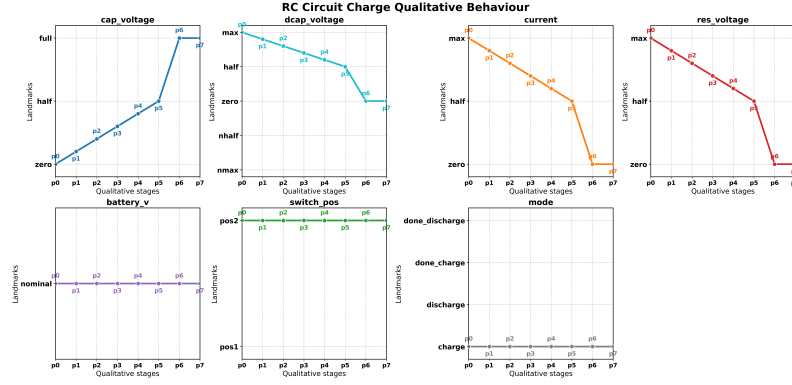
In this work, we use ASP-QSIM, an Answer Set Programming (ASP) formulation of qualitative simulation [30]. In ASP-QSIM, system states, transitions, and qualitative constraints are encoded declaratively as logic predicates. Solving the resulting ASP program yields one or more answer sets, where each answer set corresponds to one admissible qualitative evolution of the modeled system over time.

A qualitative *trace* denotes one such admissible system evolution. It consists of:

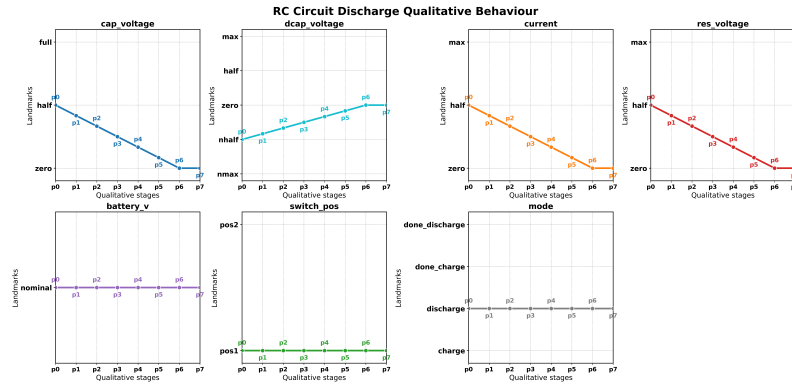
- *time points*, such as $p(0), p(1), \dots$, representing qualitative system states,
- *intervals*, such as $i(0, 1)$, representing transitions between successive states, and
- *qualitative assignments* of system variables at time points and intervals, for example $v_{\text{heating_request}}(p(0)) = \text{on}$ or $v_{\text{supply_temp}}(p(1)) = \text{high}$.

Within the ASP-QSIM encoding, predicates such as `holds/4` represent qualitative variable values and trends at particular time points or intervals, while predicates such as `holds_constraint/4` encode qualitative relations that must hold between variables. These relations may express structural dependencies, monotonic influences, qualitative proportionalities, or admissible state transitions. Hence, a trace captures not only the qualitative state of the system, but also how this state evolves over time.

Since qualitative simulation generally admits multiple valid evolutions, the model defines a set of possible qualitative behaviors rather than a single deterministic trajectory [19, 18]. This is important for our setting because the exact numerical trajectory of a cyber-physical system may vary across parameter settings, initial conditions, and operating regimes. Instead of requiring one exact numerical prediction, the qualitative model specifies which classes of behavior are physically admissible. However, while local qualitative simulation algorithms,



(a) RC circuit charging behavior.



(b) RC circuit discharging behavior.

Figure 1 Example qualitative behaviors generated by the ASP-QSIM model for the RC circuit benchmark. The charging and discharging cases illustrate admissible qualitative evolutions of the nominal system, which are used as reference behaviors for downstream conformance checking.

such as QSIM, are able to produce every physically possible behavior, they may also generate impossible spurious traces. This follows from the inherently local nature of state expansion: each successor is derived from the information in the current state alone [17].

Figure 1 illustrates this idea for a simple RC circuit. The charging and discharging regimes are represented by separate ASP-QSIM reference behaviors. In the charging case, the capacitor voltage increases toward the supply voltage, while in the discharging case the capacitor voltage decreases. These figures illustrate that the QSIM model captures qualitative evolution patterns rather than exact numerical curves. Such admissible qualitative behaviors form the reference set used for conformance checking.

This property is central to our framework. We use the set of admissible qualitative traces generated by the nominal ASP-QSIM model as a reference model of expected system behavior. Observed CPS traces are first abstracted into the same qualitative representation using benchmark-specific landmarks and intervals. The abstracted observations are then checked for conformance against the admissible behaviors of the nominal model. Intuitively, an observed trace is conforming if there exists at least one admissible ASP-QSIM behavior that is consistent with the observed qualitative assignments. If no such behavior exists, the observation is considered non-conforming (see Das et al. [5]).

Operationally, conformance checking is performed by adding the observed qualitative assignments as constraints to the ASP-QSIM program. If the resulting program has at least one answer set, then the observation is compatible with the nominal qualitative model. If the program becomes unsatisfiable, then the observed behavior cannot be explained by the nominal model and is treated as a potential fault. Thus, ASP-QSIM provides a model-based mechanism for converting qualitative physical consistency into binary conformance outcomes.

In the proposed weak-supervision framework, these conformance outcomes are used as pseudo-labels for a downstream ML task. The next section describes our approach.

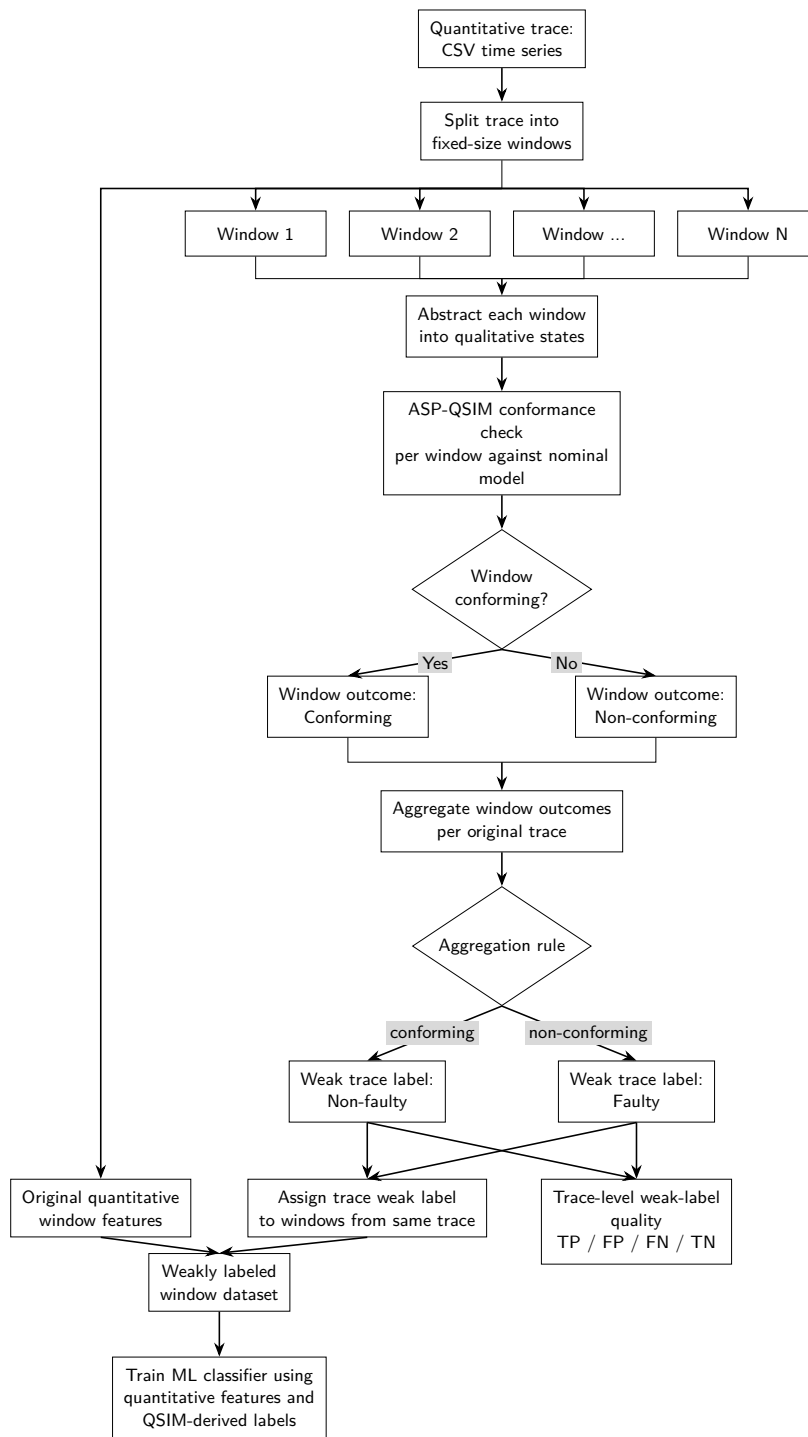
4 Approach

We address the cold-start problem by combining a qualitative model of nominal system behavior with conformance checking [1, 23], using qualitative simulation as a source of weak supervision for ML. The central assumption is that, in many CPS, expert knowledge about nominal behavior is available before large amounts of labeled fault data can be collected. We encode this knowledge as a nominal QSIM model [19]. Observed quantitative traces are then abstracted and checked for conformance against the admissible behaviors generated by this model. The resulting conformance outcomes are interpreted as pseudo-labels and used to train downstream classifiers.

Figure 2 summarizes the weak-label generation pipeline. Each quantitative time-series trace, obtained either from simulation or system operation, is divided into fixed-length temporal windows. These windows define the quantitative feature vectors used by the ML models. In parallel, each window is abstracted into qualitative states using the benchmark-specific landmarks and intervals. The abstracted window is then checked for conformance against the nominal ASP-QSIM model. If the window is consistent with at least one admissible qualitative behavior, it is considered conforming; otherwise, it is considered non-conforming.

The window-level conformance outcomes are aggregated per original trace to obtain a trace-level pseudo-label. This step is necessary because the benchmark ground-truth labels are defined at the trace level, while the ML instances are defined at the window level. The resulting QSIM-derived pseudo-label is assigned to the quantitative windows from the same trace, producing a weakly labeled dataset for classifier training. Thus, the classifier learns from concrete quantitative features, while the supervision signal is derived from qualitative physical consistency.

This design has three main advantages. First, it reduces the need for manually labeled fault data, which is particularly important in cold-start settings. Second, windowing increases the number of training instances compared with labeling only complete traces. Third, the learned classifier can capture numerical regularities in the concrete system behavior that are not explicitly represented in the qualitative abstraction. The trade-off is that window-level conformance provides less temporal context than complete-trace conformance and can therefore be sensitive to abstraction thresholds, window boundaries, and initial-state assumptions. We therefore treat the resulting labels as weak labels rather than ground truth. The binary weak labels that result from these compliance results are then used to train ML models downstream. The framework eliminates the need for laborious human labeling by transforming model-based behavioral consistency into a learning signal. In order to determine whether qualitative conformance can facilitate efficient fault detection under sparse annotation, the final models are assessed on held-out traces and in low-label circumstances.



■ **Figure 2** Window-level QSIM weak-label generation pipeline. Quantitative traces are segmented into windows, abstracted into qualitative states, and checked for conformance against the nominal ASP-QSIM model. Window-level conformance outcomes are aggregated into trace-level pseudo-labels, which are assigned to quantitative window features for downstream ML training and trace-level evaluation.

5 Evaluation

In this section, we outline our initial experiments. First, we describe the four benchmark systems, before detailing the experimental setup.

5.1 Benchmark Description

We examine four benchmarks that vary in terms of both system complexity and the kind of qualitative dynamics they display in order to assess the suggested framework. From a qualitative perspective, the benchmarks cover distinct classes of system dynamics. Together, these benchmarks allow us to study the proposed approach on both compact illustrative systems and a practically relevant control application.

5.1.1 RC circuit

The RC circuit, as shown in Figure 3a, serves as a compact benchmark for transient qualitative behavior. Its dynamics are governed by the interaction between switch position, voltage source, capacitor voltage, current, resistor voltage, and the derivative of the capacitor voltage. Qualitatively, the benchmark exhibits structured charging and discharging processes with clear landmark transitions and trend changes. This makes it particularly suitable for analyzing how nominal and faulty behaviors can be distinguished through qualitative traces and how such distinctions can be converted into weak labels.

5.1.2 Bathtub

The bathtub benchmark provides a simple but expressive example of accumulation-driven behavior [19, 11]. As can be seen in Figure 3b, the system consists of a reservoir with controlled inflow and gravity-driven outflow, such that the water level evolves according to the balance between incoming and outgoing flow. In qualitative terms, this benchmark captures filling, draining, and equilibrium behavior, making it well suited for examining how deviations in flow conditions affect trace evolution and conformance. For this benchmark, we used the public ASP-QSIM implementation of Wiley et al. [30] as a starting point and adapted it to our experimental setting.³

5.1.3 Spring–mass–damper

The spring–mass–damper benchmark represents a classical mechanical system with oscillatory behavior and our ASP-QSIM implementation follows the qualitative formulation introduced by Coghill et al. [3]. It consists of a mass connected to a fixed support through a spring and a viscous damper operating in parallel, and is subject to an external force (see Figure 3c). The resulting dynamics are characterized by restoring forces, dissipation, and damped oscillations. From a qualitative perspective, this benchmark is useful because it introduces non-monotonic behavior and repeated changes in trend, which are more challenging than purely charging or accumulation-based dynamics. It therefore provides a complementary test case for assessing whether conformance-based weak supervision can handle mechanically induced temporal patterns.

³ <https://github.com/timothy-wiley/aspqsim>

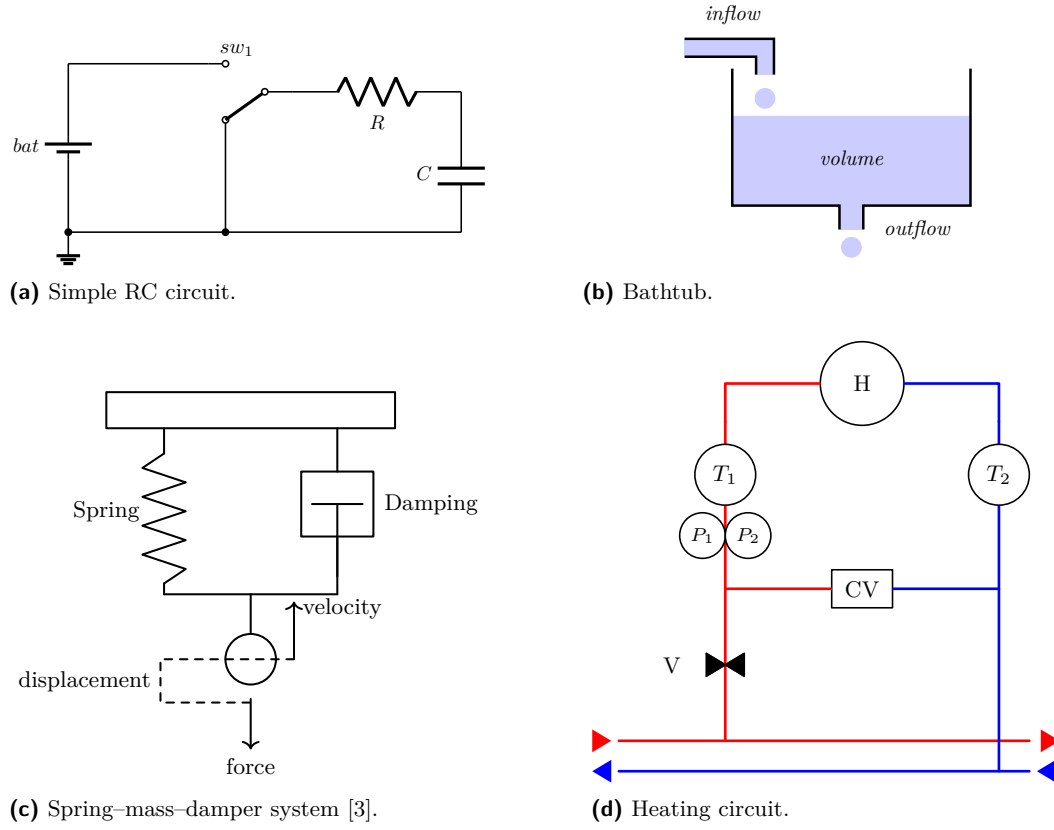


Figure 3 The four benchmark systems used in this study: (a) an RC circuit with resistor R , capacitor C , switch sw_1 , and battery bat . (b) a bathtub with inflow, outflow, and volume; (c) a spring-mass-damper with displacement, velocity, and external force; (d) a heating circuit with heat exchanger H , two alternating pumps P_1 and P_2 , a globe valve V , a check valve CV , and temperature sensors T_1 and T_2 .

5.1.4 Heating circuit

The heating circuit, shown in Figure 3d [16], constitutes the most application-oriented benchmark in our study as it is the only system where real-world traces are available to us. It models a hydraulic heating loop that supplies heat from a district heating station to a commercial building. District heating systems are an important component of sustainable urban energy infrastructure [29], and faults in such systems can lead to substantial energy losses [2].

For this benchmark, we use a simplified QSIM model to evaluate whether compact qualitative knowledge can provide useful weak labels on real data. The model covers active heating operation under two pump regimes, *pump1_only* and *pump2_only*, with one regime-specific nominal model for each configuration. In the first regime, pump P_1 is active and pump P_2 is inactive; in the second regime, pump P_2 is active and pump P_1 is inactive. A stable-regime constraint prevents unintended switching between pump regimes during conformance checking.

The qualitative variables considered in the simplified heating model include outdoor temperature, heating request, pump states, heat-exchanger flow, valve position, supply temperature and return temperature. Outdoor temperature and valve position are treated as

contextual variables to tolerate delays and feedback effects in the real data. Thermal behavior is captured coarsely through a monotonic relation between supply and return temperature during active circulation. The model is therefore intentionally conservative: it abstracts away the full low-level controller and tests whether compact nominal knowledge can still provide useful weak supervision for fault detection.

5.2 Experimental Setup

For each benchmark except the heating system, we generated simulated traces under both nominal and faulty conditions using Modelica models [12]. In order to assess our approach on various system configurations, we used parameter sweeps over operating regimes and initial conditions. Table 1 gives an overview of the parameter ranges considered. Ground-truth labels were obtained directly from the simulation setup: traces generated under nominal conditions were treated as *non-faulty* whereas traces generated under injected fault conditions were treated as *faulty*.

■ **Table 1** Benchmark parameter ranges and conformance results between the qualitative models and the corresponding nominal quantitative simulation behaviors.

Benchmark	Parameter	Range
RC Circuit	R resistance	{9, 10, 11}
	C capacitance	{0.0008, 0.001, 0.0012}
	V_{bat} voltage	{4.0, 4.5, 5.0}
Bathtub	q_{in} inflow	{0.04, 0.08, 0.12}
	A_d drain area	{0.002, 0.005, 0.010}
	$V(0)$ initial volume	{0.10, 0.20, 0.40}
Spring–Mass–Damper	k spring constant	{0.5, 1.0, 2.0}
	c damping coefficient	{0.3, 1.0, 3.0}
	$x(0)$ initial position	{−0.5, 0, 0.5}
	$\dot{x}(0)$ initial velocity	{−0.5, 0, 0.5}

Across the simulated benchmarks, faults were injected as separate Modelica model variants targeting the main physical components of each system. In the RC circuit, faults include resistor short and open circuits, capacitor faults, switch faults, intermittent contacts and battery-supply faults, which affect current flow and voltage dynamics. In the bathtub benchmark, faults modify the water-balance behavior through stuck or excessive inflow, drain blockage, leakage, altered tub geometry, overflow-threshold changes, and intermittent blockage. In the Spring–Mass–Damper benchmark, faults affect actuator force, spring stiffness, damping, mass and friction thereby changing the oscillatory dynamics. For the real heating circuit, the faulty traces correspond to two valve faults: a leaking valve and a stuck valve. Table 2 summarizes these fault families.

Across all benchmarks, quantitative system variables are abstracted into qualitative levels and linked through qualitative constraints. Derivatives are included when they are part of the benchmark model or can be robustly estimated from the trace; otherwise, the abstraction relies on qualitative magnitudes only. The nominal QSIM model defines the admissible qualitative behavior of the system. Observed nominal and faulty simulation traces are then abstracted and checked against this nominal behavior model to derive conformance outcomes for weak-label generation.

■ **Table 2** Summary of fault types used in each benchmark.

Benchmark	#Faults	Fault types
RC circuit	10	Resistor, capacitor, battery and switch faults including short, open, degraded and intermittent behaviors.
Bathtub	10	Inflow, drain, overflow, tub area error and intermittent blockage faults.
Spring–Mass–Damper	10	Actuator, spring, damper, mass and friction faults.
Heating circuit	2	Real valve faults: leaking valve and stuck valve.

To obtain ML instances, each continuous trace was segmented into fixed-length windows. Unless otherwise stated, we used windows of 100 consecutive samples, e.g., samples 0–99, 100–199, and so on. These windows were used to construct the quantitative feature vectors for downstream learning. To ensure comparability across methods, the amount of raw operational data, trace duration, and number of windows were kept as consistent as possible across the anomaly detection, weak-supervision, and low-label supervision settings.

Weak labels were generated using window-level qualitative conformance checking. Each window was abstracted into qualitative states and checked against the corresponding nominal QSIM model. The resulting window-level conformance outcomes were aggregated per original trace to obtain QSIM-derived pseudo-labels. These pseudo-labels were then assigned to the quantitative windows from the same trace and used as binary labels for our weakly supervised training regime.

For the simulated benchmarks with parameter ranges, windows from all parameter configurations were pooled after applying a temporal split within each trace. The first 70% of windows from each trace were used for training, and the remaining 30% were used for testing. As a result, both portions contain examples from all simulated parameter configurations. Thus, both splits contain examples from all simulated configurations, allowing us to evaluate performance when the operating range is represented during training.

We considered five supervision regimes:

- **Full supervision**, where classifiers were trained using ground-truth labels on the first 70% of windows from each trace, including both nominal and faulty traces. The remaining 30% of windows from each trace were used for testing. As applied logistic regression, random forests and decision trees for supervised learning.
- **Weak supervision**, where classifiers were trained on quantitative window features using QSIM-derived pseudo-labels instead of manually provided labels. The same temporal split as in full supervision was used and performance was evaluated against the ground-truth labels.
- **Low-label supervision**, where classifiers were trained using only a small subset of ground-truth labels from the training split. This setup represents a cold-start scenario in which only a few manually labeled examples are available. The selected labeled windows were balanced at the binary class level between *non-faulty* and *faulty* examples. Thus, LL-5 used 3 non-faulty and 2 faulty labeled windows, LL-10 used 5 non-faulty and 5 faulty labeled windows, and LL-20 used 10 non-faulty and 10 faulty labeled windows.
- **Unsupervised anomaly detection**, where models were trained only on nominal windows. This setting reflects the assumption that some fault-free operation may be available during system start-up or commissioning, while fault labels are unavailable. We trained One-Class SVM and Isolation Forest on the first 70% of nominal windows and tested them on the remaining nominal windows together with faulty windows.

- **Full Supervision across configurations**, where models are trained on one configuration, i.e., parameter setting, and evaluated on held-out configurations, emulating the transferal of a model learned on one instance to another instance with varying parameters. As with the supervised case we use the ground truth, however we train on 100% of the samples of the source configuration and test on 100% of the other configurations.

To assess the quality of the weak-label signal itself, we additionally compare QSIM-derived labels against the simulation ground truth using precision, i.e., $\text{Precision} = \frac{TP}{TP+FP}$, recall, i.e., $\text{Recall} = \frac{TP}{TP+FN}$, F1 score, i.e., $F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$, and the underlying classification statistics (i.e., true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN)).

We report accuracy, i.e., $\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$, and Macro-F1 as the main predictive metrics for downstream classification. Macro-F1 is the unweighted average of the classwise F1-scores, so that faulty and non-faulty classes contribute equally even under class imbalance, i.e., $\text{Macro-F1} = \frac{F_1+F_2+\dots+F_N}{N}$, where F_1 is the F1 score for class 1 etc. and N is the total number of classes.

All benchmarks use the same general weak-supervision idea, with benchmark-specific modifications where necessary. Bathtub and Spring–Mass–Damper use the generic window-level conformance pipeline. For the RC circuit, we additionally use phase-aware aggregation and trace-level representations because RC traces contain structured charging and discharging phases separated by switching transitions. These extensions allow the weak-labeling procedure to account for stable phases, transition windows and settling behavior more explicitly than a plain fixed-window approach.

For the heating system benchmark, the raw operational CSV traces were first transformed into the same window-level format used for the other benchmarks. Since the real data contain multiple operating regimes, each window was assigned to a pump regime based on the dominant pump state within that window. Windows dominated by `pump1_only` behavior were checked against the pump1 nominal model, while windows dominated by `pump2_only` behavior were checked against the pump2 nominal model. Mixed or inactive windows were treated conservatively as non-conforming, since they do not match either active nominal regime model.

6 Results

In this section, we analyze the results of our initial study. First, we discuss the accuracy of our labeling approach, before elaborating on the classification results of the various supervision regimes.

In particular, we first explain the results on the fully supervised, weakly supervised and low-label approaches, before elaborating on the results of the unsupervised learning models. In addition, we report on cross-configuration results in the full supervision case.

6.1 Weak Labeling Accuracy

Table 3 reports the quality of the QSIM-derived weak labels before any downstream classifier is trained. Faulty behavior is treated as the positive class. Thus, TP are faulty examples correctly assigned faulty/non-conforming weak labels by QSIM, TN are nominal examples correctly assigned non-faulty/conforming weak labels, FP are nominal examples incorrectly rejected by the QSIM-based weak-labeling procedure and FN are faulty examples incorrectly

■ **Table 3** Quality of QSIM weak labels per benchmark, evaluated against ground-truth trace labels. TP/FP/FN/TN counts treat “Faulty” as the positive class. Precision (Prec.), recall (Rec.), and F1-score characterize the weak-label signal before any classifier is trained.

Benchmark	Ground truth		Counts				Quality		
	Faulty	Non-Faulty	TP	FP	FN	TN	Prec.	Rec.	F1
RC circuit	270	27	243	0	27	27	1.000	0.900	0.947
Bathtub	270	27	198	21	72	6	0.904	0.733	0.810
Spring–mass–damper	810	81	635	57	175	24	0.918	0.784	0.846
Heating	920	805	198	53	722	752	0.789	0.215	0.338

TP: QSIM = Faulty, Ground truth = Faulty; FP: QSIM = Faulty, Ground truth = Not faulty; FN: QSIM = Not faulty, Ground truth = Faulty; TN: QSIM = Not faulty, Ground truth = Not faulty.

assigned conforming weak labels. Precision measures how reliable a QSIM-generated fault label is, whereas recall measures how many ground-truth faulty examples are recovered by the conformance check.

For the RC circuit, the weak labels are highly reliable. The absence of false positives ($FP = 0$) shows that nominal traces are consistently accepted by the qualitative model, while the high number of true positives ($TP = 243$) indicates that most injected faults produce qualitative deviations. The remaining errors are false negatives ($FN = 27$), meaning that some faulty traces are not rejected by the QSIM-based labeling procedure. This is consistent with the fault-level analysis: most capacitor, resistor, supply, and switch faults are detected, whereas the fault representing a loose connection or failing contact at the resistor, remains often undetected.

The bathtub benchmark shows a noisier but still useful weak-label signal. QSIM detects many faulty traces ($TP = 198$), resulting in good precision (0.904), but it also misses a non-negligible number of faults ($FN = 72$). The false positives ($FP = 21$) indicate that some nominal traces are over-rejected by the operational labeling pipeline. This does not necessarily mean that the qualitative model fails to describe the intended nominal behavior. Faults that strongly alter the qualitative structure, such as the fault where the effective area is scaled, are detected well, while several inflow-related faults like `InflowStuckClosed` and `InflowStuckOpenHigh` remain closer to nominal behavior and are therefore more frequently missed.

For the Spring–Mass–Damper benchmark, the weak labels also provide a useful signal, with high precision (0.918) and reasonably high recall (0.784). However, the model still produces both false positives and false negatives. The false positives indicate occasional over-rejection of nominal traces while the false negatives show that some faulty traces remain difficult to distinguish from nominal behavior under the current qualitative abstraction. This behavior is expected for faults that mainly affect quantitative magnitudes without producing a clear qualitative change in the observed variables. In the fault-level analysis, actuator-related faults and several structural faults are captured well, whereas an increased spring stiffness and some friction-related cases are more difficult.

The most challenging example is the heating benchmark. It is based on actual operational data and includes controller delays, operational variability, and regime shifts that the simplified qualitative model is unable to completely reflect, in contrast to the simulated benchmarks. The heating model treats mixed or inactive windows conservatively as non-conforming and uses distinct locked nominal models for the `pump1_only` and `pump2_only` regimes. The weak-label statistics reveal low recall (0.215) and moderate precision (0.789),

with many faulty windows obtaining conforming labels ($FN = 722$). This occurs because the valve faults do not always produce an immediate qualitative change in the observed pump state, flow, heating request or temperature variables. According to these findings, under the existing abstraction leaking-valve and stuck-valve faults frequently remain qualitatively close to nominal active heating behavior. This is plausible because the observed variables mainly capture coarse pump-regime, flow, request, and temperature states. Valve faults may change the quantitative efficiency or timing of heat transfer without necessarily changing these variables enough to cross qualitative landmarks within a window. Consequently, leaking-valve and stuck-valve faults can remain qualitatively close to nominal active heating behavior under the current abstraction.

Overall, the results show that QSIM weak labels are useful but imperfect supervision signals. They are strongest when faults cause clear qualitative deviations from nominal behavior, as in the RC circuit and weaker when faults primarily induce subtle quantitative changes, as in the heating benchmark. This motivates the combination with ML: QSIM provides physically grounded labels without manual annotation, while the downstream classifier learns from the original quantitative window features and can capture numerical regularities not explicitly represented in the qualitative abstraction.

6.2 Fault Detection Accuracy

Table 4 summarizes the predictive performance across the four benchmarks under full supervision, QSIM-based weak supervision, and low-label supervision. We report both accuracy and Macro-F1, and focus primarily on Macro-F1 in the discussion because it gives equal weight to faulty and non-faulty classes and is therefore more informative under class imbalance.

Overall, the results show that QSIM-derived weak labels can provide a useful supervisory signal when manually labeled data are scarce although the strength of this signal varies across systems. The clearest case is the RC circuit benchmark. Here, weak supervision retains a substantial fraction of the fully supervised performance across all three classifiers especially for logistic regression, where weak QSIM supervision slightly exceeds the fully supervised baseline. This reflects the strong alignment between the qualitative RC model and the simulated quantitative behavior: most injected faults produce clear qualitative deviations and nominal behavior is consistently accepted by the model.

The bathtub benchmark is more challenging. Full supervision performs strongly, especially for Random Forest and Decision Tree classifiers but QSIM-based weak supervision is less accurate. This is consistent with the weak-label analysis in Table 3, where the bathtub model produced both false positives and false negatives. In this benchmark, several faults, particularly inflow-related faults remain qualitatively close to nominal behavior at the window level. Consequently, the QSIM weak labels are less discriminative for downstream learning. The LL-5 Decision Tree result further highlights the need for Macro-F1 under imbalance: despite high accuracy, the substantially lower Macro-F1 indicates that the model does not achieve balanced performance across both classes when trained from only five labels. It is also worth noting that the low-label setting can outperform QSIM weak supervision for this benchmark, especially for Decision Tree and Logistic Regression at larger label budgets. This suggests that a small number of accurate ground-truth labels may be more informative than a larger set of noisy QSIM-derived labels when the qualitative abstraction does not separate nominal and faulty behavior well.

For the Spring–Mass–Damper benchmark, QSIM-based weak supervision remains competitive with full supervision, but the effect depends on the classifier. Weak supervision is below full supervision for Random Forest, approximately matches or slightly exceeds full supervision

■ **Table 4** Accuracy (Acc) and Macro F1 by supervision regime across all benchmarks. Full and Weak QSIM use all training samples; LL- k uses only k labeled samples.

Model	Full		Weak QSIM		LL-5		LL-10		LL-20	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
<i>RC circuit</i> (train: 1242, test: 594)										
Random Forest	0.965	0.894	0.889	0.759	0.226	0.224	0.485	0.431	0.800	0.594
Decision Tree	0.939	0.849	0.837	0.714	0.131	0.129	0.689	0.565	0.848	0.577
Logistic Regression	0.736	0.619	0.758	0.637	0.219	0.218	0.495	0.440	0.783	0.660
<i>Bathtub</i> (train: 4158, test: 1782)										
Random Forest	0.937	0.853	0.493	0.400	0.357	0.331	0.433	0.367	0.552	0.461
Decision Tree	0.888	0.776	0.385	0.339	0.852	0.460	0.659	0.470	0.574	0.486
Logistic Regression	0.540	0.473	0.477	0.391	0.447	0.382	0.478	0.392	0.586	0.495
<i>Spring-mass-damper</i> (train: 12474, test: 5346)										
Random Forest	0.869	0.550	0.698	0.491	0.438	0.364	0.558	0.429	0.361	0.340
Decision Tree	0.581	0.490	0.698	0.491	0.304	0.282	0.495	0.388	0.372	0.348
Logistic Regression	0.456	0.406	0.701	0.493	0.368	0.331	0.670	0.478	0.414	0.375
<i>Heating</i> (train: 1207, test: 518)										
Random Forest	0.983	0.983	0.558	0.508	0.597	0.595	0.705	0.675	0.888	0.888
Decision Tree	0.888	0.888	0.573	0.538	0.446	0.433	0.542	0.445	0.788	0.787
Logistic Regression	0.927	0.927	0.577	0.551	0.687	0.686	0.672	0.668	0.807	0.806

for Decision Tree and clearly improves over full supervision for Logistic Regression. This is an important result because the system exhibits oscillatory continuous dynamics, making purely window-level classification difficult. The QSIM-derived labels appear to provide a useful qualitative smoothing of the dynamics: rather than relying only on local numerical values, they encode whether a window is compatible with the qualitative structure of nominal oscillatory behavior. At the same time, the moderate Macro-F1 values show that some faults remain close to nominal oscillation patterns and are therefore difficult to separate.

The heating benchmark is the most difficult real-world setting. Full supervision achieves high performance, indicating that the quantitative window features contain useful information about valve faults. In contrast, QSIM weak supervision is more conservative, with Macro-F1 between 0.508 and 0.551. This matches the weak-label results: the final heating model was designed to preserve nominal conformance over active pump-regime segments, and therefore many faulty windows remain conforming under the qualitative abstraction. Nevertheless, the weakly supervised classifiers still learn a non-trivial signal from the QSIM labels, despite the absence of manually supplied fault labels during training. Low-label supervision generally improves with larger label budgets, with LL-20 achieving strong performance for all classifiers.

Taken together, these results support the main hypothesis of the paper. QSIM-based weak supervision is not a replacement for full supervision when many accurate labels are available. Instead, it provides a physically grounded source of supervision in cold-start settings, where labeled fault data are limited or unavailable. Its effectiveness depends on how strongly faults alter the qualitative behavior of the system. When faults create clear qualitative deviations, as in the RC circuit, weak supervision is highly effective. When faults mainly induce subtle quantitative changes, as in the heating benchmark, the QSIM signal is weaker but still useful as an initial labeling mechanism.

■ **Table 5** Unsupervised anomaly detection across benchmarks. Models are trained on the first 70% of each nominal trace and tested on the remaining nominal windows together with the faulty windows from the same configurations.

Model	RC circuit		Bathtub		Spring-mass-damper		Heating	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Isolation Forest	0.325	0.269	0.386	0.312	0.348	0.289	0.419	0.417
One-Class SVM	0.840	0.544	0.826	0.555	0.703	0.461	0.786	0.722

6.3 Unsupervised Anomaly Detection

The results of the unsupervised anomaly-detection baselines are shown in Table 5. Only nominal windows are used to train the models in this configuration, and they are assessed using both held-out nominal windows and faulty windows from the same benchmark configuration. This is consistent with a frequent cold-start assumption: nominal data may be available during early system operation or commissioning, while labeled fault examples are not.

One-Class SVM consistently performs better than Isolation Forest across all benchmarks, especially with respect to Macro-F1. This suggests that, for these window-level feature distributions a boundary based description of nominal behavior is more effective than Isolation Forest’s isolation-based criterion. One-Class SVM performs strongly on the bathtub and heating benchmarks, where many faulty windows appear to deviate from the nominal feature distribution, while QSIM weak supervision is more competitive on the RC circuit and Spring–Mass–Damper benchmarks. This indicates that the two approaches capture different aspects of the data: unsupervised anomaly detection identifies distributional deviations from nominal examples, whereas QSIM-based weak supervision encodes qualitative physical consistency with a nominal behavioral model.

The Spring–Mass–Damper benchmark remains difficult for both approaches, most likely because oscillatory and regime dependent behavior creates substantial overlap between nominal and faulty windows. The heating result should also be interpreted carefully because the real operational data contain structured nominal variability and some valve-related faults can remain near nominal under particular operating regimes. Overall, these results show that unsupervised anomaly detection is a competitive cold-start baseline but not a replacement for model-based weak supervision. The distinction between these baselines and supervised or QSIM-based weakly supervised models motivates the employment of qualitative conformance as an additional physically grounded supervision signal.

6.4 Cross-configuration Generalization

Table 6 reports cross-configuration generalization under full supervision for the simulated benchmarks. In the *Same* setting, classifiers are trained and tested on windows from the same parameter configuration using ground-truth labels and the standard temporal split. In the *Cross* setting, classifiers are trained using ground-truth labels from one parameter configuration and evaluated on held-out parameter configurations. Overall, the findings indicate a low level of cross-configuration transfer. Random Forest and Decision Tree models provide very good same-configuration performance for the bathtub benchmark. However, Macro-F1 significantly declines under cross-configuration testing. This suggests that when inflow, drain area or beginning volume vary, these models learn configuration specific numerical patterns that are poorly transferable. The Spring–Mass–Damper benchmark exhibits a similar pattern: Macro-F1 remains low, indicating that the classifiers do not consistently distinguish

■ **Table 6** Cross-configuration generalization across benchmarks. “Same” trains and tests on the same parameter configuration; “Cross” tests on held-out configurations. Macro F1 drops sharply under Cross across all benchmarks, indicating limited transferability.

Model	Setting	RC		Bathtub		Spring	
		Acc	F1	Acc	F1	Acc	F1
Random Forest	Same	0.909	0.476	0.999	0.997	0.882	0.514
	Cross	0.909	0.476	0.898	0.511	0.894	0.477
Decision Tree	Same	0.173	0.162	0.997	0.992	0.850	0.514
	Cross	0.164	0.154	0.898	0.511	0.885	0.481
Logistic Regression	Same	0.315	0.238	0.693	0.590	0.576	0.482
	Cross	0.304	0.237	0.719	0.453	0.696	0.470

between nominal and defective behavior across several oscillatory regimes, while accuracy remains quite high in some cross-configuration scenarios. Additionally, the RC circuit benchmark exhibits weak Macro-F1 in both settings suggesting that it is challenging to capture the specific trace-level or phase-dependent structure of this benchmark with a straightforward cross-configuration window-level classifier.

The heating data do not offer numerous systematically varied parameter configurations such as various resistances, inflows, or spring constants, relative to the simulated benchmarks. Therefore, there is no direct analogue of training on one parameter configuration and testing on held-out configurations.

7 Conclusion

Our initial study explores whether qualitative simulation can serve as a weak supervision source for fault detection in CPS. We propose a framework that exploits qualitative conformance checking to generate binary pseudo-labels for observed traces, which are then used to train a downstream binary classifier. With this approach, we address the cold-start problem by replacing manually labeled training data with annotation derived from a QSIM model of the nominal behavior. To determine the feasibility of our approach, we evaluate the technique on four benchmarks and compared our method to full supervision, unsupervised anomaly detection, supervised learning with limited labels, and transferring of a supervised learning model to another system configuration. Across the benchmarks our method was able to produce labels, although the quality of the resulting weak labels varied with the discriminative structure of the underlying qualitative model. On systems, such as the RC circuit, with clearly distinguishable phases, our trained classifier approached the performance of the fully supervised method. In the other case, the weak labels were less informative and thus our approach could not outperform the fully supervised classifier.

A natural question is why a classifier is needed at all if qualitative conformance already yields a label and thus could be used directly for fault identification. Conformance checking is costly, scales poorly with model size (and possibly model number in case fault models are also considered), and is less intuitive to deploy. This makes the QSIM approach less suitable for continuous online monitoring. An ML classifier is trained once, inference then is efficient and independent of the foundational qualitative model. Yet, the classifier still benefits from the physical grounding of the qualitative model. Additionally, beyond runtime efficiency, the trained classifier captures the concrete behavior of the system rather than

only its qualitative abstraction. Hence, once trained on QSIM-derived labels, the classifier may detect faults (e.g., large deviations between the observed and the concrete behavior), even when such faults remain invisible to the QSIM model itself. However, the classifier may also learn spurious quantitative patterns that the qualitative model cannot detect, e.g., unrepresentative operating regimes.

Our approach is subject to several limitations. First, it presumes the availability of a QSIM model of the nominal behavior. While qualitative models are generally less demanding to construct than precise numerical ones, they still require domain expertise. Hence, in domains where this expert knowledge is scarce, the development of the model might be a barrier. Second, the qualitative model must adequately capture the admissible nominal behavior. On the one hand, if the model is too permissive, faulty traces are not detected. On the other hand, if it is too restrictive, nominal traces may be rejected. Our initial study has shown that achieving an appropriate level of discriminative rigor can be challenging, particularly for systems with multiple operating regimes. Thus, the quality of the resulting pseudo-labels is dependent on the fidelity of the qualitative model.

Several directions for future research remain. First, we currently only consider fault detection, thus extending this to multi-classification is the next logical step. This would require richer labels, which we initially plan to generate by considering qualitative models that encompass typical system faults. Second it would be to formalize the weak-supervision setting more explicitly. Sensitivity to window length, stride, abstraction thresholds and aggregation techniques should be particularly studied in future research, along with uncertainty estimates for labels generated using QSIM. Third, to counteract the weak label quality on some systems we plan on combining symbolic supervision with self-supervised learning. The classifier first learns a representation from unlabeled operational data and qualitative conformance labels are then used only for fine-tuning the fault-detection head.

References

- 1 Bernhard K. Aichernig, Harald Brandl, and Franz Wotawa. Conformance testing of hybrid systems with qualitative reasoning models. *Electronic Notes in Theoretical Computer Science*, 253(2):53–69, 2009. doi:10.1016/j.entcs.2009.09.051.
- 2 Ivan Bellanco, Elena Fuentes, Manel Vallès, and Jaume Salom. A review of the fault behavior of heat pumps and measurements, detection and diagnosis methods including virtual sensors. *Journal of Building Engineering*, 39:102254, 2021. doi:10.1016/j.jobe.2021.102254.
- 3 George M. Coghill, Ashwin Srinivasan, and Ross D. King. Qualitative system identification from imperfect data. *Journal of Artificial Intelligence Research*, 32:825–877, 2008. doi:10.1613/jair.2374.
- 4 Marco Cook, Cory Paterson, Angelos K Marnerides, and Dimitrios Pezaros. Anomaly diagnosis in cyber-physical systems. In *ICC 2022-IEEE International Conference on Communications*, pages 5445–5450. IEEE, 2022. doi:10.1109/ICC45855.2022.9838968.
- 5 Ankita Das, Roxane Koitz-Hristov, and Franz Wotawa. Using qualitative simulation models for monitoring and diagnosis. In Marcos Quiñones-Grueiro, Gautam Biswas, and Ingo Pill, editors, *36th International Conference on Principles of Diagnosis and Resilient Systems, DX 2025, Nashville, TN, USA, September 22-24, 2025*, volume 136 of *OASICS*, pages 4:1–4:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/OASICS.DX.2025.4.
- 6 Ankita Das, Roxane Koitz-Hristov, and Franz Wotawa. Towards parameter-independent fault detection using qualitative simulation. Under review for Qualitative Reasoning 2026 at IJCAI, Bremen, Germany, 2026.
- 7 Swagat Das, Devin Drake, Harindra S. Mavikumbure, Victor Cobilean, and Milos Manic. Learning beyond labels: Self-supervised methods for anomaly detection in cyber-physical systems. In *IECON 2025 – 51st Annual Conference of the IEEE Industrial Electronics Society*, pages 1–8, 2025. doi:10.1109/IECON58223.2025.11221272.

- 8 Johan De Kleer and John Seely Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24(1):7–83, 1984.
- 9 Johan de Kleer and John Seely Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24(1-3):7–83, 1984. doi:10.1016/0004-3702(84)90037-7.
- 10 Hongwen Dou and Kun Zhang. Transfer learning for cross-building forecasting of building energy and indoor air temperature in model predictive control applications. *Journal of Building Engineering*, 111:113341, 2025. doi:10.1016/j.jobe.2025.113341.
- 11 Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24(1-3):85–168, December 1984. doi:10.1016/0004-3702(84)90038-9.
- 12 Peter Fritzson. Modelica - a language for equation-based physical modeling and high performance simulation. In *Proceedings of the 4th International Workshop on Applied Parallel Computing, Large Scale Scientific and Industrial Problems, PARA '98*, pages 149–160, London, UK, UK, 1998. Springer-Verlag. doi:10.1007/BFB0095332.
- 13 Mikhail Genkin and J. J. McArthur. Using reinforcement learning with transfer learning to overcome smart building cold start. In *International Conference on Computational Science and Computational Intelligence, CSCI 2022, Las Vegas, NV, USA, December 14-16, 2022*, pages 713–718. IEEE, 2022. doi:10.1109/CSCI58124.2022.00130.
- 14 Jie Gui, Tuo Chen, Jing Zhang, Qiong Cao, Zhenan Sun, Hao Luo, and Dacheng Tao. A survey on self-supervised learning: Algorithms, applications, and future trends. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9052–9071, 2024. doi:10.1109/TPAMI.2024.3415112.
- 15 Srinivas Katipamula and Michael R. Brambley. Review article: Methods for fault detection, diagnostics, and prognostics for building systems—a review, part i. *HVAC&R Research*, 11(1):3–25, 2005. doi:10.1080/10789669.2005.10391123.
- 16 Roxane Koitz-Hristov, Liliana Marie Prikler, and Franz Wotawa. Beyond Static Diagnosis: A Temporal ASP Framework for HVAC Fault Detection. In Marcos Quinones-Grueiro, Gautam Biswas, and Ingo Pill, editors, *36th International Conference on Principles of Diagnosis and Resilient Systems (DX 2025)*, volume 136 of *Open Access Series in Informatics (OASIs)*, pages 1:1–1:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIs.DX.2025.1.
- 17 Benjamin Kuipers. The limits of qualitative simulation. In Aravind K. Joshi, editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence. Los Angeles, CA, USA, August 1985*, pages 128–136. Morgan Kaufmann, 1985. URL: <http://ijcai.org/Proceedings/85-1/Papers/023.pdf>.
- 18 Benjamin Kuipers. Qualitative reasoning: Modeling and simulation with incomplete knowledge. *Automatica*, 25(4):571–585, 1989. doi:10.1016/0005-1098(89)90099-X.
- 19 Benjamin J. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29(3):289–338, 1986. doi:10.1016/0004-3702(86)90073-1.
- 20 Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2019. doi:10.1109/TKDE.2018.2876857.
- 21 Yuan Luo, Ya Xiao, Long Cheng, Guojun Peng, and Danfeng (Daphne) Yao. Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities. *ACM Comput. Surv.*, 54(5), May 2021. doi:10.1145/3453155.
- 22 Antonio M. Martínez-Heredia and Sebastián Ventura. Weak supervision: A survey on predictive maintenance. *WIREs Data Mining and Knowledge Discovery*, 15(2):e70022, 2025. doi:10.1002/widm.70022.
- 23 Morteza Mohaqeqi, Mohammad Reza Mousavi, and Walid Taha. Conformance testing of cyber-physical systems: A comparative study. *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.*, 70, 2014. doi:10.14279/TUJ.ECEASST.70.982.

- 24 Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *The VLDB Journal*, 29(2):709–730, 2020. doi:10.1007/s00778-019-00552-1.
- 25 Jagath Sri Lal Senanayaka, Huynh Van Khang, and Kjell G. Robbersmyr. Toward self-supervised feature learning for online diagnosis of multiple faults in electric powertrains. *IEEE Transactions on Industrial Informatics*, 17(6):3772–3781, 2021. doi:10.1109/TII.2020.3014422.
- 26 Maxim Shcherbakov and Cuong Sai. A hybrid deep learning framework for intelligent predictive maintenance of cyber-physical systems. *ACM Trans. Cyber-Phys. Syst.*, 6(2), May 2022. doi:10.1145/3486252.
- 27 Georgios Tertytchny, Nicolas Nicolaou, and Maria K. Michael. Classifying network abnormalities into faults and attacks in iot-based cyber physical systems using machine learning. *Microprocessors and Microsystems*, 77:103121, 2020. doi:10.1016/j.micpro.2020.103121.
- 28 Louise Travé-Massuyès and Robert Milne. Application oriented qualitative reasoning. *The Knowledge Engineering Review*, 10(2):181–204, 1995. doi:10.1017/S0269888900008146.
- 29 Sven Werner. International review of district heating and cooling. *Energy*, 137:617–631, 2017. doi:10.1016/j.energy.2017.04.045.
- 30 Timothy Wiley, Claude Sammut, and Ivan Bratko. Qualitative simulation with answer set programming. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, pages 915–920, 2014. doi:10.3233/978-1-61499-419-0-915.
- 31 Shichao Xu, Yixuan Wang, Yanzhi Wang, Zheng O’Neill, and Qi Zhu. One for many: Transfer learning for building hvac control. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, BuildSys ’20*, pages 230–239, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3408308.3427617.
- 32 Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence, AAAI’23/IAAI’23/EAAI’23*. AAAI Press, 2023. doi:10.1609/aaai.v37i9.26317.
- 33 Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, January 2018. doi:10.1093/nsr/nwx106.

Reinforcement Learning-Driven Predictive Maintenance Planning via Timed Automata Modeling

Ferhat Tamssaouet ✉ 

LAAS-CNRS, Université de Toulouse, France

Pauline Ribot ✉ 

LAAS-CNRS, Université de Toulouse, France

Yannick Pencolé ✉ 

LAAS-CNRS, CNRS, Université de Toulouse, France

Abstract

Predictive maintenance of multi-component systems requires decisions that account for both local component degradation and system-level redundancy. This paper proposes a formal framework for synthesizing interpretable maintenance policies from a system functional architecture. The system is modeled as a network of Dynamic Priced Timed Game Automata (DPTGA): each component automaton encodes degradation dynamics, stochastic failure windows, and repair and replacement costs, while a System Health Monitor automaton is automatically derived from the minimal cut sets of the functional architecture and detects system failure as soon as any cut set becomes unavailable. We execute the DPTGA semantics in an explicit-state simulator and optimize a compact threshold policy with the Cross-Entropy Method (CEM). The policy contains one age-fraction threshold per component and an embedded subsystem-safe predicate that forbids any preventive action that would immediately realize a cut set, making the learned strategy provably safe and directly interpretable as a rule-based maintenance plan. On a seven-component benchmark comprising a parallel subsystem and a 2-out-of-5 redundant block, the CEM policy reduces mean cumulative cost by 15% and median cost by 16% relative to the best rule-based heuristic (corrective, systematic, and condition-based maintenance), while maintaining a 98.7% system survival rate. The approach therefore provides a compact and explainable alternative to deep reinforcement learning for architecture-aware predictive maintenance.

2012 ACM Subject Classification Computer systems organization → Dependable and fault-tolerant systems and networks; Computing methodologies → Reinforcement learning; Software and its engineering → Model-driven software engineering

Keywords and phrases System Remaining Useful Life, Predictive Maintenance, Timed Automata, Reinforcement Learning

Digital Object Identifier 10.4230/OASICS.DX.2026.12

1 Introduction

Maintaining complex systems requires a proactive approach to ensure reliability, minimize downtime, and extend the operational lifespan of the whole system. There are several types of maintenance. Corrective maintenance consists in restoring the system to its normal condition after a failure without any kind of anticipation. With such maintenance strategy, the system always has unexpected downtime and due to the type of failure, the restoring time might not be anticipated which drastically decreases the level of reliability of the system. To avoid such problems, preventive maintenance consists of a set of planned actions to avoid failures and extend the lifespan of the system, such plans can be systematic (e.g. actions performed after a fixed and regular amount of time) or conditional (based on some system conditions). With preventive maintenance, action plans are scheduled in advanced with anticipated costs



© Ferhat Tamssaouet, Pauline Ribot, and Yannick Pencolé;
licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 12; pp. 12:1–12:20

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

to minimize downtime however such strategy can be costly as the planned action, like a replacement action, might not be necessary. Predictive maintenance is going one step forward and aims at scheduling repair or replacement actions before the occurrence of any failure by performing an estimation of the Remaining Useful Life (RUL) based on the use of health indicators that collect measurements at operating time. At the component-based level, the estimation of RUL usually relies on physical measurements and degradation or ageing models that are either obtained by expertise or by machine learning techniques. At the system-based level, the objective is to estimate the System Remaining Useful Life (SRUL). In complex systems like aircrafts [16], semiconductor manufacturing systems [22], railways systems [11], the estimation of the SRUL cannot simply be summarized as the minimal RUL on one of its equipment as these systems are composed of redundant components ensuring that if one is failing the system still operates. Therefore, the SRUL not only depends on the RUL of its components but also on the functional structure of the system [29].

This paper addresses the problem of determining a predictive maintenance strategy for such complex systems based on the functional structure of the system and a real-time estimation of the components' RULs. We first propose the definition of a formal architecture based on *Dynamic Priced Time Game Automata* (DPTGA) that models the set of possible maintenance strategies with their respective costs. The architecture is firstly expressed through functional minimal cut sets, which may be extracted from a fault tree of the system, a reliability block diagram, or another dependability model. These cut sets are then used to build a network of DPTGA embedded with some maintenance/repair costs. Component automata model degradation clocks, stochastic failure windows, repair and replacement durations, and costs. A System Health Monitor (SHM) synchronizes with component events and detects system failure whenever a cut set becomes unavailable. We then propose a first method to search for a maintenance strategy that relies on a reinforcement-learning-based policy search. More specifically, this policy is obtained by simulating this DPTGA network and optimizing a compact threshold policy with the Cross-Entropy Method (CEM). The central design choice is to keep the learned policy interpretable. Instead of learning an unconstrained neural controller, the policy contains one preventive-maintenance threshold per component and a system-aware safety predicate that forbids preventive actions that would immediately realize a cut set. This makes the learned strategy easy to inspect and directly usable as a rule-based maintenance plan.

The paper is organized as follows. Section 2 positions the contribution with respect to timed-automata-based maintenance modeling, reinforcement learning for predictive maintenance, and learning-augmented controller synthesis. Section 3 introduces the necessary background on fault trees, minimal cut sets and dynamic priced timed game automata. Section 4 presents the DPTGA models for the components and the SHM. Section 5 formulates the cost-minimization problem and the CEM policy search. Section 6 reports the case study and discusses the experimental results and the learned policy. Section 7 concludes and outlines directions for future work.

2 Related Work

The proposed framework lies at the intersection of three threads: (i) timed-automata-based maintenance modeling, (ii) reinforcement learning for multi-component predictive maintenance, and (iii) learning-augmented controller synthesis on timed game automata.

Timed Automata for Fault-tree-driven Maintenance

The closest line of work translates fault trees and their extensions into networks of priced timed automata and uses Statistical Model Checking (SMC) to estimate dependability metrics. Ruijters and Stoelinga's *Fault Maintenance Trees* [33, 32] extend industry-standard fault trees with preventive maintenance and repair templates, translate them to UPPAAL priced timed automata and use UPPAAL-SMC [6] to estimate reliability, mean-time-to-failure and maintenance cost on railway and smart-building case studies. Kumar et al. [14] introduce *Attack-Fault-Maintenance Trees* (AFMTs), a safety-security hybrid that extends AFTs [15] with proactive maintenance and reactive repair, again analyzed via SMC. In a broader dependability perspective, Rubin and Du [30] review Dynamic Fault Tree (DFT) analysis for complex redundant systems with capacity degradation, showing that DFTs can capture sequence-dependent failures, reconfiguration, and redundancy, while still facing scalability issues and difficulties in representing partial degradation states. These works provide rich formal modeling frameworks for maintenance analysis, but the maintenance strategy remains a manual exploration of a finite set of templates or scenarios, the optimization does not exploit online per-component RUL information that Prognostics and Health Management (PHM) pipelines now make available.

Reinforcement Learning for Multi-component Maintenance

Several recent works formulate predictive maintenance as a sequential decision problem and solve it with optimization or (deep) reinforcement learning techniques. Regattieri et al. [28] propose a simulation-based framework for optimizing aircraft maintenance policies by combining failure-process modeling, repair-process modeling, and Monte Carlo evaluation of preventive/corrective maintenance mixes. Their objective minimizes a total service cost including corrective, preventive, inspection, and spare-part stock terms, but these costs are fixed parameters rather than degradation-dependent quantities. Reinforcement-learning (RL) approaches go further by learning state-dependent policies: Andriotis and Papakonstantinou [3] couple constrained Partially Observable Markov Decision Processes (POMDPs) with multi-agent Deep Reinforcement-learning (DRL) to derive inspection-and-maintenance policies for deteriorating infrastructure under budget and risk constraints. Najafi and Lee [23] embed a proportional-hazards degradation model into a DRL agent for multi-unit repair-based maintenance. Liu et al. [19] address multi-component systems with complex structure, and Phuc et al. [7] use multi-agent DRL for systems with multi-dependent components. Pincirolì et al. [25, 26] demonstrate that PPO with imitation pre-training significantly outperforms scheduled and condition-based baselines on wind-farm operation and maintenance equipped with PHM signals. Two recent surveys [24, 20] catalogue state designs, reward shaping, and open issues in scalability, interpretability, and certification. However, these approaches typically rely on hand-coded transition kernels, simulation models, or discrete-time Markov Decision Process (MDP) abstractions, and do not derive the decision environment from a *by-construction* guarantee that the synthesized policy respects a system-level safety predicate.

Coupling RL with PTGA: Model-checking versus Simulation

A complementary line couples reinforcement learning with verification engines that operate on (Priced) Timed Game Automata (PTGA). UPPAAL Stratego [5] synthesizes near-optimal strategies for stochastic priced timed games by combining symbolic exploration with reinforcement learning, and has been instantiated for online controller synthesis in floor heating [17],

energy-aware buildings [1], STOchastic Model-Predictive Control (STOMPC) [9] and, more recently, discretized Euclidean MDPs [12]. While model-checking-based RL provides strong formal guarantees, it inherits the well-known state-space explosion of dense-time verification: the cost per training episode grows steeply with the number of components and clocks, which is precisely the regime targeted by predictive maintenance of multi-component systems [10]. To the best of our knowledge, no previous work has reported a population-based policy search on a PTGA model of a fault-tree-derived multi-component maintenance problem with RUL-dependent dynamic maintenance costs.

Positioning of this paper

We retain the PTGA modeling language of the SMC/Stratego line – locations, clocks, guards, MCS-derived health monitor, costs – but switch from symbolic exploration to an explicit-state simulator written from the formal semantics of the model. This trade-off sacrifices on-line model-checking guarantees in exchange for a simulation throughput that is compatible with population-based policy search; the safety properties that the synthesized policy must satisfy are then enforced *by construction* via a parameterized policy class with an embedded subsystem-safe predicate (Section 5). Compared with the DRL line, we trade representational power for interpretability and sample efficiency by restricting the policy to a compact threshold class with one parameter per component, optimized by the CEM [31]. The result is a pipeline that maps a fault tree to a deployable maintenance policy without requiring either a model-checking back-end or a deep neural network in the inner loop.

3 Preliminaries

3.1 Failure Prognostics

Failure prognostics aims to estimate the future health state of a component from observable quantities such as sensor measurements, usage profiles, environmental conditions, or degradation indicators. The central output of a prognostic model is the RUL, defined as the time remaining before a component can no longer fulfill its intended function. Depending on the approach, the RUL can be delivered as a point estimate, a probability distribution, or a confidence interval, each carrying different implications for decision-making under uncertainty [4, 37]. Prognostic methods broadly fall into three families: physics-based models, which exploit degradation equations derived from domain knowledge; data-driven models, which learn degradation patterns from historical run-to-failure data; and hybrid approaches that combine both.

While most of the prognostics literature addresses single-component systems, industrial assets are inherently multi-component, and the health of one subsystem can influence or accelerate the degradation of others. System-level prognostics therefore extends individual RUL estimation to the joint health state of an assembly, accounting for load sharing, failure propagation, and structural and functional dependencies [35].

In the present work, each component i is characterized by a stochastic failure deadline sampled uniformly from the interval $[\text{minRUL}_i, \text{maxRUL}_i]$, which represents the prognostic uncertainty window produced by an upstream health-monitoring module. The maintenance policy does not observe this deadline directly; it only has access to the degradation clock c_i and the publicly available RUL bounds. This formulation captures a realistic PHM scenario in which prognostic models constrain the failure window without revealing the exact failure instant, thereby requiring the policy to reason and act under uncertainty.

3.2 Minimal Cut Sets, Fault Trees

The proposed framework handles multi-component systems with complex architectural configurations (see for instance the functional architecture of components in Figure 4). In such an architecture, an *event* is the occurrence of a failure in a part of the system, and among them *elementary events* (also called *basic events*) are failures at the component level. Our proposed framework relies on *Minimal Cut Sets* (MCS for short) that are minimal sets of elementary events whose occurrence implies the whole system's failure. Minimal cut sets can be automatically extracted for instance from *fault trees*. Fault tree analysis is a top-down, deductive failure analysis method that uses a boolean logic tree structure to represent the elementary events and their combinations leading to the occurrence of an undesirable state (called the top event), representing system failure [18]. The combinations of elementary events are modeled using standard logic gates (e.g., AND, OR, k-out-of-n, XOR). Fault tree analysis is a very common analysis method that is available for many existing or future systems in almost all sectors of activity [18]. There are several algorithms that can be used to obtain minimal cut sets, such as MOCUS (Method of Obtaining CUt Sets) [8], BDD (Binary Decision Diagrams) [27], etc. The proposed framework only relies on minimal cut sets. While a fault tree approach can be used for representing the system architecture, it is not necessary, similar approaches can be leveraged within the same framework, such as reliability/functional block diagrams, bond graphs, etc., as long as minimal cut sets can be extracted from them.

► **Example 1.** The functional architecture of the system that is studied throughout this paper is illustrated in Figure 4. It is a system with 7 components with functional redundancies for high reliability.

- **Subsystem 1 (Components A and B):** A parallel redundancy where both components A and B must fail simultaneously to trigger a system-level failure (i.e., a logic AND gate).
- **Subsystem 2 (Components C, D, E, F, G):** A *k*-out-of-*n* configuration. In this setup, the subsystem fails if more than 3 components fail (i.e., 4 out of 5 components become unavailable). This implies the system requires at least two of these five components to remain operational to maintain system integrity.

In this example, there are 6 minimal cut sets $\mathcal{K}_1, \dots, \mathcal{K}_6$: $\mathcal{K}_1 = \{A, B\}$, $\mathcal{K}_2 = \{C, D, E, F\}$, $\mathcal{K}_3 = \{C, D, E, G\}$, $\dots$, $\mathcal{K}_6 = \{D, E, F, G\}$.

3.3 Dynamic Priced Timed Game Automata and Strategies

The maintenance optimization framework that we propose relies on a set of *Dynamic Priced Timed Game Automata* (DPTGA for short). DPTGA is an extension of the classical Timed Automata from [2].

► **Definition 2** (Dynamic Priced Timed Game Automata). A Dynamic Priced Timed Game Automaton $\mathcal{G}$ is a tuple

$$(L, X, \ell_0, \Sigma, C, E, \text{inv}, f)$$

where L is a finite set of locations, X is a set of variables x over a domain $\text{Dom}(x)$, $\ell_0 \in L$ is the initial location, $\Sigma = \Sigma_c \cup \Sigma_u$ is the set of actions (partitioned into controllable and uncontrollable actions), C is a finite set of real-valued clocks, $E \subseteq L \times \text{Guard}(C) \times \Sigma \times 2^C \times 2^X \times L$ is a finite set of transitions, $\text{inv} : L \mapsto \text{Guard}(C)$ maps an invariant to each location, $f : E \times \mathbb{R}^{|C|} \rightarrow \mathbb{R}$ maps a time dependent cost function to each transition.

As any timed automaton, a DPTGA is a finite-state automaton enriched with a finite set of real-valued clocks. Guards and invariants are conjunctions of atomic constraints of the form $c_1 \sim v$ or $c_1 - c_2 \sim v$ with $c_1, c_2 \in C$, $v \in \mathbb{R} \cup X$, and $\sim \in \{<, \leq, =, \geq, >\}$. The first extension provided by a DPTGA is that the set of actions is split into two disjoint sets (controllable and uncontrollable actions) so that the DPTGA is considered as a two-player game (an agent which is able to decide controllable actions from Σ_c in interaction with its environment that can produce uncontrollable actions from Σ_u). The second extension is that locations L and transitions E are associated with a cost function f that determines the cost of triggering the transition depending on the current values of the clocks. The third extension is the possibility to update variables $x := v, v \in \text{Dom}(x)$ in a transition, similarly as a clock.

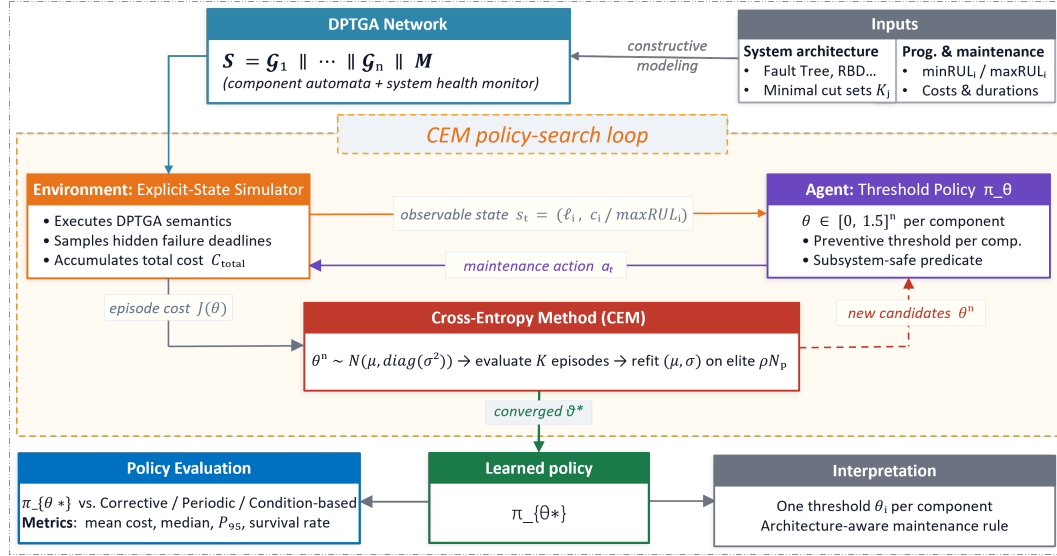
The semantics of a DPTGA is a *Dynamic Priced Timed Transition System* (DPTTS) over states (ℓ, v_C, v_X) with $\ell \in L$, clock valuations $v_C : C \rightarrow \mathbb{R}_{\geq 0}$, variable valuations $v_X : X \rightarrow \text{Dom}(X)$ satisfying $\text{inv}(\ell)$. Transitions are either (i) *time elapse* (labelled $(\ell, v_C, v_X) \xrightarrow{d} (\ell, v_C + d, v_X)$), advancing all clocks by a non-negative delay d while continuously satisfying the invariant, or (ii) *discrete* edges (labelled $(\ell, v_C, v_X) \xrightarrow{a} (\ell', v'_C, v'_X)$) resulting from the fire of a transition $(\ell, g, a, R_C, R_X, \ell') \in E$ with $(v_C, v_X) \models g$, performing action $a \in \Sigma$, resetting clocks in R_C , updating variables in R_X , and moving to ℓ' with $v'_C = v_C[R_C] \wedge v'_X = v_X[R_X] \models \text{inv}(\ell')$. A *run* $\alpha \in \text{Runs}(\mathcal{G})$ is a finite sequence of transitions $s_0 \xrightarrow{a_1} s_1 \dots \xrightarrow{a_n} s_n$ from the underlying DPTTS starting at state $s_0 = (\ell_0, \{0, \dots, 0\}, \{v_{x_1}, \dots, v_{x_m}\})$. The cost of α is defined as:

$$\text{cost}(\alpha) = \sum_{\substack{(\ell_{i-1}, v_{C_{i-1}}, v_{X_{i-1}}) \xrightarrow{a_i} (\ell_i, v_{C_i}, v_{X_i}), \\ a_i \in \Sigma, \\ t_i = (\ell_i, g, a_i, R_C, R_X, \ell_{i+1}) \in E}} f(t_i, v_{C_{i-1}}). \quad (1)$$

The proposed framework relies on the composition of DPTGA's defined here below. As detailed in Section 4, the objective is to determine an optimal maintenance strategy for a single agent which implies that any controllable (resp. uncontrollable) action at the component level is a controllable (resp. uncontrollable) action at the system level. Moreover, the global cost of a shared action (if any) is the sum of the costs.

► **Definition 3** (Composition of DPTGA). *Let $\mathcal{G}_i = (L_i, X_i, \ell_{0,i}, \Sigma_{ci} \cup \Sigma_{ui}, C_i, E_i, \text{inv}_i, f_i), i \in \{1, 2\}$ be a couple of DPTGA, the composition $\mathcal{G}_1 \parallel \mathcal{G}_2$ is the DPTGA $\mathcal{G} = (L, X, \ell_0, \Sigma, C, E, \text{inv}, f)$ such that:*

- $L = L_1 \times L_2$;
- $X = X_1 \cup X_2$;
- $\ell_0 = (\ell_{0,1}, \ell_{0,2})$;
- $\Sigma_u = \Sigma_{u1} \cup \Sigma_{u2}$, $\Sigma_c = \Sigma_{c1} \cup \Sigma_{c2}$ with $\Sigma_u \cap \Sigma_c = \emptyset$, $\Sigma = \Sigma_u \cup \Sigma_c$;
- $C = C_1 \cup C_2$;
- E is the set of transitions:
 - synchronized transitions: $((l_1, l_2), g_1 \wedge g_2, a, R_{C1} \cup R_{C2}, R_{X1} \cup R_{X2}, (l'_1, l'_2))$ if $(l_1, g_1, a, R_{C1}, R_{X1}, l'_1) \in E_1$ and $(l_2, g_2, a, R_{C2}, R_{X2}, l'_2) \in E_2$;
 - asynchronous transitions: $((l_1, l_2), g_1, a_1, R_{C1}, R_{X1}, (l'_1, l_2))$ if $(l_1, g_1, a_1, R_{C1}, R_{X1}, l'_1) \in E_1$ and $a_1 \notin \Sigma_2$, $((l_1, l_2), g_2, a_2, R_{C2}, R_{X2}, (l_1, l'_2))$ if $(l_2, g_2, a_2, R_{C2}, R_{X2}, l'_2) \in E_2$ and $a_2 \notin \Sigma_1$;
- $\text{inv}(l_1, l_2) = \text{inv}_1(l_1) \wedge \text{inv}_2(l_2)$;
- $f = f_1 + f_2$.



■ **Figure 1** Overview of the proposed DPTGA-based maintenance optimization framework.

A *strategy* is a policy $\pi : \text{Runs}(\mathcal{G}) \rightarrow \Sigma_c$ over the possible runs α of the DPTGA that associates a set of possible controllable actions from Σ_c to perform with each possible run α . A strategy π is said to be *memoryless* if the set of controllable actions only depends on the last state s of the run ($\pi(s) \in \Sigma_c$). A *winning strategy* is a strategy that ensures any run respecting the strategy reaches one location of a set *Goal* (a run respects a strategy π if, in any state of the run, one of the controllable actions proposed by the strategy π (if not empty) is performed by the run). Such a strategy is said to be *optimal* if it leads to runs with a minimal cost $\text{cost}(\alpha)$.

3.4 Reinforcement Learning

Reinforcement Learning (RL) formalizes sequential decision-making through the framework of a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P}(s' | s, a)$ the transition kernel, $\mathcal{R}(s, a)$ the reward function, and $\gamma \in [0, 1]$ a discount factor. An agent observes the current state s_t , selects an action a_t according to a policy $\pi(a | s)$, transitions to s_{t+1} , and receives a scalar reward r_t . The objective is to find a policy π^* that maximizes the expected discounted return $\mathbb{E}_\pi \left[\sum_{t=0}^T \gamma^t r_t \right]$.

RL algorithms are broadly divided into value-based methods, which estimate the optimal action-value function $Q^*(s, a)$ and derive a greedy policy from it (e.g., Q-learning [36], DQN [21]), and policy-gradient methods, which directly optimize the parameters of a parameterized policy π_θ using gradient estimates (e.g., REINFORCE, PPO [34]).

When the policy class is smooth and the parameter space is low-dimensional, gradient-free population-based methods offer a compelling alternative. The CEM [31] is one such approach: it maintains a distribution over policy parameters, iteratively samples a population of candidate policies, retains the elite fraction that achieves the highest return, and refits the distribution to these elites. CEM avoids the variance issues of stochastic gradient estimators and is particularly well-suited to maintenance scheduling problems, where the action space is combinatorial but the policy can often be compactly parameterized. In this work, we rely on CEM to optimize the maintenance policy described in Section 5.

4 Modeling

This section presents the formal modeling of the set of possible predictive maintenance strategies of a multi-component system. It provides a constructive mapping from architectural minimal cut sets (see Section 3.2) to a network of dynamic priced timed game automata (see Section 3.3) composed of component models and a SHM.

4.1 Component Modeling

In the proposed modeling framework, we assume that the system is monitored by a Health Management System that is composed of a set of prognostic agents (one per component). The prognostic agent for component i is in charge of estimating at a given time t the RUL of the component by providing a couple of pessimistic and optimistic RULs, namely $\min\text{RUL}_i$ and $\max\text{RUL}_i$.

Each component i is then formalized as a DPTGA $\mathcal{G}_i = \langle L_i, X_i, \ell_{0i}, \Sigma_i, C_i, E_i, \text{inv}_i, f_i \rangle$ with locations $L_i = \{\text{faultless}_i, \text{faulty}_i, \text{repair}_i, \text{replacement}_i\}$ and initial location $\ell_{0i} = \text{faultless}_i$ (see Figure 2). Location faultless_i models the fact that the prognostic agent considers the component has no fault and its remaining useful life is greater or equal to $\min\text{RUL}_i$. Location faulty_i models the health of the component after the occurrence of the fault that is estimated to be within time interval $[\min\text{RUL}_i, \max\text{RUL}_i]$ from the initial location. Location replacement_i models the state of the component at the time when a corrective maintenance action is performed, that is the full replacement of the faulty component. In this model, we also might expect that the component can be fixed (location repair_i) before the complete failure of the component. DPTGA $\mathcal{G}_i$ handles two clocks $C_i = \{c_i, r_i\}$. Clock c_i tracks the component's degradation level. Initialized with $c_i = 0$ in the initial state, it ensures that the health state becomes faulty with the RUL estimation $[\min\text{RUL}_i, \max\text{RUL}_i]$ if no repair action beginRep_i is performed before. This behavior is modeled in $\mathcal{G}_i$ by the invariant $\text{inv}(\text{faultless}_i) = c_i \leq \max\text{RUL}_i$ (which enforces exiting the faultless location before exceeding the component's absolute maximum lifespan) combined with the guard $c_i \geq \min\text{RUL}_i$ on the transition from faultless_i to faulty_i (which allows this transition only when clock c_i exceeds the pessimistic RUL estimation). This clock is reinitialized ($c_i := 0$) every time the component leaves the replacement or repair locations to become once again faultless . Clock r_i tracks the duration of maintenance activities. The repair duration is $t_{\text{rep},i}$ and the replacement duration is $t_{\text{repl},i}$ ¹. Each component contributes to a shared variable $X_i = \{\text{Down}\}$, a set of component indices initialized to $\text{Down} = \emptyset$. Whenever component i enters location faulty_i or repair_i , index i is added to Down ; whenever it returns to faultless_i , index i is removed. Components in replacement_i already carry index i in Down (set upon entering faulty_i) and release it only upon returning to faultless_i . Hence Down always reflects the set of currently unavailable components.

The action space Σ_i is partitioned into controllable actions $\Sigma_{ci} = \{\text{beginRep}_i, \text{replace}_i\}$, representing maintainer decisions, and uncontrollable actions $\Sigma_{ui} = \{\text{failure}_i, \tau_i\}$. Action failure_i is stochastic: it is fired by the environment when the degradation clock c_i enters the uncertainty window $[\min\text{RUL}_i, \max\text{RUL}_i]$, reflecting the fact that the exact failure instant is hidden from the agent. Action τ_i is autonomous: it fires deterministically once the maintenance clock r_i has elapsed the prescribed duration ($t_{\text{rep},i}$ for a repair, $t_{\text{repl},i}$ for a replacement), returning the component to faultless_i with both clocks reset to zero. The transition dynamics integrates the cost model described below.

¹ For simplicity, maintenance durations are fixed constants; in general they can be represented by intervals.

- **Failure (uncontrollable):** $t_1 = \text{faultless}_i \xrightarrow{\text{failure}_i} \text{faulty}_i$ represents the failure of the component i . The associated cost models the loss of residual value through a linear depreciation:

$$f_i(t_1, \{c_i, r_i\}) = C_{\text{fail}}(c_i) = \left| C_{\text{comp},i} - \frac{C_{\text{comp},i}}{\text{maxRUL}_i} c_i \right|. \quad (2)$$

where $C_{\text{comp},i}$ represents the initial component cost. This implies that if the system fails before reaching its maximum RUL, a proportional fraction of its asset value is lost. The current formulation assumes a linear decrease of asset value over time and uniform failure probability inside the RUL interval. However, the DPTGA framework readily accommodates non-linear degradation profiles and alternative reliability distributions.

- **Preventive maintenance (controllable):** $t_2 = \text{faultless}_i \xrightarrow{\text{beginRep}_i} \text{repair}_i$ sends the component for repair while it is still faultless, setting $r_i := 0$ and inserting i into *Down*. It incurs the cost defined as follows:

$$f_i(t_2, \{c_i, r_i\}) = C_{\text{rep}}(c_i) = C_{\text{fixed_rep},i} + C_{\text{rate},i} t_{\text{rep},i} + C_{\text{comp},i} \left(1 - \frac{c_i}{\text{maxRUL}_i} \right), \quad (3)$$

which separates a fixed logistic overhead $C_{\text{fixed_rep},i}$, the cost $C_{\text{rate},i}$ (typically the labor cost) proportional to the repair duration $t_{\text{rep},i}$, and the residual asset value lost when scrapping a component that still had useful life remaining. This decomposition ensures that preventive maintenance is not strictly dominated by corrective maintenance: a repair executed early (small c_i) recovers more residual value and thus costs more than a late repair, restoring the intuitive ordering “preventive cheaper when executed near the end of life, corrective cheaper when failure was imminent anyway.” The cost remains linear in c_i and is directly usable as a DPTGA transition cost.

- **Corrective replacement (controllable):** $t_3 = \text{faulty}_i \xrightarrow{\text{replace}_i} \text{replacement}_i$ initiates the corrective replacement procedure, resetting the maintenance clock ($r_i := 0$). It incurs the cost

$$f_i(t_3, \{c_i, r_i\}) = C_{\text{repl},i} = C_{\text{comp},i} + C_{\text{fixed_repl},i}, \quad (4)$$

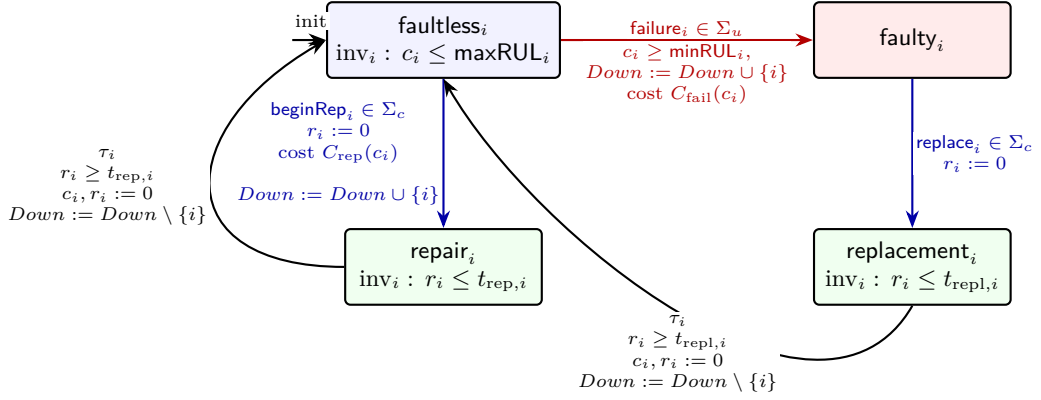
where $C_{\text{comp},i}$ is the procurement cost of a new component and $C_{\text{fixed_repl},i}$ is the logistic overhead of the replacement operation. Once the replacement clock reaches $r_i \geq t_{\text{repl},i}$, the autonomous action τ_i fires and returns the component to faultless_i , resetting both clocks and sampling a new hidden failure deadline.

4.2 Generating the System Model from the Fault Tree

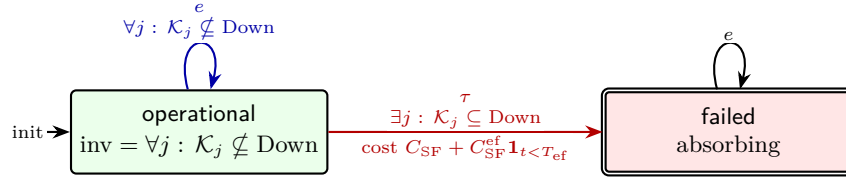
The SHM is a DPTGA $\mathcal{M}$ that tracks the system-level operational status by reading the shared variable *Down*, maintained by the component automata as described in Section 4.1. Recall that after each component transition, *Down* contains the indices of all components that are currently unavailable (i.e., in location faulty_i , repair_i , or replacement_i). The SHM has two locations: *operational* (initial) and *failed* (absorbing). It stays in *operational* as long as no minimal cut set is entirely contained in *Down*, encoded by the location invariant

$$\text{inv}(\text{operational}) = \forall j : \mathcal{K}_j \not\subseteq \text{Down}.$$

Whenever a component transition causes *Down* to cover a complete cut set, the SHM synchronizes on the triggering event e and either loops in *operational* if the invariant is still satisfied, or fires the autonomous τ -transition to the absorbing *failed* location when $\exists j : \mathcal{K}_j \subseteq \text{Down}$ (see Figure 3). Upon entering *failed*, a system-failure cost C_{SF} is incurred, augmented by an early-failure penalty $C_{\text{SF}}^{\text{ef}}$ if the collapse occurs before a prescribed horizon T_{ef} .



■ **Figure 2** Dynamic Priced Timed Game Automaton $\mathcal{G}_i$ for component i . Red denotes the uncontrollable failure transition, while blue denotes controllable maintenance actions. Clock c_i measures degradation and clock r_i measures the duration of the ongoing maintenance activity. Returning to faultless_i resets both clocks and samples a new hidden failure deadline in the RUL interval.



■ **Figure 3** DPTGA of the SHM $\mathcal{M}$. After each component event e , the SHM reads the shared variable $Down$. It remains in **operational** as long as no minimal cut set $\mathcal{K}_j$ is entirely contained in $Down$. As soon as $\exists j : \mathcal{K}_j \subseteq Down$, the invariant on **operational** is violated and the autonomous τ -transition fires, moving the SHM to the absorbing location **failed** and incurring the system-failure cost.

5 Threshold Policy and Cross-Entropy Optimization

The objective is to find a policy that minimizes the expected cumulative operational cost over a finite horizon T while preventing catastrophic system failures. The global system is the composition $\mathcal{S} = \mathcal{G}_1 \parallel \dots \parallel \mathcal{G}_n \parallel \mathcal{M}$ of the component automata and the SHM of Figure 3. The control problem is intrinsically a Partially Observable Markov Decision Process (POMDP): the full state at time t includes the location $\ell_{i,t}$ and both clock valuations $(c_{i,t}, r_{i,t})$ of each component automaton $\mathcal{G}_i$, but the per-component failure deadline sampled in $[\text{minRUL}_i, \text{maxRUL}_i]$ is hidden from the agent. The observable state is therefore the projection

$$o_t = \left(\ell_{i,t}, \frac{c_{i,t}}{\text{maxRUL}_i} \right)_{i=1}^n \in \mathcal{O}, \quad (5)$$

where the normalized degradation ratio $c_{i,t}/\text{maxRUL}_i \in [0, 1]$ indicates the extent to which the component i has aged relative to its worst-case lifespan. At each step, the agent selects an action from

$$\mathcal{A} = \{\text{wait}\} \cup \{\text{beginRep}_i, \text{replace}_i \mid i = 1, \dots, n\}, \quad (6)$$

consistent with the controllable action sets Σ_{ci} defined in Section 4.1. The objective is to minimize the expected total cost

$$J(\pi) = \mathbb{E}_\pi[C_{\text{total}}] = \mathbb{E}_\pi \left[\sum_{\text{transitions}} f_i(t_k, v_{C,k}) + C_{\text{SF}} \mathbf{1}_{\mathcal{M} \rightarrow \text{failed}} \right], \quad (7)$$

where $f_i(t_k, v_{C,k})$ are the DPTGA transition costs of Section 4.1 and C_{SF} is the system-failure penalty if $\mathcal{M}$ reaches the absorbing **failed** location (Section 4.2).

Standard value-based RL – learning the action-value function $Q(o, a)$ via Bellman updates – fails to produce useful policies in this setting. Even with optimistic initialization, the agent collapses to the degenerate “never trigger preventive maintenance” strategy. The failure is structural: the catastrophic failure penalty is a sparse terminal reward, while the cost of every preventive action is paid immediately. When the credit for avoiding a future system failure must be propagated backward through hundreds of single-step updates, the discounted contribution of the terminal penalty becomes negligible relative to the immediate maintenance cost, and the greedy policy learns to avoid all interventions. This pathology is well-known for value-based RL with sparse, long-horizon rewards and factored credit assignment.

To circumvent this difficulty, we restrict the policy to a compact parameterized class and optimize it directly at the episode ² level using the CEM, which evaluates each candidate policy by its total cost over complete simulation episodes and is therefore immune to the credit-assignment problem. Here, although the learned maintenance policy may exhibit a simpler structure, this parameterization enables a direct and transparent explanation of the optimization results. We restrict the agent to a compact, human-interpretable policy class parameterized by a vector $\theta = (\theta_1, \dots, \theta_n) \in [0, 1.5]^n$. Given observation $o_t = (\ell_{i,t}, c_{i,t}/\text{maxRUL}_i)_{i=1}^n$, the action for component i is:

$$\pi_\theta(o_t)_i = \begin{cases} \text{replace}_i & \text{if } \ell_{i,t} = \text{faulty}_i, \\ \text{beginRep}_i & \text{if } \ell_{i,t} = \text{faultless}_i, \frac{c_{i,t}}{\text{maxRUL}_i} \geq \theta_i, \\ & \text{and } \forall \mathcal{K}_j \ni i : (\mathcal{K}_j \setminus \{i\}) \not\subseteq \text{Down}_t, \\ \text{wait} & \text{otherwise,} \end{cases} \quad (8)$$

where the third condition on the **beginRep** _{i} branch is the *subsystem-safe* predicate: it blocks any preventive action that would complete a minimal cut set and immediately trigger a system failure. Formally, it requires that, for every cut set $\mathcal{K}_j$ containing i , at least one component of $\mathcal{K}_j$ other than i is still available (not in Down_t). The policy has only n continuous parameters, one per component, interpretable as an age-fraction threshold: component i is sent for preventive repair as soon as its normalized degradation $c_{i,t}/\text{maxRUL}_i$ reaches θ_i , provided the action is architecturally safe. By construction, the policy provably never triggers a preventive action that would by itself cause a system failure.

The search domain is clipped to $[0, 1.5]^n$ rather than $[0, 1]^n$. Any $\theta_i > 1$ encodes the “never trigger preventive maintenance on component i ” sub-policy: the invariant $c_{i,t} \leq \text{maxRUL}_i$ on location **faultless** _{i} (Figure 2) prevents $c_{i,t}/\text{maxRUL}_i$ from ever exceeding 1 while $\mathcal{G}_i$ remains in **faultless** _{i} , so the **beginRep** _{i} branch is never reached. Allowing values above 1 lets CEM discover that forgoing preventive maintenance is sub-optimal for components in tight cut sets, while it may be cost-efficient for components protected by abundant k -out-of- n redundancy.

² An episode is a simulation of $\mathcal{S}$ over the fixed horizon T , terminating either at $t = T$ or when $\mathcal{M}$ enters **failed**, whichever comes first.

■ **Algorithm 1** CEM optimization of the threshold policy (8).

Data: DPTGA simulator $\mathcal{S}$; population size N_p ; elite fraction ρ ; number of generations Gen ; episodes per candidate K ; μ_0, σ_0

Result: Optimal threshold vector θ^* .

- 1 Initialise $\mu \leftarrow \mu_0, \sigma \leftarrow \sigma_0$;
- 2 **for** $gen = 1, \dots, Gen$ **do**
- 3 Sample $\theta^{(1)}, \dots, \theta^{(N_p)} \sim \mathcal{N}(\mu, \text{diag}(\sigma^2))$ and clip each entry to $[0, 1.5]$;
- 4 **for** $k = 1, \dots, N_p$ **do**
- 5 $J^{(k)} \leftarrow \frac{1}{K} \sum_{e=1}^K C_{\text{total}}(\pi_{\theta^{(k)}}; \omega_{gen,k,e})$; // avg. cost over K episodes
 with different RUL draws $\omega_{gen,k,e}$
- 6 $Id \leftarrow$ indices of the $\lceil \rho N_p \rceil$ candidates with lowest $J^{(k)}$;
- 7 $\mu \leftarrow \frac{1}{|Id|} \sum_{k \in Id} \theta^{(k)}$;
- 8 $\sigma \leftarrow \text{std}(\{\theta^{(k)}\}_{k \in Id}) + \varepsilon$;
- 9 **return** $\theta^* \leftarrow \mu$;

Because π_θ has only n parameters and the cost landscape is noisy but smooth, we optimize θ with the CEM (Algorithm 1). At each generation gen , CEM samples N_p candidate vectors from a diagonal Gaussian $\mathcal{N}(\mu_g, \text{diag}(\sigma_g^2))$ and evaluates each candidate by its average total cost $J(\theta^{(k)})$ over K simulated DPTGA episodes. Each episode is identified by a random seed $\omega_{g,k,e}$ that encodes all stochastic draws of the simulation – in particular the hidden failure deadlines sampled uniformly in $[\text{minRUL}_i, \text{maxRUL}_i]$ for each component i – so that averaging $C_{\text{total}}(\pi_{\theta^{(k)}}; \omega_{g,k,e})$ over $e = 1, \dots, K$ yields a Monte-Carlo estimate of the expected cost defined in Eq. (7). CEM then retains the $\lceil \rho N_p \rceil$ elite candidates with the lowest cost and refits the Gaussian to these elites. Since the cost is evaluated at the episode level, CEM bypasses the credit-assignment difficulty that defeats step-level value-based RL.

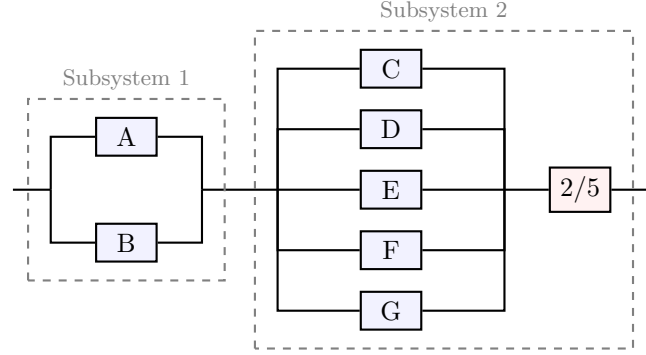
Regarding the scalability of the approach, the temporal complexity of CEM is in $N_p \times Gen \times K$. Note that, as it is a simulation-based algorithm, the simulator $\mathcal{S}$ does not require the explicit computation of the composition $\mathcal{G}_1 \parallel \dots \parallel \mathcal{G}_n \parallel \mathcal{M}$ but only simulates some runs from it. It means that the space complexity of the algorithm is linear with the number of components n . The quality of the result is then based on a proper selection of N_p and Gen that depends on the simulation space defined by $\mathcal{S}$.

6 Case Study and Results

6.1 Experimental Setup

We consider a multi-component system comprising seven distinct components, denoted A through G . The functional architecture of the system is shown in Figure 4. The system fails as soon as either subsystem 1 (A AND B down) or subsystem 2 (any 4 of 5 components in $\{C, D, E, F, G\}$ down) is realized. These failure conditions induce the system minima cut sets which can be extracted with several classical algorithms as explained in Section 3.2. The component parameters used in this case study (RUL bounds, maintenance durations and cost coefficients) are reported in Table 1.

The CEM-trained policy is benchmarked against three traditional strategies. *Corrective* maintenance triggers a replacement only after a component has failed. *Periodic* maintenance triggers a preventive intervention round-robin over the components every 60 time units



■ **Figure 4** Functional architecture of the studied system.

■ **Table 1** Component parameters.

ID	minRUL	maxRUL	C_{comp}	t_{rep}	t_{repl}	$C_{\text{rep_rate}}$	$C_{\text{rep_fix}}$	$C_{\text{repl_fix}}$
A	120	200	200	10	15	30	50	100
B	130	170	180	5	8	20	40	80
C	120	250	190	8	12	25	45	90
D	100	180	110	6	10	15	30	60
E	150	220	120	7	11	20	35	70
F	130	200	120	7	11	20	35	70
G	90	140	90	6	10	15	30	60

(that is an empirically chosen interval intended to reflect the cost impact of systematic maintenance), regardless of degradation level. *Condition-based* maintenance is a threshold heuristic that triggers a preventive intervention as soon as $c_i \geq \text{minRUL}_i$.

The time horizon is $T = 1200$, the early-failure threshold $T_{\text{ef}} = 1000$, the base failure penalty 10 000 with an additional 10 000 when the system fails before T_{ef} . The CEM agent is trained with $Gen = 20$ generations, a population of $N_p = 20$, an evaluation budget of $K = 10$ episodes per candidate, and elite fraction $\rho = 0.25$. Initial parameters are $\mu_0 = 0.7 \mathbf{1}_n$ and $\sigma_0 = 0.25 \mathbf{1}_n$. Training takes ~ 72 s on a single CPU core, including 4 000 stochastic episodes. The four strategies are evaluated on a fresh batch of 800 independent episodes sharing the same seed offset to remove between-policy variance (2 000 episodes for the final reported numbers). Costs are reported as means, standard deviations, medians and 95th percentiles; survival is the fraction of episodes in which the system remained operational over the entire horizon.

6.2 Performance Comparison

Table 2 reports the headline metrics. Three observations stand out. First, the corrective and periodic baselines incur both a higher mean cost and a much lower survival rate ($\approx 40\%$): under the calibration of Section 4 the catastrophic failure penalty dominates the cost budget once the system fails before the horizon, and neither corrective nor a 60-step periodic schedule provides enough preventive throughput to keep the seven components inside their joint safety envelope. Second, the condition-based policy, in the current setting, is the safe risk-averse strategy: it achieves 100% survival at a deterministic cumulative cost of 16,135, because triggering preventive maintenance at $c_i < \text{minRUL}_i$ guarantees that no component ever

■ **Table 2** Comparison of maintenance strategies (2000 evaluation episodes; horizon $T=1200$).

Strategy	Mean cost	Std	Median	P95	Surv. (%)
Corrective	24 176.3	10 583.3	30 593.2	38 002.1	38.3
Periodic	26 625.4	10 745.5	32 497.2	40 612.9	42.4
Condition-based	16 134.8	0.0	16 134.8	16 134.8	100.0
RL (CEM)	13 793.2	2 476.0	13 537.1	14 104.7	98.7

reaches its failure window. It should be noted, however, that this guarantee breaks down when all components belonging to a minimal cut share the same minRUL value: in that specific configuration (that we avoid intentionally), their preventive actions are scheduled simultaneously, the system crosses a failure cut before any intervention can be achieved, and the survival rate of this policy collapses to 0%.

Third, the CEM-trained threshold policy attains a mean cost of 13 793 and a *median* cost of 13 537 at 98.7% survival, that is, a 15% improvement of the mean cost and a 16% improvement of the median cost over the condition-based heuristic. The improvement stems from the agent exploiting the architectural redundancies of the system: in subsystem 2 where two of five components must remain operational, the agent entirely forgoes preventive maintenance on component F (whose learned threshold exceeds 1, see Section 6.3) while keeping tight schedules for the components that belong to smaller, more dangerous cut sets. Crucially, the residual failure rate is only 1.3%, and the 95th-percentile cost of 14 105 is now close to the median, confirming that the policy essentially eliminates the heavy-tail risk that would be present in more aggressive parameterizations. The cost–survival trade-off is therefore *Pareto*: condition-based is the conservative pole, RL the cost-efficient pole, and the practitioner can pick one or the other depending on how the failure penalty is weighted by the operating context. Indeed, for safety-critical applications, a practitioner might increase the failure penalty (C_{SF}) to steer the CEM policy toward 100% survival. However, within the current policy class, this parameter heavily penalizes failure but cannot strictly guarantee survival. A practical solution is a hybrid strategy where the condition-based heuristic acts as a strict safety fallback: the CEM policy optimizes costs when degradation is low, and the heuristic overrides it once degradation approaches critical thresholds to ensure survival. Figure 5 shows the corresponding cost distributions, and Figure 6 shows the cost–survival trade-off across the four strategies.

Figure 6 summarises the cost–survival trade-off: the CEM policy achieves the lowest expected cost among the evaluated strategies, whereas the condition-based policy provides the conservative point with perfect survival. The convergence plot of Figure 7 confirms that CEM reaches its plateau within twenty generations, well within the training budget reported above.

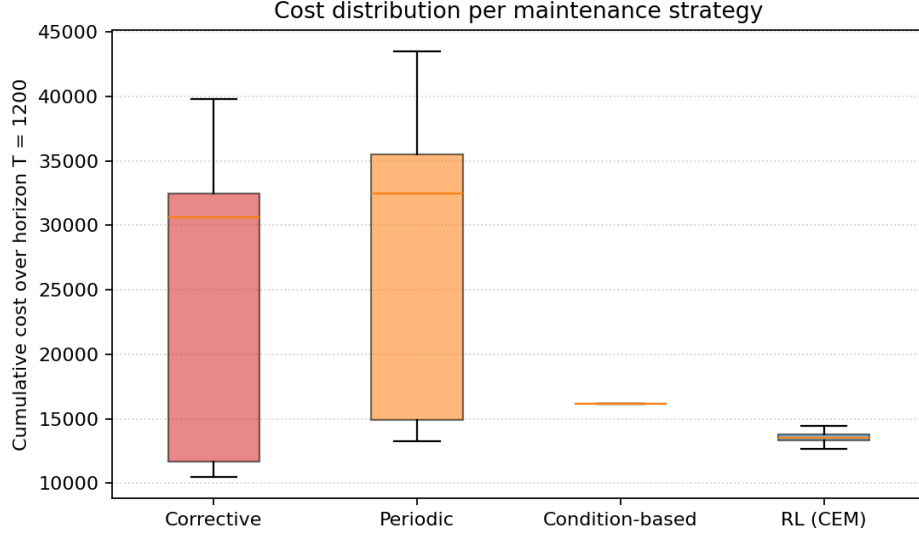
6.3 Policy Interpretation and Validation

The learned threshold vector is

$$\theta^* \approx (0.60, 0.75, 0.88, 0.81, 0.81, 1.03, 0.88), \quad (9)$$

which translates into the per-component preventive triggers reported in Table 3. Three observations:

- Component F receives the highest threshold ($\theta_F^* = 1.03 > 1$), which lies above the domain ceiling: the invariant $c_F \leq \max\text{RUL}_F$ forces $\mathcal{A}_F$ out of the faultless location before $c_F/\max\text{RUL}_F$ can reach 1.03, so preventive maintenance on F is *never* triggered.



■ **Figure 5** Cumulative-cost summary over the horizon $T = 1200$. Markers show the mean and median costs reported in Table 2; vertical intervals show the empirical standard deviation and the 95th percentile.

■ **Table 3** Learned per-component preventive triggers ($\text{age-trigger}_i = \theta_i^* \cdot \text{maxRUL}_i$).

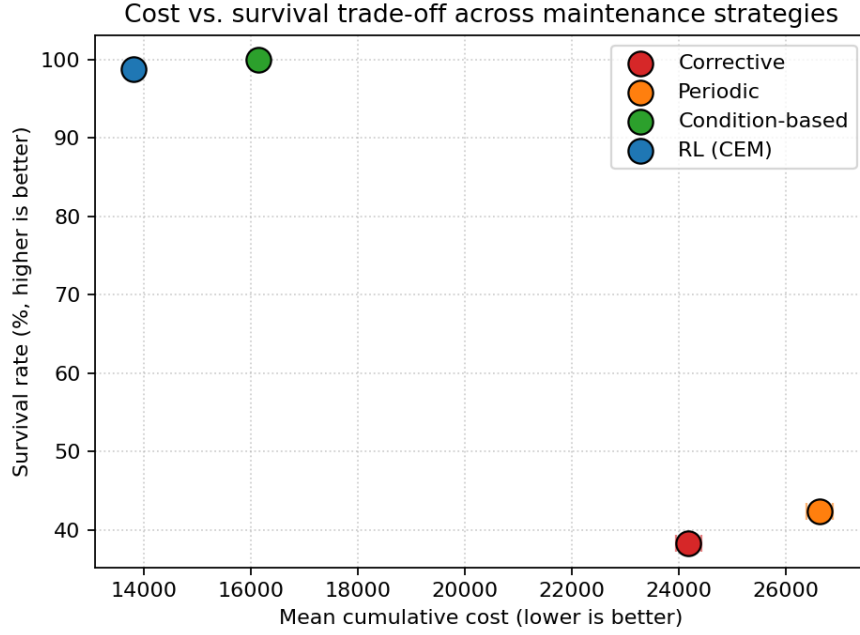
Component	θ_i^*	maxRUL_i	age-trigger_i	minRUL_i
A	0.60	200	120	120
B	0.75	170	128	130
C	0.88	250	220	120
D	0.81	180	146	100
E	0.81	220	178	150
F	1.03	200	>200	130
G	0.88	140	123	90

Because F belongs only to the larger four-component cuts of subsystem 2, the k -out-of- n redundancy makes proactive intervention on F unnecessary: allowing it to fail correctively is cheaper than the maintenance overhead it would incur.

- The remaining components (C , D , E , G) receive thresholds in the range 0.81–0.88, all above their respective $\text{minRUL}_i/\text{maxRUL}_i$ ratios. The policy therefore consistently delays preventive maintenance beyond the local condition-based trigger whenever the subsystem retains sufficient spare redundancy.

The policy is *architecture-aware*: it does not apply the same margin to all components. It anticipates maintenance on the tight $\{A, B\}$ cut set, entirely skips preventive maintenance on F thanks to the k -out-of- n redundancy, and allows the remaining components in subsystem 2 to age beyond the local condition-based trigger when the subsystem still has sufficient spare capacity.

To rule out an implementation artefact, the simulator was instrumented to log every preventive trigger fired by the CEM policy across 50 independent episodes (a total of 60 000 time steps). The *subsystem-safe* predicate of Eq. (8) was satisfied at every single firing: the



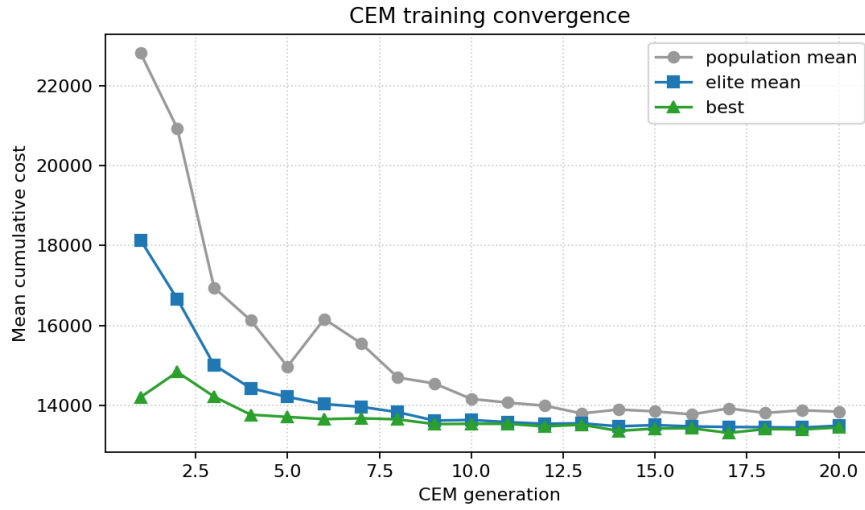
■ **Figure 6** Cost–survival trade-off across the four strategies. The CEM policy strictly dominates the corrective and periodic baselines, and forms a Pareto frontier with the condition-based heuristic.

agent never triggered a preventive maintenance that would itself bring the system below the failure boundary. The 1.3% residual failure rate is therefore due solely to the stochastic RUL sampling – adverse draws in which two components of the same minimal cut set fail before either of them has reached its CEM threshold – not to a violation of the designed safety constraint.

The deterministic cost incurred by the condition-based policy was cross-checked against the closed-form per-component cycle cost predicted by Eq. (3). The analytical sum of $\sim 16\,135$ matches the empirical 16 135 to within 0.1%, confirming that the updated cost parameters are mutually consistent. The simulator and the cost model are therefore in agreement. Overall, the CEM policy leverages the architectural redundancies revealed by the minimal cut sets in a way that the purely local condition-based rule cannot: components belonging to the tightest cuts trigger preventive maintenance earlier (or before their failure window opens, as with component *B*), while components shared by many large cuts in the k -out-of- n block are either allowed to age further before any intervention or, in the extreme case of component *F*, left to fail correctively because the subsystem redundancy makes proactive maintenance economically sub-optimal.

7 Conclusion and perspectives

We presented a full pipeline that synthesizes a predictive maintenance policy for a multi-component system from its functional architecture. Components and the System Health Monitor are modeled as a network of priced timed game automata, and the model semantics is reproduced verbatim by a lightweight explicit-state simulator that bypasses the state-space explosion of a model-checking back-end. On a seven-component case study, this enables a population-based policy search (CEM) over a compact, interpretable threshold policy with



■ **Figure 7** Cross-Entropy Method training curve. The mean of the elite population converges within twenty generations, confirming that the seven-dimensional threshold space is well-conditioned for population-based search.

one parameter per component. The learned policy reaches a mean cumulative cost of 13 793 and a median of 13 537 at a 98.7% survival rate, improving on the best rule-based strategy by 15% in mean cost and 16% in median cost, and strictly dominating the corrective and periodic baselines on both cost and reliability. The 95th-percentile cost of 14 105 confirms that the policy nearly eliminates the heavy-tail failure risk. A key interpretive finding is that the agent learns to forgo preventive maintenance entirely on component F , relying instead on the k -out-of- n redundancy of subsystem 2, while maintaining tight schedules for the more critical cut-set members. The condition-based heuristic and the CEM policy form a Pareto frontier in the cost–survival plane, leaving the practitioner the freedom to pick the operating point that best fits the failure-cost ratio of the asset under management.

This work is preliminary and opens several directions for future research. i) The current cost model assumes that the residual value of a component decreases linearly with its degradation clock and that every maintenance action restores the component to an as-good-as-new state. Both assumptions are standard but debatable. On the one hand, alternative economic depreciation models (declining-balance, sum-of-years-digits) could better capture the non-linear loss of asset value in practice. On the other hand, the hypothesis of perfect maintenance could be relaxed, for instance, by adopting the Kijima model [13], in which a repair of type I reduces the effective age of the component by a fraction of the elapsed degradation, or a repair of type II resets the age of the entire system. Embedding a Kijima-type imperfect maintenance model into the DPTGA transitions would make the cost structure richer and the learned policy more realistic. ii) The DPTGA network introduced in Section 4 has not yet been studied from a formal verification standpoint. Questions of decidability (is the optimal maintenance strategy computable?) and complexity (what is the state-space growth with the number of components and clocks?) remain open and the statistical evidence reported in Section 6 should be complemented. iii) In the present work, the RUL intervals $[\text{minRUL}_i, \text{maxRUL}_i]$ are fixed parameters shared across all episodes. Real PHM pipelines, however, produce predictions whose uncertainty varies over time: the interval narrows as the component ages and the prognostic model accumulates observations. Given

the computational efficiency of the CEM approach, the optimization could simply be re-run at each prognostic update. Alternatively, assuming successive prognostic predictions remain relatively stable, the existing policy could be efficiently fine-tuned rather than recalculated from scratch. Another extension is to train a deep reinforcement learning agent that receives the current RUL interval as part of its state, so that the policy can adapt online to the instantaneous quality of the predictions. This would replace the static threshold class of Section 5 with a neural controller capable of exploiting time-varying prognostic confidence, at the cost of interpretability.

References

- 1 Mads Kronborg Agesen, Søren Enevoldsen, Thibaut Le Guilly, Anders Mariegaard, Petur Olsen, and Arne Skou. *Energy Consumption Forecast of Photo-Voltaic Comfort Cooling Using UPPAAL Stratego*, pages 603–622. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-63121-9_30.
- 2 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994. doi:10.1016/0304-3975(94)90010-8.
- 3 Charalampos P. Andriotis and Kostas G. Papakonstantinou. Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints. *Reliability Engineering & System Safety*, 212:107551, 2021. doi:10.1016/j.ress.2021.107551.
- 4 Vepa Atamuradov, Kamal Medjaher, Pierre Dersin, Benjamin Lamoureux, and Nouredine Zerhouni. Prognostics and health management for maintenance practitioners-review, implementation and tools evaluation. *International Journal of Prognostics and Health Management*, 8(3):1–31, 2017. doi:10.36001/ijphm.2017.v8i3.2667.
- 5 Alexandre David, Peter Gjør Jensen, Kim G. Larsen, Marius Mikuvccionis, and Jakob H. Taankvist. Uppaal Stratego. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 9035 of *LNCS*, pages 206–211. Springer, 2015. doi:10.1007/978-3-662-46681-0_16.
- 6 Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikuvccionis, and Danny B. Poulsen. Uppaal SMC tutorial. *Int. J. Software Tools for Technology Transfer*, 17(4):397–415, 2015. doi:10.1007/s10009-014-0361-y.
- 7 Phuc Do, Van-Thai Nguyen, Alexandre Voisin, Benoit Iung, and Waldomiro Alves Ferreira Neto. Multi-agent deep reinforcement learning-based maintenance optimization for multi-dependent component systems. *Expert Syst. Appl.*, 245(C), July 2024. doi:10.1016/j.eswa.2024.123144.
- 8 JB Fussell, EB Henry, and NH Marshall. Mocus: A computer program to obtain minimal sets from fault trees. Technical report, Aerojet Nuclear Co., Idaho Falls, ID (United States), 1974. URL: <https://digital.library.unt.edu/ark:/67531/metadc1016681>.
- 9 Martijn A. Goorden, Peter Gjør Jensen, Kim G. Larsen, Andrey Samusev, Jiri Srba, and Penut Vongmasa. STOMPC: Stochastic model-predictive control with Uppaal Stratego. In *Automated Technology for Verification and Analysis (ATVA)*, volume 13505 of *LNCS*. Springer, 2022. doi:10.1007/978-3-031-19992-9_21.
- 10 Iwona Grobelna, Krystian Gajewski, and Andrei Karatkevich. A systematic review on the applications of uppaal. *Sensors*, 25(11):3484, 2025. doi:10.3390/s25113484.
- 11 Zhiyi Huang and Xiao Li. A critical review of operations research on the operation and maintenance of railway systems. *Journal of Railway Science and Technology*, 1(2):47–58, 2025. doi:10.1016/j.jrst.2025.08.001.
- 12 Manfred Jaeger and Kim Guldstrand Larsen. Reinforcement learning for discretized euclidean mdps. In Bernhard Steffen, editor, *Bridging the Gap Between AI and Reality*, pages 312–335, Cham, 2025. Springer Nature Switzerland. doi:10.1007/978-3-031-75434-0_22.
- 13 Masaaki Kijima. Some results for repairable systems with general repair. *Journal of Applied Probability*, 26(1):89–102, 1989. doi:10.2307/3214319.

- 14 Rajesh Kumar, Bhavesh Narra, Rohan Kela, and Siddhant Singh. Afmt: Maintaining the safety-security of industrial control systems. *Computers in Industry*, 136:103584, 2022. doi:10.1016/j.compind.2021.103584.
- 15 Rajesh Kumar, Enno Ruijters, and Mariëlle Stoelinga. Quantitative attack tree analysis via priced timed automata. In *Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 9268 of *LNCS*, pages 156–171. Springer, 2015. doi:10.1007/978-3-319-22975-1_11.
- 16 Melih Cemal Kushan and Seyid Fehmi Diltemiz. Aircraft health monitoring systems (ahms) applications in aviation. In Melih Cemal Kuşhan and Seyid Fehmi Diltemiz, editors, *Aircraft Manufacturing, Safety and Control*, chapter 5. IntechOpen, London, 2025. doi:10.5772/intechopen.1013417.
- 17 Kim G. Larsen, Marius Mikuvccionis, Marco Muniz, Jiri Srba, and Jakob H. Taankvist. Online and compositional learning of controllers with application to floor heating. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 9636 of *LNCS*, pages 244–259. Springer, 2016. doi:10.1007/978-3-662-49674-9_14.
- 18 Wen-Shing Lee, Doris L Grosh, Frank A Tillman, and Chang H Lie. Fault tree analysis, methods, and applications a review. *IEEE transactions on reliability*, 34(3):194–203, 2009. doi:10.1109/TR.1985.5222114.
- 19 Yu Liu, Yuanjian Chen, and Tao Jiang. A deep reinforcement learning approach for maintenance planning of multi-component systems with complex structure. *Neural Computing and Applications*, 35(26):19625–19642, 2023. doi:10.1007/s00521-023-08542-9.
- 20 Alberto Pliego Marugán. Applications of reinforcement learning for maintenance of engineering systems: a review. *Advances in Engineering Software*, 183, 2023. doi:10.1016/j.advengsoft.2023.103487.
- 21 Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. doi:10.1038/nature14236.
- 22 Lars Mönch, John W. Fowler, and Scott J. Mason. *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*. Springer, 2013. doi:10.1007/978-1-4614-4472-5.
- 23 Saeed Najafi and Chi-Guhn Lee. A deep reinforcement learning approach for repair-based maintenance of multi-unit systems using proportional hazards model. *Reliability Engineering & System Safety*, 234:109179, 2023. doi:10.1016/j.ress.2023.109179.
- 24 Oluwaseyi Ogunfowora and Homayoun Najjaran. Reinforcement and deep reinforcement learning-based solutions for machine maintenance planning, scheduling policies, and optimization. *Journal of Manufacturing Systems*, 70:244–263, 2023. doi:10.1016/j.jmsy.2023.07.014.
- 25 Luca Pincirolì, Piero Baraldi, Guido Ballabio, Michele Compare, and Enrico Zio. Optimization of the operation and maintenance of renewable energy systems by deep reinforcement learning. *Renewable Energy*, 183:752–763, 2022. doi:10.1016/j.renene.2021.11.052.
- 26 Luca Pincirolì, Piero Baraldi, Michele Compare, and Enrico Zio. Deep reinforcement learning based on proximal policy optimization for the maintenance of a wind farm with multiple crews. *Energies*, 14(20):6743, 2021. doi:10.3390/en14206743.
- 27 Antoine Rauzy. New algorithms for fault trees analysis. *Reliability Engineering & System Safety*, 40(3):203–211, 1993. doi:10.1016/0951-8320(93)90060-C.
- 28 Alberto Regattieri, A Giazzi, Mauro Gamberi, and Rita Gamberini. An innovative method to optimize the maintenance policies in an aircraft: General framework and case study. *Journal of air transport management*, 44:8–20, 2015. doi:10.1016/j.jairtraman.2015.02.001.
- 29 Pauline Ribot, Yannick Pencolé, and Michel Combacau. Generic characterization of diagnosis and prognosis for complex heterogeneous systems. *International Journal of Prognostics and Health Management*, 4(2), 2013. doi:10.36001/ijphm.2013.v4i2.2120.

- 30 Michael Rubin and Dongping Du. A review of dynamic fault tree analysis and capacity degradation for complex redundant systems. *Quality and Reliability Engineering International*, 41(3):1161–1181, 2025. doi:10.1002/qre.3693.
- 31 Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004. doi:10.1007/978-1-4757-4321-0.
- 32 Enno Ruijters, Dennis Guck, Peter van Noort, and Mariëlle Stoelinga. Modelling smart buildings using fault maintenance trees. In *Computer Performance Engineering (EPEW)*, volume 11178 of *LNCS*, pages 110–125. Springer, 2018. doi:10.1007/978-3-030-02227-3_8.
- 33 Enno Ruijters, Daniel Reijbergen, Pieter-Tjerk de Boer, and Mariëlle Stoelinga. Better railway engineering through statistical model checking. In *Int. Symp. on Leveraging Applications of Formal Methods (ISoLA)*, volume 9952 of *LNCS*, pages 151–165. Springer, 2016. doi:10.1007/978-3-319-47166-2_10.
- 34 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, 2017. doi:10.48550/arXiv.1707.06347.
- 35 Ferhat Tamssaouet, Khanh Tp Nguyen, Kamal Medjaher, and Marcos Eduardo Orchard. System-level failure prognostics: Literature review and main challenges. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 237(3):524–545, 2023. doi:10.1177/1748006X221118448.
- 36 Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992. doi:10.1007/BF00992698.
- 37 Enrico Zio. Prognostics and health management (phm): Where are we and where do we (need to) go in theory and practice. *Reliability Engineering & System Safety*, 218:108119, 2022. doi:10.1016/j.ress.2021.108119.

Statistical Foundations of Local Geometric Fault Detection: The Curvature Channel

Gregory Provan 

School of Computer Science and IT, University College Cork, Ireland

Abstract

Geometric and topological methods have recently emerged as a promising approach to fault detection in hybrid and cyber-physical systems. Unlike classical residual-based methods, which detect violations of a parametric model, geometric methods monitor changes in the local structure of trajectories through quantities such as intrinsic dimension, smoothness, tangent-cone geometry, and consistency with physical constraints. Despite encouraging empirical results, there is currently no theory characterising when such geometric signatures are detectable, distinguishable, or reliable under noise. This paper develops a statistical theory of local geometric fault detection, with a focus on curvature-based approaches. We study faults detectable by displacement in a curvature signature space, and derive the noise floor of geometric estimators based on finite differences and local neighbourhood averaging. The resulting estimator variance scales as $\sigma_{\text{est}} = O(\sigma/(\Delta t^2 \sqrt{k}))$, exposing the fundamental trade-off between sampling rate, noise level, and neighbourhood size. Using this noise model, we derive geometric analogues of the classical detectability and isolability conditions of fault-detection theory, expressed directly in terms of separation in signature space. We further prove an exclusion result showing that a broad class of smooth parametric faults cannot activate curvature-based geometric channels, yielding a form of label-free fault-class identification. These results lead to a practical pre-deployment assessment procedure that determines, from nominal data alone, whether a curvature-based fault detector is expected to achieve a specified false-alarm rate, detection probability, and isolation capability. The theory thereby places local geometric fault detection on the same statistical footing as classical residual-based diagnosis while clarifying its distinctive strengths, limitations, and operating regime.

2012 ACM Subject Classification Computing methodologies → Causal reasoning and diagnostics

Keywords and phrases Geometric metrics, Diagnosis methods, Theory

Digital Object Identifier 10.4230/OASICS.DX.2026.13

Funding Funded by Research Ireland grant 13/RC/2094_P2.

1 Introduction

Fault detection has traditionally been formulated as the problem of identifying statistically significant deviations from a nominal model. Classical approaches include analytical redundancy relations, observer-based residual generation, parity-space methods, generalized likelihood ratio tests, and Bayesian state-estimation techniques. Although these methods differ in representation and implementation, they share a common principle: fault information is extracted through a residual measuring disagreement between observations and a parametric model of the system.

Over the last decade, a different view has begun to emerge. Rather than monitoring model mismatch, one may monitor the *geometry* of the observed trajectory itself. Hybrid systems, switched systems, and many cyber-physical systems naturally generate trajectories that lie on unions of locally smooth manifolds with boundaries, intersections, and singularities. Structural faults can alter this geometry even when local algebraic relationships remain approximately satisfied [4, 8]. Examples include mode-creation and mode-deletion events, stick-slip transitions, changing guard conditions, loss of smoothness, and changes



© Gregory Provan;

licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 13; pp. 13:1–13:20

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

in accessibility between operating regimes. Such phenomena motivate geometric diagnostic quantities including intrinsic dimension, manifold smoothness, tangent-cone structure, and consistency of local physical constraints.

Recent work has demonstrated that these geometric quantities can be estimated from sampled trajectories and can reveal classes of faults that are difficult to detect using traditional residuals. However, the diagnostic literature currently lacks answers to three fundamental questions.

1. **Detectability.** When does a fault produce a sufficiently large change in geometric signature to be distinguished from estimator noise?
2. **Isolability.** When are two fault classes sufficiently separated in geometric signature space to permit reliable classification?
3. **Feasibility.** Can these questions be answered before deployment, using only nominal data and a proposed fault taxonomy?

The absence of such results makes it difficult to design geometric detectors systematically. In practice, neighbourhood size, sampling rate, and detection thresholds are often chosen heuristically. Consequently, success or failure of a geometric detector may be determined as much by implementation choices as by the underlying physical system.

This paper develops a statistical theory of local geometric fault detection. Our starting point is a geometric signature vector $\Phi(\varphi) = (d_{\text{eff}}, \kappa_{\text{gen}}, \kappa_{\text{OR}}, C)$, whose components measure local dimension, smoothness, tangent-cone structure, and semantic consistency. A fault is represented by the displacement it induces in this signature space. The central observation is that geometric estimators are simultaneously affected by two opposing mechanisms: finite differencing amplifies observation noise, whereas local averaging suppresses it. Quantifying this balance yields an explicit estimator noise scale that becomes the fundamental quantity governing all subsequent results.

Building on this noise model, we derive geometric analogues of the classical detectability and isolability conditions from the fault diagnosis literature. The resulting criteria are expressed as simple separation conditions in signature space and yield a geometric signal-to-noise ratio interpretation analogous to that used in residual-based diagnosis. We then establish an exclusion theorem showing that a broad class of smooth parametric faults cannot activate curvature-based geometric channels. This result enables a limited but rigorous form of label-free fault-class identification: certain geometric signatures can exclude entire classes of faults without requiring fault examples or a dictionary of fault patterns.

The theoretical results lead directly to a practical contribution: a Geometric Feasibility Assessment procedure that determines whether a proposed detector is expected to achieve a specified false-alarm rate, power, and class separability. The procedure requires only nominal data and therefore provides a pre-deployment certification mechanism for geometric fault detection.

The paper does not propose a new fault-detection algorithm. Instead, it develops the statistical foundations needed to analyse existing and future geometric detectors in the same way that classical detectability and isolability theory analyses residual-based methods. Our aim is to clarify the operating regime of geometric fault detection, identify the conditions under which it can be expected to succeed, and establish a common language for comparing geometric and classical diagnostic approaches.

Contributions. The main contributions are:

1. A derivation of the noise floor of local geometric estimators, showing

$$\sigma_{\text{est}} = O\left(\frac{\sigma}{\Delta t^2 \sqrt{k}}\right).$$

2. A geometric detectability theorem relating fault detectability to separation in geometric signature space.
3. A geometric isolability theorem characterising when fault classes are distinguishable from geometric observations alone.
4. An exclusion theorem showing that smooth parametric faults cannot activate curvature-based geometric channels.
5. A Geometric Feasibility Assessment procedure that translates the theory into a practical pre-deployment design workflow.

2 Preliminaries

This paper studies fault detection using local geometric properties of measured trajectories. Before introducing detectability and isolability results, we define the observation model, geometric measurement field, and signature representation used throughout.

2.1 Hybrid-System Observations

Let $x(t) \in \mathcal{X}$ denote the state trajectory of a dynamical system. We assume observations are available through a sensor map $O : \mathcal{X} \rightarrow \mathbb{R}^p$, giving measurements $y(t) = O(x(t)) + \epsilon(t)$, where $\epsilon(t) \sim \mathcal{N}(0, \Sigma)$ is observation noise.

In classical fault detection the diagnostic signal is typically a residual $r(t) = y(t) - \hat{y}(t)$, constructed from a model prediction. The geometric approach considered here replaces residuals by local geometric descriptors extracted directly from the observed trajectory.

2.2 Local Neighbourhoods

Let $y_t := y(t)$ denote a sampled observation. For each sample we define a local neighbourhood $N_k(y_t) = \{y_{t_1}, \dots, y_{t_k}\}$, consisting of the k nearest observations under a chosen distance metric.

All geometric quantities in this paper are computed from such neighbourhoods. Consequently the approach is local and does not require a global model of the state space.

2.3 Geometric Measurement Fields

► **Definition 1** (Geometric Measurement Field). *A geometric measurement field is a map $M : \mathbb{R}^p \rightarrow \mathbb{R}^r$ that converts a local neighbourhood into a collection of geometric descriptors.*

The canonical field used in this work is

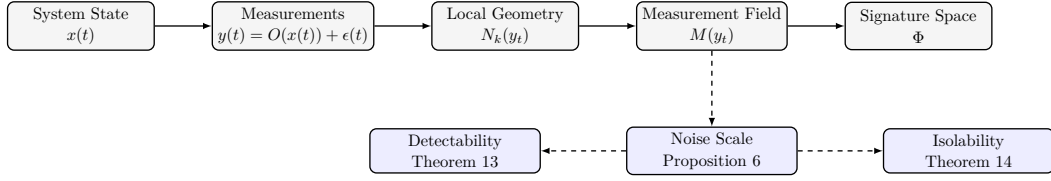
$$M(y_t) = (d_{\text{eff}}(y_t), \kappa_{\text{gen}}(y_t), \kappa_{\text{OR}}(y_t), C(y_t)),$$

consisting of:

- d_{eff} : local intrinsic dimension estimated from PCA eigenvalue structure;
- κ_{gen} : smoothness or curvature residual measuring departure from a locally smooth chart;
- κ_{OR} : tangent-cone splitting statistic measuring local multi-directionality;
- C : semantic-consistency score measuring violation of local physical laws or constraints.

These quantities define a geometric analogue of the residual vector used in classical fault diagnosis.

13:4 Local Geometric Fault Detection



■ **Figure 1** Conceptual framework of local geometric fault detection. The observed trajectory is transformed into a local geometric signature. Proposition 6 characterises the estimator noise floor, which in turn determines the detectability and isolability conditions of Theorems 13 and 14.

2.4 Geometric Signature Space

The output of the measurement field is called the *geometric signature*:

$$\Phi = (d_{\text{eff}}, \kappa_{\text{gen}}, \kappa_{\text{OR}}, C) \in S,$$

where $S \subseteq \mathbb{R}^4$ is the signature space.

A nominal operating condition corresponds to a region $\Phi(\nu) \in S$, while a fault φ induces a displaced signature $\Phi(\varphi)$.

The central question of this paper is the *geometric detectability problem*:

► **Definition 2** (Geometric detectability problem). *When is the displacement $\|\Phi(\varphi) - \Phi(\nu)\|$ large enough to be distinguished from estimator noise?*

2.5 Fault Classes

We distinguish between two broad categories of faults.

► **Definition 3** (Parametric fault). *A parametric fault modifies parameters of the underlying dynamics while preserving the qualitative geometry of the trajectory. Examples include coefficient drift, gain changes, and smooth degradation processes.*

► **Definition 4** (Structural fault). *A structural fault changes the local geometry of the trajectory, such as its smoothness, dimension, accessibility structure, or constraint consistency. Examples include stick-slip transitions, mode creation, guard perturbations, and topology-changing events.*

The purpose of the geometric measurement field is to detect structural faults through changes in signature space.

2.6 From Observations to Decisions

The complete information flow studied in this paper is

$$x(t) \rightarrow y(t) \rightarrow N_k(y_t) \rightarrow M(y_t) \rightarrow \Phi(y_t) \rightarrow \text{fault decision}.$$

Sections 3–5 analyse the statistical properties of this pipeline and derive conditions under which faults are detectable and distinguishable.

3 Geometric Detectability and Isolability

This section states the condition under which the geometric approach succeeds, proves it, relates it to the classical parametric results it generalises, and turns it into a decision procedure a practitioner can run *before* deployment. We are explicit at the outset about

what is and is not new: the condition is the *geometric instance* of a well-established family of detectability/isolability results [1, 2, 3, 5]. Its novelty lies not in its form but in the *signal* it applies to (a geometric invariant rather than a model residual) and in the *specific noise scale* that differentiation and local averaging induce. We denote the geometric approach defined in this section as Geometric Model-Based Diagnosis (GMBD).

3.1 Setup

We define a cause-to-symptom map using $\Phi_M = O_M \circ (\text{physics}) : \mathcal{C} \rightarrow \mathcal{S}_M$, with ν the nominal cause. For GMBD, O_M is the geometric field and the symptom space is the signature vector

$$\Phi(\phi) = (d_{\text{eff}}, \kappa_{\text{gen}}, \kappa_{\text{OR}}, C)(\phi) \in \mathcal{S} \subseteq \mathbb{R}^4. \quad (1)$$

Observation is noisy, $y = O(x) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \Sigma)$, $\Sigma = \sigma^2 I$, sampled at interval Δt , with the field estimated from neighbourhoods of k samples in d dimensions. Write $\hat{\Phi}$ for the estimated signature.

► **Definition 5** (Estimator noise scale). $\sigma_{\text{est}}(k, \sigma, \Delta t)$ is the standard deviation of $\hat{\Phi}(\nu)$ about its mean on nominal data: the radius of the nominal ball in signature space.

3.2 The noise scale of a geometric estimator

The distinguishing quantitative feature of the geometric approach is that its invariants are *differential*: they are built from finite differences of the observed trajectory, which amplify noise, and from local averages over k neighbours, which suppress it. These two effects compose.

► **Proposition 6** (Geometric noise scale). For the curvature channel κ_{gen} , computed as a windowed second difference over neighbourhoods of k samples,

$$\sigma_{\text{est}}(k, \sigma, \Delta t) = \frac{C \sigma}{\Delta t^2 \sqrt{k}} (1 + o(1)), \quad C = O(1), \quad (2)$$

i.e. linear in the observation noise σ , quadratically amplified by the sampling interval, and suppressed as $k^{-1/2}$ by local averaging.

Proof. Let $y_t = x_t + \epsilon_t$ with $\epsilon_t \stackrel{iid}{\sim} \mathcal{N}(0, \sigma^2 I_d)$. The second difference is $D^2 y_t = (y_{t+1} - 2y_t + y_{t-1})/\Delta t^2$. Its noise component is $(\epsilon_{t+1} - 2\epsilon_t + \epsilon_{t-1})/\Delta t^2$, a linear combination of independent Gaussians with coefficient vector $(1, -2, 1)$, hence

$$\text{Var}[D^2 y_t] = \frac{\sigma^2 \|(1, -2, 1)\|_2^2}{\Delta t^4} = \frac{6 \sigma^2}{\Delta t^4}, \quad (3)$$

so a single second difference has noise standard deviation $\sqrt{6} \sigma / \Delta t^2$: the Δt^{-2} amplification. The estimator averages $\|D^2 y\|$ over a window of k samples. The window terms are weakly dependent (each ϵ_t enters three consecutive second differences), so the effective number of independent terms is k/c_0 for a constant $c_0 \leq 3$; by the central limit theorem for m -dependent sequences the standard error of the window mean is $\sqrt{c_0}$ times that of the independent case, giving standard deviation $\sqrt{6c_0} \sigma / (\Delta t^2 \sqrt{k})$. Setting $C = \sqrt{6c_0}$ yields (2). The $o(1)$ absorbs the curvature of the true trajectory, which contributes a deterministic offset (the signal, not the noise) and the m -dependence correction, both vanishing relative to the leading term as $\sigma/\Delta t^2 \rightarrow \infty$. ◀

► **Remark 7** (Empirical confirmation). Both factors in (2) are confirmed on the Cart–Pole. Sweeping σ over three decades at fixed k , the measured nominal floor satisfies $\kappa_{\text{gen}}^{\text{noise}} \Delta t^2 / \sigma = 2.95 \pm 0.03$, constant as predicted (Fig. 4, left). Sweeping k at fixed σ , the product $\text{sd}(\kappa_{\text{gen}}) \sqrt{k}$ is constant at 127 ± 1 for $k \in [40, 160]$, confirming the $k^{-1/2}$ law. The measured coefficient of variation falls $0.091 \rightarrow 0.047$ as $k : 40 \rightarrow 200$, consistent with $k^{-1/2}$.

3.3 Identifiability of causes: an image/fibre condition

Theorem 14 will give a *statistical* condition for separating fault classes at a given noise level. Before that, it is worth isolating the *information-theoretic* question underneath it: are the classes distinguishable *even in principle*, with noise set aside? The two questions have different answers and different remedies, and conflating them is a common source of confusion – a limit that no estimator can beat is a different object from a limit that more data would fix.

Fix a detector M with observation map O_M and let

$$\Phi_M = O_M \circ (\text{physics}) : \mathcal{C} \longrightarrow \mathcal{S}_M \quad (4)$$

be the induced *cause-to-symptom* map from the space of causes $\mathcal{C}$ (faults, together with the nominal cause ν) to the detector’s symptom space $\mathcal{S}_M$; for the geometric detector, $\mathcal{S}_M$ is the signature space of (1). Let $\sim$ denote the *diagnostic equivalence* on $\mathcal{C}$: $\phi \sim \phi'$ when the application does not require the two causes to be told apart, and write $[\phi]$ for the class of ϕ and $\mathcal{C}/\sim$ for the set of classes. A *decision rule* is a measurable map $\delta : \mathcal{S}_M \rightarrow \mathcal{C}/\sim$, possibly randomised, in which case $\delta(s)$ is a probability distribution on $\mathcal{C}/\sim$.

The following definition names the only structure that matters.

► **Definition 8** (Fibre purity). *The fibre of Φ_M over $s \in \mathcal{S}_M$ is $\Phi_M^{-1}(s) = \{\phi \in \mathcal{C} : \Phi_M(\phi) = s\}$. The fibre is pure if it meets exactly one $\sim$ -class, i.e. if $\phi, \phi' \in \Phi_M^{-1}(s) \Rightarrow \phi \sim \phi'$. The map Φ_M has pure fibres if every fibre over a point of its image $\text{Im } \Phi_M$ is pure.*

Detection and identification are then governed by two *different* features of the same map: detection by its *image*, identification by its *fibres*.

► **Proposition 9** (Detectability = image separation). *In the noiseless setting, a detector M can decide the presence of a fault ϕ from its symptom alone if and only if $\Phi_M(\phi) \neq \Phi_M(\nu)$. Detection is therefore a property of the image of Φ_M : it asks only whether the faulted symptom is distinct from the nominal one.*

Proof. ($\Leftarrow$) If $\Phi_M(\phi) \neq \Phi_M(\nu)$, the rule that reports “fault” exactly on $\mathcal{S}_M \setminus \{\Phi_M(\nu)\}$ decides correctly at ϕ and at ν . ($\Rightarrow$) If $\Phi_M(\phi) = \Phi_M(\nu) =: s$, then any rule δ returns the same verdict $\delta(s)$ under both causes, so it is wrong under at least one of them. ◀

► **Proposition 10** (Identifiability = fibre purity). *Consider the noiseless setting, in which the detector observes $s = \Phi_M(\phi)$ exactly. Then:*

- (i) **(Sufficiency.)** *If Φ_M has pure fibres (Def. 8), there exists a decision rule δ with $\delta(\Phi_M(\phi)) = [\phi]$ for every $\phi \in \mathcal{C}$: the causes are identifiable up to $\sim$.*
- (ii) **(Necessity, and impossibility.)** *If some fibre $\Phi_M^{-1}(s^*)$ meets two distinct classes $[\phi_1] \neq [\phi_2]$, then for every decision rule δ – deterministic or randomised – the worst-case error probability over $\{\phi_1, \phi_2\}$ is at least $1/2$. No rule reading only the symptom can separate the two causes, at any sample size.*

Identification is therefore a property of the fibres of Φ_M .

Proof.

- (i) Suppose every fibre over $\text{Im } \Phi_M$ is pure. For $s \in \text{Im } \Phi_M$ the set $\Phi_M^{-1}(s)$ is nonempty and, by purity, contained in a single class; define $\delta(s)$ to be that class, and define δ arbitrarily off $\text{Im } \Phi_M$ (such symptoms never occur). The map is well defined precisely because purity makes the choice unambiguous, and by construction $\delta(\Phi_M(\phi)) = [\phi]$ for all ϕ .
- (ii) Suppose $\phi_1, \phi_2 \in \Phi_M^{-1}(s^*)$ with $[\phi_1] \neq [\phi_2]$. Both causes produce the *same* symptom s^* , so the rule's output is the same random variable $\delta(s^*)$ in both cases; it cannot depend on which of ϕ_1, ϕ_2 occurred, since the data it reads are identical. Let $q = \Pr[\delta(s^*) = [\phi_1]]$. The error probability is $1 - q$ under ϕ_1 and at least q under ϕ_2 (as $[\phi_1] \neq [\phi_2]$, reporting $[\phi_1]$ is an error under ϕ_2). Hence

$$\max\{\Pr[\text{err} \mid \phi_1], \Pr[\text{err} \mid \phi_2]\} \geq \max\{1 - q, q\} \geq \frac{1}{2},$$

the last step because $\max\{1 - q, q\} \geq \frac{1}{2}$ for every $q \in [0, 1]$. A deterministic rule is the case $q \in \{0, 1\}$, giving error 1 under one of the two causes. The bound involves no sample size: the obstruction is that the observation map has already discarded the distinguishing information, so no amount of data recovers it. ◀

► **Corollary 11** (Detection does not imply identification). *A detector may satisfy Proposition 9 for every fault of interest – so that every fault is detectable – while violating Proposition 10, so that no fault cause is identifiable. Image separation and fibre purity are independent properties.*

Proof. Take $\mathcal{C} = \{\nu, \phi_1, \phi_2\}$ with $[\phi_1] \neq [\phi_2]$ and a map with $\Phi_M(\phi_1) = \Phi_M(\phi_2) = s^* \neq \Phi_M(\nu)$. Every fault is detectable by Proposition 9, since $s^* \neq \Phi_M(\nu)$; but the fibre over s^* straddles two classes, so by Proposition 10(ii) no rule identifies the cause. ◀

Corollary 11 is not a contrivance: it is the generic situation for geometric detectors, and the reason is structural rather than incidental. The geometric field reports *regularity properties of the observed trajectory*. Distinct physical mechanisms routinely share those properties – a seizing friction element and a snapped kinematic link both destroy C^r regularity and so both raise the curvature channel – so their common fibre straddles two cause classes even though each is comfortably separated from nominal. The geometric detector therefore *detects* both and *identifies* neither. Escaping the conclusion requires enlarging O_M – an additional channel (for instance a constraint/law residual, which the snapped link violates and the seizing friction does not), or an external signature-to-cause dictionary. Neither is a better decision rule on the same data; both change the data.

► **Remark 12** (Fibre purity is the $\sigma \rightarrow 0$ limit of isolability). Proposition 10 and Theorem 14 are the noiseless and noisy halves of one statement, and it is useful to see how they compose. Theorem 14 requires class centroids to satisfy $\min_{K \neq K'} \|\Phi(K) - \Phi(K')\| > 2z_{1-\alpha/2} \sigma_{\text{est}}$. Letting $\sigma_{\text{est}} \rightarrow 0$ – by Proposition 6, the joint limit $\sigma \rightarrow 0$ or $k \rightarrow \infty$ – the right-hand side vanishes and the condition degenerates to $\Phi(K) \neq \Phi(K')$, i.e. to fibre purity. So:

- *Fibre purity is necessary but not sufficient* for identification under noise. Pure fibres whose centroids are closer than $2z\sigma_{\text{est}}$ are statistically inseparable at that noise level, even though they are information-theoretically distinct.
- The two failure modes have different remedies. A violation of Theorem 14 with pure fibres is a *statistical* shortfall: raising k shrinks σ_{est} as $k^{-1/2}$ and can restore separability, at a latency cost. A violation of Proposition 10 is an *information-theoretic* obstruction: no k , no sample size, and no estimator improvement helps, because the observation map has already collapsed the classes.

The practical import is that the two must be checked in order. Verifying fibre purity is a modelling question about Φ_M , answered from the fault taxonomy; verifying the isolability margin is a statistical question, answered from nominal data. A detector that fails the first cannot be rescued by the procedure of Section 3.7.

3.4 Detectability and isolability

With the noise scale in hand, the conditions follow from a standard separation-of-balls argument.

► **Theorem 13** (Geometric detectability). *Fix a false-alarm level α and let $z = z_{1-\alpha}$ be the corresponding Gaussian quantile. A fault ϕ is detectable by the geometric field with false-alarm rate at most α and power at least $1 - \beta$ if*

$$\underbrace{\|\Phi(\phi) - \Phi(\nu)\|}_{\text{signature displacement}} > (z_{1-\alpha} + z_{1-\beta}) \sigma_{\text{est}}(k, \sigma, \Delta t). \quad (5)$$

Define the geometric SNR $\text{SNR}_g(\phi) := \|\Phi(\phi) - \Phi(\nu)\|/\sigma_{\text{est}}$; detection succeeds iff SNR_g exceeds the required z -sum.

Proof. Under the nominal cause, $\hat{\Phi}(\nu)$ concentrates about $\Phi(\nu)$ with scale σ_{est} (Prop. 6); a threshold placed at $\Phi(\nu) + z_{1-\alpha}\sigma_{\text{est}}$ therefore has false-alarm rate α . Under ϕ , $\hat{\Phi}(\phi)$ concentrates about $\Phi(\phi)$ with the same scale (the estimator's variance is a property of the noise and window, not of the cause). The test has power $\geq 1 - \beta$ iff the faulted mean exceeds the threshold by at least $z_{1-\beta}\sigma_{\text{est}}$, i.e. iff $\|\Phi(\phi) - \Phi(\nu)\| > (z_{1-\alpha} + z_{1-\beta})\sigma_{\text{est}}$, which is (5). ◀

► **Theorem 14** (Geometric isolability). *Two cause classes $K \neq K'$ are distinguishable by the geometric field at level α if*

$$\min_{K \neq K'} \|\Phi(K) - \Phi(K')\| > 2 z_{1-\alpha/2} \sigma_{\text{est}}(k, \sigma, \Delta t), \quad (6)$$

i.e. the class centroids are separated by more than the sum of their confidence radii. If (6) fails, the classes' fibres overlap at resolution σ_{est} and, by Proposition 10, no decision rule reading only the signature can separate them.

Proof. Each class occupies a ball of radius $z_{1-\alpha/2}\sigma_{\text{est}}$ about its centroid at confidence $1 - \alpha$. Two such balls are disjoint iff the centroid distance exceeds the sum of radii, $2z_{1-\alpha/2}\sigma_{\text{est}}$. Disjointness gives a decision rule (nearest centroid) with error $\leq \alpha$; overlap means some signature is consistent with both classes, so the fibre $\Phi^{-1}(s)$ straddles classes and Proposition 10 applies. ◀

► **Remark 15** (Measured separability). On the Cart–Pole at $\sigma = 10^{-2}$, measured centroid distances divided by pooled within-class spread are: nominal vs. topological 8.11; parametric vs. topological 4.66; nominal vs. parametric 1.81. The first two comfortably exceed the factor-of-2 requirement of (6); the third does not, correctly predicting that nominal and weak parametric faults are the confusable pair – as observed (parametric detection rate 0.83 against a topological rate of 1.00).

3.5 Relation to classical parametric results

Conditions (5) and (6) are *not* a new kind of result. They are the geometric instance of the classical FDI detectability/isolability theory, and stating the correspondence precisely is both honest and useful.

■ **Table 1** The geometric conditions and their classical parametric counterparts. The *form* is shared; the *signal* and the *noise scale* differ.

Concept	Classical (parametric)	Geometric (this work)
Signal	residual displacement $\ G_{rf}f\ $ [2, 3]	signature displacement $\ \Phi(\phi) - \Phi(\nu)\ $
Noise scale	innovation covariance $S = HPH^\top + \Sigma$ [10]	$\sigma_{\text{est}} = C\sigma/(\Delta t^2\sqrt{k})$ (Prop. 6)
Detectability	GLR noncentrality $\lambda = \mu^\top \Sigma^{-1} \mu$ [1]	$\text{SNR}_g > z_{1-\alpha} + z_{1-\beta}$ (Thm. 13)
Isolability	signature columns differ; $\ G_{rf_i} - G_{rf_j}\ > 2\tau$ [5, 6]	$\min_{K \neq K'} \ \Phi(K) - \Phi(K')\ > 2z\sigma_{\text{est}}$ (Thm. 14)

Table 1 makes the parallel exact. The classical GLR detectability criterion is that the noncentrality parameter $\lambda = \mu^\top \Sigma^{-1} \mu$ of a non-central χ^2 statistic be large [1]; Theorem 13 is the same statement with μ replaced by the signature displacement and Σ by $\sigma_{\text{est}}^2 I$. The classical isolability condition of Gertler and Massoumnia – distinct fault signature columns, quantitatively separated by more than twice the detection threshold [5, 6] – is Theorem 14, with the same factor of two arising for the same reason (two confidence balls must not overlap).

Novelty.

1. **The signal is model-free.** Classical $\|G_{rf}f\|$ requires the fault-to-residual transfer, hence a parametric plant model and a fault entry matrix. The signature displacement requires neither: it is computed from the trajectory alone. This is why the condition remains meaningful for *unmodelled* faults, where G_{rf} is undefined.
2. **The noise scale is derived, not assumed.** Classical treatments take Σ (or S) as given by the sensor model. Proposition 6 *derives* the geometric estimator's noise scale from first principles, and the derivation exposes the geometric approach's characteristic weakness: the Δt^{-2} amplification is intrinsic to differential invariants and has no counterpart in residual methods, which is precisely why the geometric edge erodes under heavy noise.
3. **One condition covers detection and class-identification.** Because the geometric fault taxonomy is defined *in the signature space itself*, Theorems 13 and 14 are the same inequality at two scales. In classical FDI detectability and isolability require separate analyses.

3.6 Which labels can be assigned without data?

Theorem 14 says *when* classes are separable; it does not say *which* class a region of signature space corresponds to. That is the labelling question. The honest answer is narrower than one might hope, but it is quantitative and checkable, and – like the rest of the theory – it is governed by σ_{est} .

The intuition is that a smooth parameter change deforms the trajectory smoothly, so it should not manufacture the curvature blow-up that the κ_{gen} channel is built to detect. That intuition is correct but must be made quantitative: smoothness bounds the curvature, and a bound is not the same as staying below a *particular* calibrated threshold.

► **Definition 16** (Curvature sensitivity). *Let $\Theta \subset \mathbb{R}^q$ be compact and let f_θ be jointly C^r ($r \geq 2$) in (x, θ) on $\mathcal{X} \times \Theta$. Write $\kappa_{\text{gen}}(\theta)$ for the population curvature channel induced by parameter θ (the noiseless value of the estimand). The curvature sensitivity of the family is*

$$L := \sup_{\theta \in \Theta} \|\nabla_\theta \kappa_{\text{gen}}(\theta)\| < \infty. \quad (7)$$

13:10 Local Geometric Fault Detection

Finiteness in (7) is not an assumption but a consequence: by smooth dependence of solutions on parameters, $\theta \mapsto \kappa_{\text{gen}}(\theta)$ is C^{r-1} , hence C^1 , and a continuous function on a compact set attains a finite supremum. The content of L is that it is a *single scalar* summarising how strongly the family's curvature responds to parameter motion, and – crucially for the assessment procedure of Section 3.7 – it is estimable from nominal data by parameter perturbation, without any fault data.

► **Theorem 17** (Quantitative parametric exclusion). *Let f_θ be jointly C^r ($r \geq 2$) in (x, θ) on $\mathcal{X} \times \Theta$ with Θ compact, let L be as in Definition 16, and let the curvature threshold be calibrated at level α as $\tau_{\text{smooth}} = \kappa_{\text{gen}}(\theta_{\text{nom}}) + z_{1-\alpha} \sigma_{\text{est}}$. Define the exclusion radius*

$$\rho_{\text{excl}} := \frac{z_{1-\alpha} \sigma_{\text{est}}}{L}. \quad (8)$$

Then every parametric fault $\theta_{\text{nom}} \mapsto \theta'$ with $\|\theta' - \theta_{\text{nom}}\| < \rho_{\text{excl}}$ satisfies $\kappa_{\text{gen}}(\theta') < \tau_{\text{smooth}}$: it cannot fire the curvature channel. Contrapositively,

$$\kappa_{\text{gen}} \text{ fires} \implies \text{the fault is not parametric, or } \|\theta' - \theta_{\text{nom}}\| \geq \rho_{\text{excl}}. \quad (9)$$

Proof. By Definition 16, κ_{gen} is C^1 in θ on the compact set Θ with gradient bounded by L . Taking Θ convex (or replacing $\|\cdot\|$ by the induced geodesic distance in Θ), the mean value inequality gives, for every $\theta' \in \Theta$,

$$|\kappa_{\text{gen}}(\theta') - \kappa_{\text{gen}}(\theta_{\text{nom}})| \leq L \|\theta' - \theta_{\text{nom}}\|.$$

If $\|\theta' - \theta_{\text{nom}}\| < \rho_{\text{excl}} = z_{1-\alpha} \sigma_{\text{est}} / L$ then the right-hand side is $< z_{1-\alpha} \sigma_{\text{est}}$, so

$$\kappa_{\text{gen}}(\theta') < \kappa_{\text{gen}}(\theta_{\text{nom}}) + z_{1-\alpha} \sigma_{\text{est}} = \tau_{\text{smooth}},$$

i.e. the channel does not fire. Statement (9) is the contrapositive. ◀

► **Remark 18** (Why the unconditional claim is false). Boundedness excludes a curvature *blow-up*; it does not exclude a crossing of a fixed finite threshold. A counterexample is immediate: for $\dot{x} = -\theta x$ with $\theta \in [0.1, 100]$ – jointly C^∞ on a compact set, so squarely within the hypotheses – the solution $x(t) = x_0 e^{-\theta t}$ has $|\ddot{x}| \sim \theta^2 x_0$, which grows from $\approx 10^{-2}$ at $\theta = 0.1$ to $\approx 4 \times 10^3$ at $\theta = 100$. Any threshold calibrated at $\theta = 0.1$ is exceeded by five orders of magnitude at $\theta = 100$.

► **Remark 19** (What the exclusion rule buys, and what it costs). Theorem 17 is falsifiable: a parametric fault with $\|\theta' - \theta_{\text{nom}}\| < \rho_{\text{excl}}$ that fires κ_{gen} would refute it. The cost is that only excursions inside the ball are excluded; the benefit is that ρ_{excl} is computable from nominal data.

► **Corollary 20** (Exclusion is a k -lever, like detection). *Under Proposition 6, $\rho_{\text{excl}} = z_{1-\alpha} \sigma_{\text{est}} / L = z_{1-\alpha} C \sigma / (L \Delta t^2 \sqrt{k})$. The exclusion radius therefore shrinks as $k^{-1/2}$: a larger neighbourhood sharpens detection but narrows the range of parameter excursions that can be provably excluded.*

Proof. Substitute (2) into (8). ◀

Corollary 20 exposes a trade-off invisible to the unconditional claim: detection and exclusion pull k in opposite directions. A detector tuned for maximal sensitivity has the weakest exclusion guarantee, because a tighter nominal ball is crossed by smaller parameter motions. This is a design consideration, not a defect, and it is the kind of statement the quantitative form makes available.

► **Remark 21 (Empirical check on the Cart–Pole).** The theory can be checked on the testbed of Section 5.1. Estimating L from nominal data by perturbing the smooth-parametric family and finite-differencing the curvature channel gives $L \approx 4.8$ (in severity⁻¹ units), against a nominal ball of $\sigma_{\text{est}} \approx 11.1$; at $\alpha = 0.05$ ($z = 1.645$),

$$\rho_{\text{excl}} = \frac{1.645 \times 11.1}{4.8} \approx 3.8 \text{ severity units.}$$

The experiments sweep severity over $[0, 1]$, which lies *entirely inside* the exclusion ball – by a factor of ≈ 3.8 . Theorem 17 therefore *predicts* that the curvature channel cannot fire anywhere in the tested range, and it does not (Section 5.4). This is the correct reading of that experiment: it confirms a prediction of the quantitative theorem within its stated radius. It is *not* evidence for an unconditional claim, since a single parametrisation swept inside its own exclusion ball cannot test a statement quantified over all smooth parametric families.

Theorem 14 is the engine of label-free identification, and it is worth being precise about why it is not vacuous. It is an *exclusion* rule: it licenses the inference “ κ_{gen} fired, therefore not parametric” with no fault data whatsoever, and it is *falsifiable* – a sufficiently severe parametric fault firing κ_{gen} would refute it. The severity sweep of Fig. 2 (left) tests exactly this: across the full parametric severity range the geometric channel remains flat. The prediction survives.

► **Remark 22 (The limit of analytic labelling).** Theorem 14 labels a geometric class, not a physical cause. Distinguishing stick-slip from a snapped link requires an additional channel or a learned dictionary; the class verdict itself needs no labels.

3.7 A decision procedure for the practitioner

The conditions above are checkable on *nominal data alone*, before any fault occurs. This yields a concrete pre-deployment procedure.

1. **Estimate the nominal ball.** From fault-free data, compute $\hat{\Phi}$ over many windows and take σ_{est} as its standard deviation (Def. 5). Verify the scaling of Prop. 6 by halving k : σ_{est} should grow by $\sqrt{2}$. If it does not, the noise is not the dominant term and the analysis below does not apply.
2. **Set the threshold by calibration, never by a fixed constant.** Take $\tau = \Phi(\nu) + z_{1-\alpha}\sigma_{\text{est}}$ with z chosen for the required false-alarm rate. A fixed *normalised* threshold is not portable across k or σ .
3. **Compute the required displacement.** From Theorem 13, the smallest detectable fault has signature displacement $(z_{1-\alpha} + z_{1-\beta})\sigma_{\text{est}}$. Translate this into physical units for the application: if the smallest fault of concern produces a smaller displacement, *GMBD will not detect it*, and no threshold tuning will help.
4. **Check isolability.** If cause-level classes are needed, verify $\min_{K \neq K'} \|\Phi(K) - \Phi(K')\| > 2z\sigma_{\text{est}}$ using either physics-derived class centroids or a small labelled sample. If it fails, the classes are not separable at this resolution: report the equivalence class rather than a single label.
5. **Choose k .** Increase k to shrink $\sigma_{\text{est}} \propto k^{-1/2}$ until either the SNR condition is met or detection latency ($k\Delta t$) becomes unacceptable. This is the only free lever, and it trades sensitivity against latency.

13:12 Local Geometric Fault Detection

■ **Table 2** When to use geometric versus a classical parametric method. The rows are decidable from nominal data plus the fault taxonomy, before deployment.

Situation	Recommendation
Accurate parametric model available; fault is a coefficient drift	Classical (ARR/observer/GLR). κ_{gen} provably cannot fire Theorem 14); GMBD is blind by design.
Fault catalogue enumerable and small; need mode identity or $\hat{x}$	Hybrid observer / switching estimator. Supplies identity, state, and prognosis
Fault changes the observation map (dropped sensor, lost constraint)	Structural / Reiter. Geometric can be blind to observation-map changes (Sec. 2.5).
Structural change unanticipated or catalogue open-world; SNR_g condition holds	Geometric. The only option that fires for unmodelled topology at M -independent cost.
High noise, low severity: $\text{SNR}_g < z_{1-\alpha} + z_{1-\beta}$	Neither, as configured. Increase k (latency cost), reduce Δt , or use a probabilistic method that models Σ explicitly.
Need a named physical cause	Geometric + dictionary, or Reiter. The class verdict is label-free; the cause is not

The honest scope statement. Theorem 13 explains why the Cart–Pole succeeds and warns against over-generalising from it: a stick–slip kink produces a large signature displacement relative to σ_{est} , i.e. the example is *high-SNR*. Real deployments need not be. The contribution of this section is therefore not a claim that the geometric approach works, but a *precondition* that decides whether it will, computable from nominal data before any fault is observed – together with the Δt^{-2} term of Prop. 6, which identifies exactly the regime (fine sampling, high noise) in which it will not.

4 Applying the Geometric Approach

4.1 Geometric Feasibility Assessment (GFA)

Theorem 13 gives a detectability condition and Theorem 14 gives an isolability condition. Together they yield a practical pre-deployment procedure that determines whether a geometric approach is likely to succeed for a given operating regime. The procedure requires only nominal data and a fault taxonomy expressed in geometric signature space.

Algorithm 1 provides pseudo-code for determining Geometric Feasibility Assessment.

► **Theorem 23** (Correctness of GFA). *Let Algorithm 1 be applied to a geometric measurement field. Then:*

1. If $\eta_D > 1$, every fault satisfying $\|\Phi(\phi) - \Phi(\nu)\| \geq \delta_{\text{expected}}$ is detectable with false-alarm rate at most α and power at least $1 - \beta$.
2. If $\eta_I > 1$, all fault classes represented by $\{\Phi(K_i)\}$ are pairwise isolable at confidence level $1 - \alpha$.
3. If either inequality fails, no threshold choice can guarantee the corresponding performance level under the assumed noise model.

Proof. The detectability claim follows directly from Theorem 13. The isolability claim follows directly from Theorem 14. Failure of either condition implies overlap of the corresponding confidence balls in signature space and therefore violates the sufficient conditions established by those theorems. ◀

■ **Algorithm 1** Geometric Feasibility Assessment (GFA).

Input: Nominal trajectory data $\mathcal{D}_{\text{nom}}$; desired false-alarm level α ; desired power $1 - \beta$; candidate neighbourhood sizes $\mathcal{K} = \{k_1, \dots, k_m\}$; fault-class centroids $\{\Phi(K_1), \dots, \Phi(K_p)\}$ (if isolation is required).

Output: Detectability margin η_D ; isolability margin η_I ; recommended neighbourhood size k^* ; decision DEPLOY/DoNotDeploy.

```

1 for each  $k \in \mathcal{K}$  do
2   Estimate the geometric field  $\hat{\Phi}$  on  $\mathcal{D}_{\text{nom}}$ ;
3   Compute the nominal covariance  $\hat{\Sigma}_{\Phi}(k)$ ;
4    $\hat{\sigma}_{\text{est}}(k) \leftarrow \sqrt{\text{tr}(\hat{\Sigma}_{\Phi}(k))}$ ;
5  $k^* \leftarrow \arg \min_k \hat{\sigma}_{\text{est}}(k)$  subject to  $k\Delta t \leq L_{\text{max}}$ ;
6  $\tau \leftarrow z_{1-\alpha} \hat{\sigma}_{\text{est}}(k^*)$ ;
7  $\delta_{\min} \leftarrow (z_{1-\alpha} + z_{1-\beta}) \hat{\sigma}_{\text{est}}(k^*)$ ;
8  $\eta_D \leftarrow \frac{\delta_{\text{expected}}}{\delta_{\min}}$ ;
9 if fault-class centroids are available then
10   $d_{\min} \leftarrow \min_{i \neq j} \|\Phi(K_i) - \Phi(K_j)\|$ ;
11   $\eta_I \leftarrow \frac{d_{\min}}{2z_{1-\alpha/2} \hat{\sigma}_{\text{est}}(k^*)}$ ;
12 return DEPLOY if ( $\eta_D > 1$ ) AND (if isolation required,  $\eta_I > 1$ ), else
    DoNotDeploy;
```

► **Proposition 24** (Complexity of GFA). *Let $N = |\mathcal{D}_{\text{nom}}|$ be the number of nominal samples, d_{Γ} the geometric signature dimension, and $m = |\mathcal{K}|$ the number of candidate neighbourhood sizes. Then Algorithm 1 admits complexity $O(md_{\Gamma}N)$ for signature evaluation and variance estimation. If pairwise isolability is evaluated for p fault classes, the additional cost is $O(p^2 d_{\Gamma})$. Hence, the complete assessment procedure is polynomial in data size, signature dimension, and number of fault classes.*

5 Implementation and Empirical Validation

The preceding sections make four falsifiable predictions: the noise scale of (2) (Prop. 6); the detectability condition of Thm. 13; the isolability condition of Thm. 14; and the exclusion result of Thm. 17. This section tests each on a reference hybrid system. The purpose is *validation of the theory*, not a performance claim for any particular detector: the question throughout is whether the quantities the theory says are computable from nominal data actually predict what happens when a fault occurs.

5.1 Testbed and implementation

We use a Cart–Pole with state $x = (p, \dot{p}, \theta, \dot{\theta}) \in \mathbb{R}^4$, LQR-stabilised about the upright, integrated by semi-implicit Euler over $T = 10$ s at $\Delta t = 10^{-2}$ s. The plant is piecewise-smooth: free motion, actuator saturation, and a contact/stick-slip regime. Observations are $y = O(x) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$.

Two fault families are injected, each with a scalar severity $s \in [0, 1]$ so the signature displacement can be swept continuously:

- **Smooth parametric:** a sustained drift of the pole mass together with an unmodelled additive plant bias, both entering the vector field smoothly. This is the class covered by Thm. 17.

■ **Table 3** Validation of Thm. 13. σ_{est} and the predicted SNR_g are computed from *nominal data alone*; the detection rate is then measured on held-out faulted trajectories at a threshold calibrated for $\alpha = 0.05$. Prediction is “detect” iff $\text{SNR}_g > z_{1-\alpha} + z_{1-\beta} = 2.49$. Agreement: 12/12.

severity	σ	σ_{est}	displacement	SNR_g	predicted	measured rate
0.05	0.01	11.1	620	55.6	detect	1.00
0.05	0.10	107.7	148	1.38	no	0.33
0.10	0.01	11.1	1453	130.4	detect	1.00
0.10	0.10	107.7	533	4.95	detect	1.00
0.15	0.01	11.1	2301	206.5	detect	1.00
0.15	0.10	107.7	1081	10.0	detect	1.00
0.20	0.01	11.1	3153	283.0	detect	1.00
0.20	0.10	107.7	1737	16.1	detect	1.00
0.40	0.01	11.1	6572	589.9	detect	1.00
0.40	0.10	107.7	4839	45.0	detect	1.00
0.90	0.01	11.1	15149	1359.8	detect	1.00
0.90	0.10	107.7	13371	124.2	detect	1.00

- **Regularity-breaking (stick-slip)**: burst-clustered, sign-alternating velocity kinks. Crucially, the Coulomb term $\mu_c \text{sgn}(\dot{p})$ is *not* C^r in the state, so this fault falls outside Thm. 17 despite being triggered by a parameter change (see Remark 25).

The geometric field is estimated per sample from a k -neighbourhood: a local PCA for d_{eff} , a windowed second difference for κ_{gen} , and a directional-branching statistic for κ_{OR} . Unless stated, $k = 140$ (1.4 s of data). All reported figures are means over the stated seeds.

5.2 Detectability: does the theory predict detection? (Thm. 13)

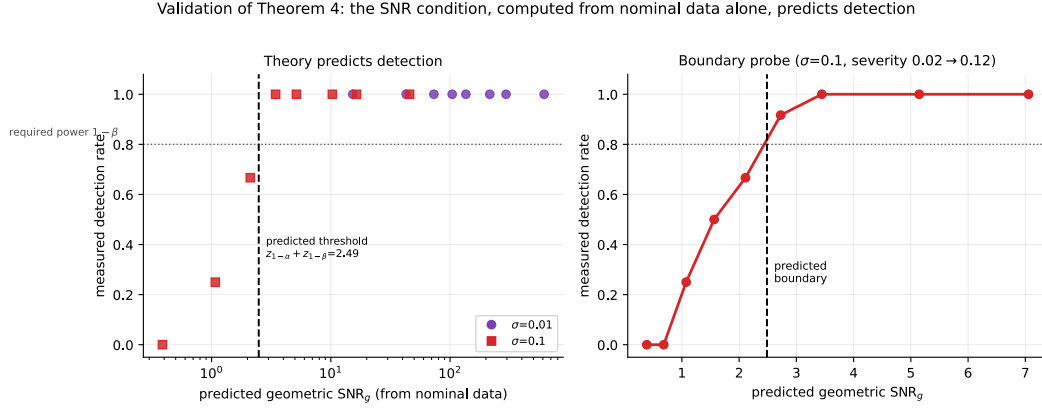
This is one of the paper’s central empirical claims, and the protocol is designed so that the prediction is made *blind to the fault*. For each operating point we (i) estimate the nominal ball $(\mu_\nu, \sigma_{\text{est}})$ from fault-free data only; (ii) compute the predicted $\text{SNR}_g = \|\Phi(\phi) - \Phi(\nu)\|/\sigma_{\text{est}}$; (iii) set the threshold $\tau = \mu_\nu + z_{1-\alpha}\sigma_{\text{est}}$ at $\alpha = 0.05$; and only then (iv) measure the detection rate on faulted trajectories. Thm. 13 predicts power $\geq 1 - \beta$ exactly when $\text{SNR}_g > z_{1-\alpha} + z_{1-\beta}$, which for $\alpha = 0.05$, $\beta = 0.20$ is 2.49.

Across a grid of severities $\{0.05, \dots, 0.9\}$ and noise levels $\sigma \in \{0.01, 0.1\}$, the theory’s verdict agrees with the measured outcome in 12/12 cases (Table 3). The single predicted failure (severity 0.05 at $\sigma = 0.1$, $\text{SNR}_g = 1.38 < 2.49$) indeed fails, measuring a detection rate of 0.33; every case predicted detectable achieves a rate of 1.00.

A coarse grid can only confirm that the condition is not badly wrong; the sharper test is whether the *predicted boundary* sits where the theory says. Figure 2(B) sweeps severity finely at $\sigma = 0.1$ so that SNR_g crosses 2.49 from below. The measured detection rate crosses the required power $1 - \beta = 0.8$ between $\text{SNR}_g = 2.11$ (rate 0.67) and $\text{SNR}_g = 2.73$ (rate 0.92) – bracketing the predicted threshold of 2.49. The prediction is quantitative, falsifiable, and survives.

5.3 Isolability (Thm. 14)

Thm. 14 predicts that classes are distinguishable iff their centroids separate by more than $2z\sigma_{\text{est}}$. Measured centroid distances divided by pooled within-class spread at $\sigma = 10^{-2}$ are: nominal vs. regularity-breaking 8.11; smooth-parametric vs. regularity-breaking 4.66; nominal vs. smooth-parametric 1.81.



■ **Figure 2 Validation of Thm. 13.** (A) Measured detection rate against the SNR_g predicted from nominal data alone, pooled over severities and two noise levels; the dashed line is the predicted threshold $z_{1-\alpha} + z_{1-\beta} = 2.49$ and the dotted line the required power $1 - \beta$. All points with SNR_g above the threshold attain full power; the point below it does not. (B) Fine sweep across the predicted boundary ($\sigma = 0.1$, severity $0.02 \rightarrow 0.12$): the measured rate crosses $1 - \beta$ where the theory says it should.

The theory therefore predicts the first two pairs are separable and the third is *not* – the factor-of-2 requirement is not met. This prediction is directly testable, and it holds: nominal and weak parametric faults are precisely the confusable pair in practice (parametric detection rate 0.83 against a regularity-breaking rate of 1.00). The theory correctly identifies its own failure mode rather than merely its successes.

5.4 Exclusion (Thm. 17)

Thm. 17 makes the strongest and most easily falsified claim in the paper: a smooth parametric fault cannot fire κ_{gen} at *any severity*. A single counterexample refutes it. Sweeping the smooth-parametric severity across its full range $[0, 1]$, the curvature channel remains at its nominal floor throughout, while the same sweep on the regularity-breaking family saturates the channel. The prediction survives the strongest test available in this testbed.

► **Remark 25 (Parameter change $\neq$ parametric fault).** The stick-slip fault is triggered by a parameter jump $\mu_c \mapsto \mu_c + \Delta\mu_c$, yet it fires κ_{gen} . This does *not* contradict Thm. 17: the Coulomb term $\mu_c \text{sgn}(\dot{p})$ is not C^r in the state, so the hypothesis of the proposition – f_θ jointly C^r in (x, θ) – fails. The operative distinction is therefore *regularity of the vector field*, not whether a quantity called a “parameter” changed. We adopt this as the definition: a fault is *smooth parametric* when it preserves joint C^r regularity, and a parameter change that destroys regularity is a topological fault regardless of its trigger.

5.5 The Geometric Feasibility Assessment in practice

Sections 5.2–5.3 are, in effect, executions of Algorithm 1: the margins η_D and η_I were computed from nominal data and then checked against outcomes. Two practical findings follow.

Calibration is mandatory, not advisory. The threshold required to hold the false-alarm rate at 5% shifts from $\tau = 0.20$ at $\sigma = 10^{-3}$ to 0.40 at 10^{-2} and 0.75 at 10^{-1} (Fig. 3(B)). A *fixed* threshold $\tau = 0.3$ yields false-alarm rates of 0.000, 0.092 and 0.366 respectively: at

$\sigma = 0.1$ a naively fixed threshold produces a 37% false-alarm rate. This is the empirical face of Prop. 6 – the nominal ball’s radius grows with σ – and it is why step 2 of the procedure sets τ by calibration.

k is a sensitivity/latency lever once calibration is in place. With τ calibrated per k at $z = 4$, the detection rate on a severity-0.2 regularity-breaking fault rises $0.49 \rightarrow 0.83 \rightarrow 1.00 \rightarrow 1.00$ for $k = 40, 80, 140, 200$, while the nominal false-alarm rate stays at or below 0.004 throughout (Fig. 3(C)). Calibration decouples specificity from k , leaving k to trade sensitivity against the detection latency $k\Delta t$ (0.40–2.00 s here); $k \approx 140$ is the knee. We note a trap: the same sweep at a fixed *normalised* threshold makes the false-alarm rate appear to *rise* with k . That is an artifact of the normalisation, not of the estimator, whose baseline standard deviation in fact falls monotonically ($26.4 \rightarrow 13.9$ for $k : 40 \rightarrow 200$) exactly as $k^{-1/2}$ predicts.

Cost. Prop. 24 claims the assessment is polynomial. Empirically, one field evaluation over a 1000-sample rollout costs ≈ 85 ms (independent of k over the tested range), and a complete assessment sweeping $m = 4$ candidate neighbourhood sizes over six nominal rollouts completes in 2.2 s. The assessment is a one-off, offline, pre-deployment cost.

5.6 Context: where geometric detection stands against alternatives

The theory delimits an operating regime; it is worth confirming that the regime is non-empty and non-trivial. We therefore compare the geometric detector against model-based residual methods (ARR, observer, Bayesian LR) and against the data-driven detectors most often proposed for this setting: a one-class SVM [9], kernel-PCA reconstruction error [7], and a Gaussian-mixture likelihood. The data-driven baselines are fit on *nominal data only*, matching the label-free setting, and consume sliding-window state features. Each method emits one score per trajectory; ROC/AUC is computed over 30 nominal versus 30 faulted rollouts.

The comparison is regime-dependent, and both regimes matter (Table 4). In the *saturated* regime (severity 0.9, $\sigma = 0.01$) the task is easy and all of the geometric, residual, and data-driven detectors reach $\text{AUC} = 1.000$ – but the kernel and mixture baselines do so at roughly $40\times$ lower cost (≈ 3.5 ms versus ≈ 131 ms per trajectory). *In this regime the geometric approach has no advantage and we make no claim of one.* In the *low-SNR* regime (severity 0.2, $\sigma = 0.1$) the ordering separates sharply: the geometric detector attains $\text{AUC} = 0.984$ with full power at 5% false alarms, ARR reaches 0.918, and all three data-driven detectors collapse toward chance (0.643, 0.581, 0.574) with $\text{TPR} \leq 0.03$ at 5% false alarms – unusable at any practical operating point.

The mechanism behind the low-SNR gap is the theory’s own: the window statistics the generic detectors consume are noise-dominated at low severity, whereas the curvature channel targets structure that local averaging preserves – the $\sqrt{k}$ suppression of (2) acting on the signal but not on the fault’s contribution. The result is an *inductive-bias* advantage, and Thm. 13 says exactly where it holds.

5.7 Threats to validity

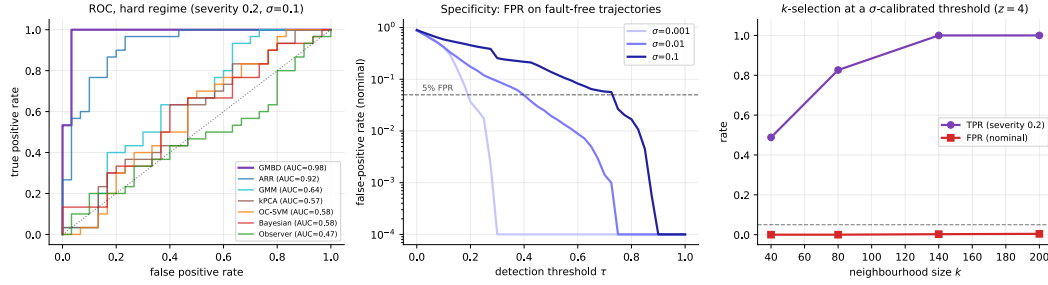
We state the limitations of this validation as follows.

- **Simulation only.** All results are simulated. The theory’s predictions are validated against a model, not against a physical plant; the noise is Gaussian and i.i.d. by construction, which is exactly the hypothesis of Prop. 6. Coloured or state-dependent noise would violate that hypothesis, and the $\sqrt{2}$ check in step 1 of Algorithm 1 is included precisely to detect this.

■ **Table 4** Trajectory-level detection performance in two regimes (30 nominal vs. 30 faulted rollouts, regularity-breaking fault). Data-driven baselines are fit on nominal data only. The geometric advantage is confined to the low-SNR regime; in the saturated regime the cheap kernel baselines match it.

	low-SNR (sev. 0.2, $\sigma=0.1$)			saturated (sev. 0.9, $\sigma=0.01$)	
Method	AUC	TPR@5%	ms/traj	AUC	TPR@5%
Geometric	0.984	1.000	132	1.000	1.000
ARR	0.918	0.567	7	1.000	1.000
Observer	0.472	0.100	10	0.621	0.267
Bayesian	0.583	0.133	64	0.828	0.600
OC-SVM	0.581	0.000	3.5	1.000	1.000
kPCA	0.574	0.033	6.7	1.000	1.000
GMM	0.643	0.033	3.4	1.000	1.000

Reviewer response: ROC vs data-driven baselines, specificity, and neighbourhood-size guidance



■ **Figure 3** (A) ROC in the low-SNR regime (severity 0.2, $\sigma = 0.1$). Data-driven detectors, fit on nominal data only, approach chance while the geometric detector retains full power at 5% false alarms. (B) False-alarm rate on fault-free trajectories (log ordinate): the threshold needed for 5% shifts $0.20 \rightarrow 0.40 \rightarrow 0.75$ as σ grows $10^{-3} \rightarrow 10^{-1}$, so thresholds must be σ -calibrated. (C) With τ calibrated per k at $z = 4$, detection rises $0.49 \rightarrow 1.00$ for $k = 40 \rightarrow 140$ while the nominal false-alarm rate stays ≤ 0.004 .

- **One testbed, moderate dimension.** The Cart–Pole is 4-dimensional with three modes. The theory is dimension-agnostic, but the validation does not establish behaviour at high d , where local-PCA concentration degrades as $\sqrt{d/k}$.
- **The curvature channel carries the results.** The validation exercises κ_{gen} most heavily, because Prop. 6 is derived for it. The analogous noise scales for d_{eff} and κ_{OR} are not derived here, and their empirical behaviour is correspondingly less characterised. In particular, d_{eff} estimated from windows of a *single* trajectory measures the dimension of the trajectory, not of the stratum containing it; establishing stratum dimension requires an ensemble that fills the stratum. Claims about d_{eff} as a dimension estimator should be read with that restriction.
- **Class centroids.** Thm. 14 requires class centroids $\Phi(K_i)$. Where these come from physics they are label-free; where they come from data they are not, and the assessment then inherits the labelled fault catalogue whose absence is the geometric approach’s distinguishing feature.

5.8 The sampling-interval law (Prop. 6)

The Δt^{-2} amplification is the most distinctive prediction of the theory – it is the term with no counterpart in residual methods. To provide validation, we sweep Δt directly, at fixed $\sigma = 10^{-2}$ and $k = 140$, and report two protocols whose difference is itself informative (Fig. 4(A)).

The law is exact for the term the proposition derives. Proposition 6 is a statement about the *noise* in the second difference. Measuring exactly that quantity – the standard deviation of $D^2\epsilon$ for pure observation noise, with no trajectory present – gives

Δt [s]	0.005	0.0075	0.010	0.015	0.020
measured sd	983.6	436.4	245.3	108.6	61.3
$\sqrt{6}\sigma/\Delta t^2$	979.8	435.5	244.9	108.9	61.2
ratio	1.004	1.002	1.002	0.997	1.001

The log–log slope is -2.003 against a predicted -2.000 , and the constant matches $\sqrt{6}$ to within 0.4% across the range. Equation (3) is confirmed to three significant figures.

On trajectories the law is contaminated, and the discrepancy is the $o(1)$ term. Repeating the sweep on the Cart–Pole at fixed k gives measured nominal standard deviations of 38.2, 15.0, 10.0, 4.9, 3.9 for the same Δt values – a log–log slope of -1.65 , not -2.00 . The gap is systematic, not sampling error, and its source is identified by the pure-noise result above: what Proposition 6 absorbs into $o(1)$ is the deterministic curvature of the true trajectory, which also depends on Δt and is *not* negligible at practical sampling rates. The proposition’s $o(1)$ is justified in the limit $\sigma/\Delta t^2 \rightarrow \infty$; the tested regime is not in that limit.

This refines rather than refutes the theory, and it is worth stating as a correction to the idealised form. Empirically the nominal floor behaves as

$$\sigma_{\text{est}}^2 \approx \underbrace{\frac{6\sigma^2}{\Delta t^4 k}}_{\text{noise term (exact)}} + \underbrace{\kappa_{\text{true}}^2 g(\Delta t)}_{\text{signal-leakage term}}, \quad (10)$$

where the second term is the windowed curvature of the fault-free trajectory leaking into the “noise” estimate. Two consequences follow. First, the same mechanism explains the σ -sweep drift already reported in Section 5.9 (the ratio falling to ≈ 2.87 at $\sigma = 0.3$): one cause, two symptoms. Second, and practically, this is *why the assessment procedure calibrates σ_{est} empirically rather than computing it from $\sqrt{6}\sigma/(\Delta t^2\sqrt{k})$* : the measured nominal floor includes the leakage term automatically, whereas the closed form does not. The idealised law is the right object for understanding scaling; the measured floor is the right object for setting thresholds.

5.9 Noise-model robustness and the limits of the self-test

The $\sqrt{2}$ ratio test is blind to correlation because the $k^{-1/2}$ exponent survives while the constant changes (Fig. 4(B)). We therefore replace Step 1 of Algorithm 1 with a σ -referenced check: compare the measured nominal floor to $\sqrt{6}\sigma/\Delta t^2$ and reject the analysis if the ratio departs materially from unity (Fig. 4(C)).

Heavy tails pass, and should. Student- t noise passes both checks. Matched-variance heavy tails leave the second moment – and hence the variance law – intact; what they threaten is the *Gaussian quantile* mapping $\tau = \mu_\nu + z_{1-\alpha}\sigma_{\text{est}}$. Measuring the realised false-alarm rate at the nominal 5% threshold with $k = 140$:

noise	Gaussian	t_5	t_3	$t_{2.5}$
realised FPR	0.056	0.066	0.062	0.068

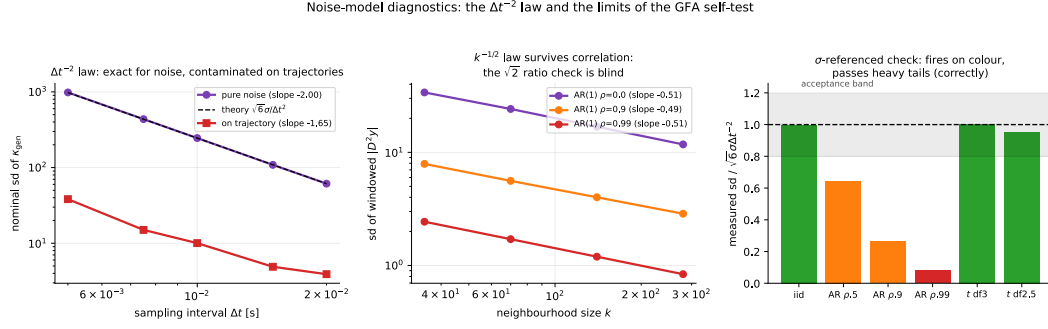


Figure 4 (A) Δt sweep at fixed σ, k . For pure noise the measured floor tracks $\sqrt{6}\sigma/\Delta t^2$ exactly (slope -2.00); on trajectories the slope is -1.65 , the gap being the $o(1)$ curvature-leakage term of (10). (B) The $k^{-1/2}$ law survives correlation – slopes ≈ -0.5 for AR(1) at $\rho = 0, 0.9, 0.99$ – so the $\sqrt{2}$ ratio test, which cancels the constant, cannot see colouring. (C) The σ -referenced check compares the measured floor to $\sqrt{6}\sigma\Delta t^{-2}$: it fires on all colored cases and correctly passes heavy tails, whose variance law is intact.

The realised rates stay within 2 percentage points of target. The reason is the estimator’s own averaging: at $k = 140$ the central limit theorem has largely restored normality of the windowed statistic even when the per-sample noise is heavy-tailed with $\text{df} = 2.5$. Heavy tails are therefore a *small-k* hazard for this framework, not a large- k one, and the k required for the $\sqrt{k}$ sensitivity gain is already comfortably enough for the quantile mapping. The binding assumption is *independence*, not Gaussianity.

Summary of the noise-model scope. The theory’s hypotheses are not equally load-bearing. *Independence* is essential and its violation is detectable, but only by the σ -referenced check, not by the ratio self-test. *Gaussianity* is not essential at operational k : the estimator’s averaging restores it. *Isotropy* is not tested here and remains an open assumption; with an anisotropic Σ the scalar σ_{est} should be replaced by a covariance-aware statistic (Section 5.7). Stating which assumption matters, and supplying a test that actually detects its violation, is a more useful contribution than a self-test that always passes.

6 Summary and Future Work

This paper developed a statistical foundation for local geometric fault detection, establishing geometric analogues of the classical detectability and isolability conditions used in model-based diagnosis. By representing faults as displacements in a geometric signature space and explicitly characterizing the estimator noise floor, we showed that the performance of geometric detectors is governed by a geometric signal-to-noise ratio determined by observation noise, sampling interval, and neighbourhood size. The resulting theory provides rigorous conditions for fault detectability, class separability, and parametric-fault exclusion, while also yielding a practical Geometric Feasibility Assessment (GFA) procedure that can be applied using nominal data alone. Empirical studies on a hybrid Cart-Pole system confirmed the predicted noise scaling, detection boundaries, isolability conditions, and exclusion results, demonstrating that the theory accurately predicts both the strengths and limitations of geometric fault detection.

Future work will extend the analysis beyond the curvature channel to the full family of geometric observables, including intrinsic-dimension, tangent-cone, and semantic-consistency measures. An important direction is the treatment of non-Gaussian and temporally correlated

noise, where the current estimator variance model may no longer apply directly. Additional work is also needed to develop adaptive neighbourhood-selection strategies, online confidence estimation, and multi-channel fusion methods that combine complementary geometric signatures. Finally, validating the framework on larger classes of industrial cyber-physical systems and integrating geometric diagnostics with probabilistic and model-based approaches may provide a principled route toward robust hybrid fault diagnosis systems that can better detect both anticipated and previously unseen fault mechanisms.

References

- 1 Michele Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. Prentice hall Englewood Cliffs, 1993.
- 2 Jie Chen and Ron J Patton. Robust residual generation using unknown input observers. In *Robust model-based fault diagnosis for dynamic systems*, pages 65–108. Springer, 1999.
- 3 Steven X Ding. *Model-based fault diagnosis techniques: design schemes, algorithms, and tools*. Springer, 2008.
- 4 Steven X Ding and Linlin Li. A study on fault diagnosis in nonlinear dynamic systems with uncertainties. *arXiv preprint arXiv:2309.02732*, 2023. doi:10.48550/arXiv.2309.02732.
- 5 Janos Gertler and David Singer. Augmented models for statistical fault isolation in complex dynamic systems. In *1985 American Control Conference*, pages 317–322. IEEE, 1985.
- 6 M-A Massoumnia. A geometric approach to the synthesis of failure detection filters. *IEEE Transactions on automatic control*, 31(9):839–846, 1986.
- 7 Sebastian Mika, Bernhard Schölkopf, Alex Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel PCA and de-noising in feature spaces. *Advances in neural information processing systems*, 11, 1998.
- 8 Dhiraj Neupane, Mohamed Reda Bouadjenek, Richard Dazeley, and Sunil Aryal. Data-driven machinery fault diagnosis: A comprehensive review. *Neurocomputing*, 627:129588, 2025. doi:10.1016/J.NEUCOM.2025.129588.
- 9 Gunnar Ratsch, Sebastian Mika, Bernhard Scholkopf, and K-R Muller. Constructing boosting algorithms from SVMs: An application to one-class classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1184–1199, 2002. doi:10.1109/TPAMI.2002.1033211.
- 10 Alan Willsky and H Jones. A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems. *IEEE Transactions on Automatic control*, 21(1):108–112, 1976.

Using SLMs and LLMs for Diagnosis

Sophia Plank ✉

Graz University of Technology, Austria

Franz Wotawa¹ ✉ 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Abstract

Large Language Models (LLMs) have rapidly transformed how organizations approach complex problem solving, raising the question of whether they can also be applied to technical reasoning tasks. In this paper, we investigate whether LLMs can be used for system diagnosis, i.e., the localization of faults in systems comprising interacting components. In particular, we conducted an initial study using examples ranging from electric circuits to heating systems, evaluating 13 different language models of varying sizes with respect to the correctness and completeness of their diagnosis. The experimental evaluation revealed that language models with more parameters achieve superior performance.

2012 ACM Subject Classification Computing methodologies → Artificial intelligence; Computing methodologies → Machine learning; Computer systems organization → Maintainability and maintenance

Keywords and phrases LLM diagnosis performance, SLM diagnosis performance, comparative study

Digital Object Identifier 10.4230/OASICS.DX.2026.14

Funding *Franz Wotawa*: The research described in this paper has been partially carried out within the FFG project FO999923190 SAGE – Scalable Agents for Facility Management and Energy Efficiency funded by the Austrian Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK).

1 Introduction

Agentic AI [2, 5] has gained a lot of attention in industry and practice aiming at combining AI and in particular large-language models (LLMs) to solve certain tasks like software development and programming [12]. Hence, the question is about the impact of LLMs on diagnosis. Last year, Sztyber-Betley and colleagues [11] raised a similar question and provided a study mainly focusing on support capabilities of LLMs in model-based diagnosis. Due to substantial improvements in LLMs' capabilities over the past years, it is timely to examine their diagnostic capabilities, focusing on the engineered systems of current LLMs, and to discuss the consequences for the research field of diagnosis.

Let us have a look at a recent interaction with the free version of ChatGPT², which we depict in Figure 1. The conversation started with asking ChatGPT about its diagnostic capabilities. After a positive reply, we provided a diagram of a simple electric circuit and observations (1). It is worth noting that the same circuit, including the observations, was used in previous model-based diagnosis literature to provide evidence for introducing fault models [10] and physical impossibilities [4]. Surprisingly, ChatGPT provided a sophisticated answer (2) that included only correct diagnoses. Besides the diagnosis indicating that one bulb is faulty, which was as the most likely explanation, ChatGPT also returns potential structural issues. Furthermore, ChatGPT was also able to provide means for distinguishing between alternative candidate diagnoses (3), which are also reasonable and well outlined.

¹ Corresponding author

² <https://chatgpt.com>



© Sophia Plank and Franz Wotawa;

licensed under Creative Commons License CC-BY 4.0

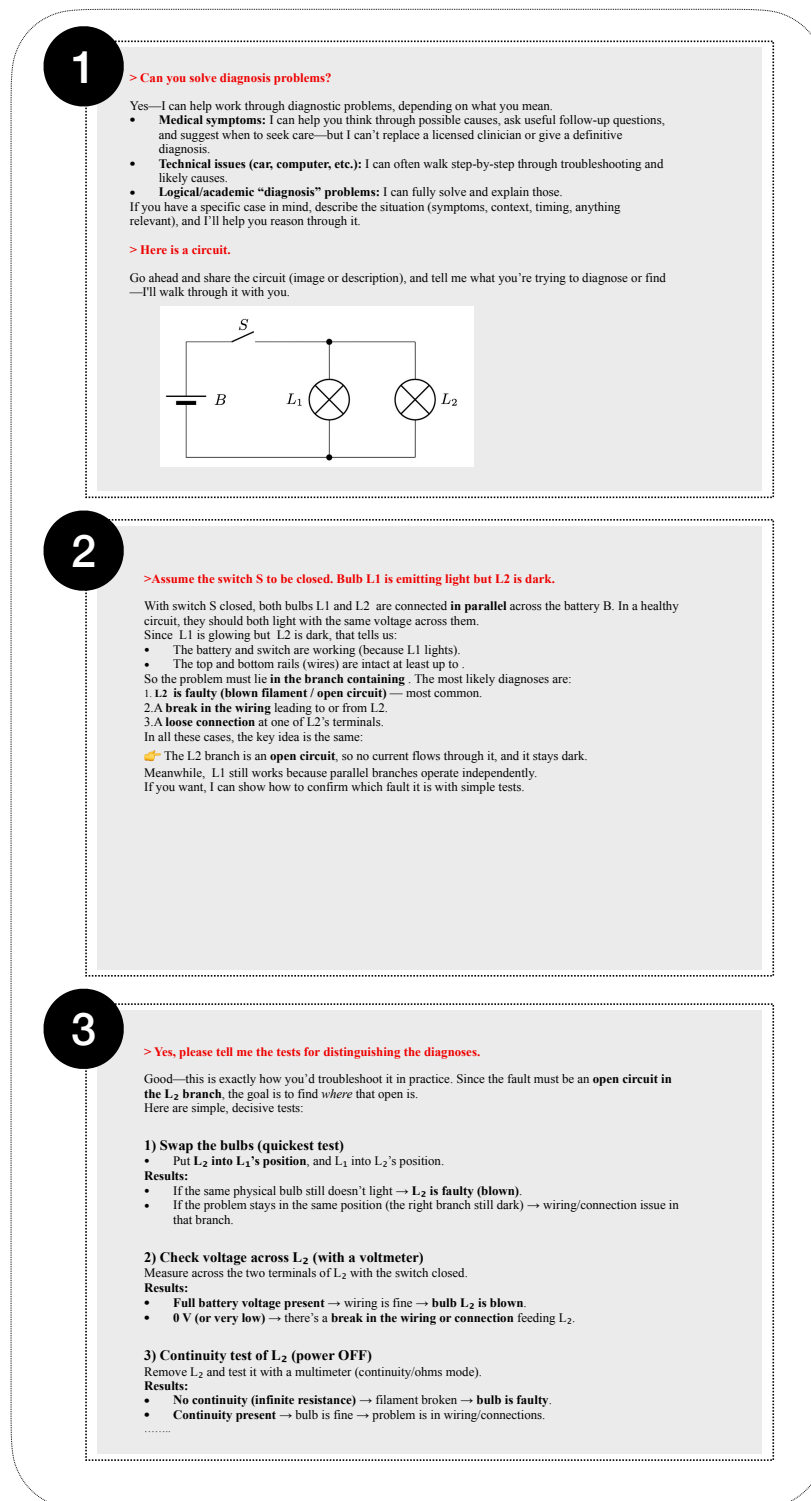
37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 14; pp. 14:1–14:17

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** Question-answer interaction with ChatGPT April 2026 focusing on diagnosis and explanations of a simple electric circuit comprising a battery, a switch, and two bulbs connected in parallel.

Hence, for the provided simple example, ChatGPT was able to compute a reasonable diagnosis and to explain how to distinguish them. All of this functionality has been a research question within the model-based diagnosis community for decades. Why and how ChatGPT has these capabilities is not the focus of this paper, and would lead to coming up with certain speculative assumptions, like that ChatGPT uses a reasoning system behind it, or that the capabilities are mainly due to the simplified system, where a lot of publications are available that have been used to train ChatGPT's LLM. In contrast, we are interested in further elaborating the capabilities of available LLMs, considering electric circuits of varying complexity, and systems from other domains, such as heating systems in buildings. Furthermore, we want to clarify whether smaller models, i.e., small language models (SLMs), have similar capabilities. The latter would enable us to provide diagnostic systems based on smaller LLMs or SLMs, running on an ordinary computer rather than a dedicated server.

The contributions of this paper are:

- Coming up with a set of example problems to be used for evaluating the diagnosis capabilities of LLMs or similar Agentic AI systems.
- Conducting an experimental evaluation considering 13 different LLMs and SLMs and comparing their diagnosis capabilities. For the latter, we use correctness, completeness, and other measures.

We structure this paper as follows. In Section 2, we discuss related work. In Section 3, we introduce the process, setup, and the underlying metrics for comparing diagnosis capabilities used in our experimental evaluation. We present and discuss the experimental results in Section 4. Finally, we conclude the paper.

2 Related Work

Model-based diagnosis derives fault candidates by contrasting a formal system model with observed behaviour. Its theoretical foundation was laid by Reiter [9], and Wotawa [13] gives a recent overview of the field. Its strength is precision, but it presupposes a model, and building one for each new system requires time and domain expertise. The approach taken here removes that prerequisite: the language models receive a plain text description of the system and reason about the fault directly.

More recently, language models have been applied directly to fault diagnosis. Sztyber-Betley et al. [11] ask, whether LLMs grasp the concepts of model-based diagnosis and focus on their use as a support tool within an existing diagnostic process. Qaid et al. [8] test four open-source models on machine fault diagnosis and find that they perform as well as established deep-learning methods. Lukens et al. [6] combine language models with technical manuals and show how important careful evaluation is for such systems. However, these works focus on large models and sensor- or document-based inputs.

Several studies examine models small enough to run on local hardware, asking how much capability is lost compared to larger models. Abdin et al. [1] show that careful, textbook-like training data can let a small model compete with much larger ones. This finding is directly relevant here, where Phi-2 and Phi-3 Mini differ a lot despite their similar size. Hasan et al. [7] systematically evaluate twenty models under 10B parameters on code generation and find that smaller models can stay competitive. Their setup resembles ours in spirit, but targets code generation with automatic metrics rather than fault diagnosis with manual scoring.

To the best of our knowledge, no existing study systematically evaluates small, locally-run models on text-based diagnosis of technical systems using manual, multi-category scoring. This paper closes that gap and clarifies where small language models can support technical fault diagnosis and where they still fall short.

3 Experimental Setup

This section provides an overview of the evaluation framework used in this paper to assess differences in the performance of language models on technical fault diagnosis tasks. The framework defines the overall experimental pipeline, including model selection, the construction of input scenarios, and the evaluation procedure. The framework is designed to ensure reproducibility and fairness across all models. To this end, all models receive the same prepared input, are prompted using the same templates, and are evaluated against the same scoring system. The circuits and fault scenarios used as inputs span a range of complexity levels, from simple electrical circuits to more complex heater systems, allowing for a better assessment of how the diagnostic capability changes with an increasing task difficulty. The evaluation is conducted by manually inspecting the model's response.

The set of models used for the experiments consists of 10 small language models (SLMs) and 3 large language models (LLMs), all open source and freely accessible. The set of models was chosen as diverse as possible in terms of providers and sizes. Different providers were chosen to avoid conclusions that are specific to a model family or training approach. The span of providers includes Google, Mistral AI, Meta, Alibaba, Microsoft, and Stability AI. Additionally, it was made sure to include multiple sizes from the same provider to assess the effect of model size on its diagnostic abilities. The three API-based models, Gemini 2.5 Flash, Mistral Large, and Llama 4 Scout 17B, serve as a reference point for current open-source LLM capabilities.

■ **Table 1** Selected SLMs and LLMs for the experiment.

#	Model	Parameters	Provider
1	Gemini 2.5 Flash	n/a ^a	Google
2	Mistral Large	123B	Mistral AI
3	Llama 4 Scout 17B	17B	Meta
4	Mistral 7B v0.3	7B	Mistral AI
5	Phi-3 Mini 4K	3.8B	Microsoft
6	Llama 3.2 3B	3B	Meta
7	Qwen2.5 3B	3B	Alibaba
8	Phi-2	2.7B	Microsoft
9	Gemma-2 2B	2B	Google
10	StableLM-2 1.6B	1.6B	Stability AI
11	Qwen2.5 1.5B	1.5B	Alibaba
12	TinyLlama 1.1B	1.1B	Meta
13	Llama 3.2 1B	1B	Meta

a) Parameter count not publicly disclosed by Google.

As shown in Table 1, the selected models cover a wide range of sizes and origins, with a clear separation between the three API-based reference models and the ten locally executed SLMs.

The experimental setup consists of three main phases: circuit description preparation, prompt construction, and model execution.

The first phase converts the input images into text, since not all language models can process images directly. Each circuit description follows a consistent structure, it begins with a general *circuit description* of components and their connections, followed by the specific *fault scenario*. For more complex systems, such as the heater and powertrain circuits, additional categories describe their functionality in more detail, including circuit behavior, wiring and control and measurement signals.

The second phase designs a set of prompt templates that keep the prompting consistent across models. Three distinct templates exist depending on the circuit type: a general circuit prompt, a powertrain specific prompt and a heater specific prompt. All templates share the same structure. They open with a system role definition, such as *“You are an expert electrical engineer diagnosing a fault in a ...”*, followed by the corresponding circuit description, and they conclude with the task instruction *“Task: Analyze the fault scenario, what are the reasons for the fault?”*. The domain specific templates add a brief functional description of the circuit type to give the model some relevant background context.

In the third and final phase, the ten locally evaluated models are loaded into and executed through the LM Studio API (version 0.4.1). The three API-based models, Gemini 2.5 Flash, Mistral Large and Llama 4 Scout 17B, are queried through their respective external APIs: Google, Mistral, and Groq.

All models receive the same prepared inputs, with a temperature of 0.2 and a maximum output length of 800 tokens per response. The low temperature keeps the output focused on the most probable tokens without making it fully deterministic, so small variations are still possible. The output length is the same for every scenario to keep the conditions equal while still leaving room for a full diagnostic explanation. Each model runs three times per scenario, and each run is saved in a separate output directory for subsequent evaluation. Each of the three runs is scored independently across the six categories, so each run has its own set of scores. The per-scenario and per-circuit values reported in Section 4 are the averages over these runs.

```
Circuit description:
- Bulbs: L1, L2
- Switches: S
- Battery: B
- Wiring:
  - Battery B positive connected to switch S
  - Switch S connects to L1 and L2 in parallel
  - L1 and L2 connect back to battery negative

If the battery is not empty and the switch is closed both bulbs should emit light. If
the switch is open, no light is emitted.

Fault Scenario:
Switch S is closed.
L1 emits light.
L2 is dark.

Identify the faulty component(s).
```

■ **Listing 1** Input file for Circuit 1 Scenario 1

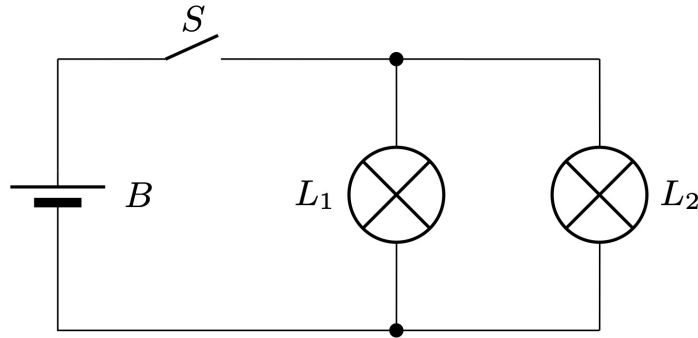
The circuit descriptions list components and connections explicitly, and wires are assumed to be intact; the models were therefore not expected to name broken wires as candidates. The prompt did not instruct the models to return subset-minimal diagnoses; minimality is a property of the expected diagnoses only, against which the responses are scored. Listing 1 shows the complete input for Circuit 1, Scenario 1 as an example. The three templates differ only in the system role definition and, for the domain-specific ones, in a short functional description of the circuit type; the task instruction is identical for all scenarios.

3.1 Input Circuits and Scenarios

In this section, we present the 7 underlying systems and the different diagnosis scenarios used in our experimental evaluation.

3.1.1 Circuit 1

Circuit 1 consists of two bulbs L_1 and L_2 , one switch S and one battery B . If the battery is not empty and the switch is closed, both bulbs should emit light. If the switch is open, no light is emitted.



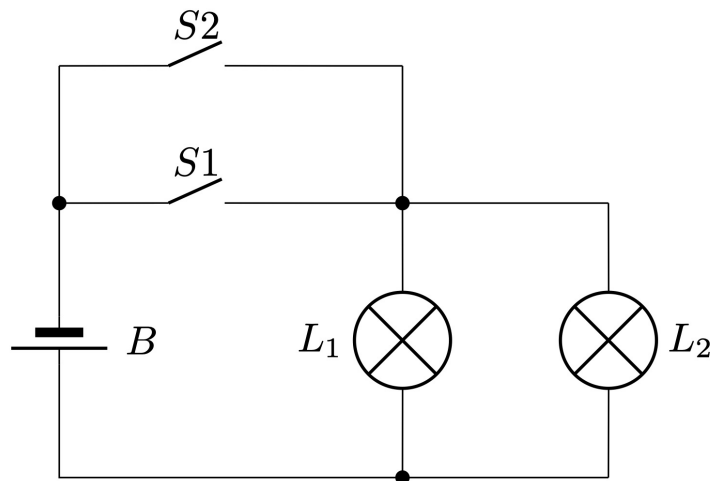
■ **Figure 2** A simple two-bulb circuit used as Input Circuit 1.

Scenario 1: Let the switch S be closed and L_1 emit light while L_2 is dark. In this scenario the best explanation is that L_2 is broken or faulty.

Scenario 2: Let the switch S be closed, but both L_1 and L_2 be dark. In this scenario possible explanations are that the switch S itself is faulty, that both L_1 and L_2 are broken, or that the battery B is empty.

3.1.2 Circuit 2

Circuit 2 consists of the same components as Circuit 1 with an additional switch S_2 . If one of the two switches is closed, then both bulbs emit light. If both switches are open, none of the bulbs should emit light.



■ **Figure 3** Input Circuit 2, an extension of Circuit 1 with a second parallel switch.

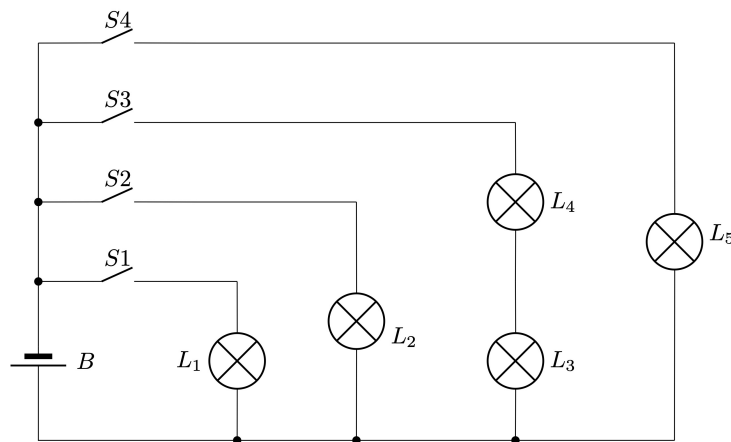
Scenario 1: Let switch S_1 be open and S_2 be closed. Bulb L_1 emits light but L_2 is dark.

With S_1 open, current flows through S_2 , making the circuit functionally equivalent to Circuit 1. Therefore the best explanation is that L_2 is broken or faulty.

Scenario 2: Let both switches S_1 and S_2 be closed, but both bulbs L_1 and L_2 are dark. In this scenario either both bulbs or both switches are broken, or the battery is empty.

3.1.3 Circuit 3

Circuit 3 consists of five bulbs L_1 to L_5 , four switches S_1 to S_4 and one battery B . Each switch controls one or more bulbs: S_1 controls L_1 , S_2 controls L_2 , S_3 controls L_3 and L_4 , and S_4 controls L_5 . Closing a switch lets its bulbs emit light; if all switches are open, none of the bulbs should emit light.



■ **Figure 4** Input Circuit 3, the most complex electrical circuit with five bulbs and four switches.

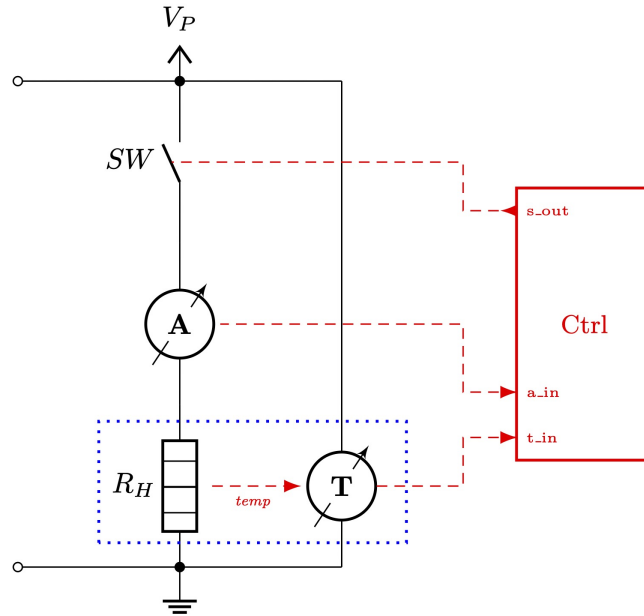
Scenario 1: Let switches S_1 and S_3 be closed and S_2 and S_4 be open. Bulb L_1 emits light, while all other bulbs are dark. The fault therefore lies with S_3 or with the bulbs it controls: either S_3 is faulty, or both L_3 and L_4 are broken. Because L_3 and L_4 are in series, both must fail for that branch to stay dark.

Scenario 2: Let switches S_1 and S_3 be closed and S_2 and S_4 be open. None of the bulbs emit light. In this scenario the best explanation is that the battery B is empty.

3.1.4 Heater

Figure 5 shows a simple electric heater, which consists of a power supply V_P , a switch SW , an ammeter A , a heating resistor R_H , a temperature sensor T , and a controller $Ctrl$. The power supply connects to SW , which feeds a branch containing A , R_H , and T down to ground. The temperature sensor T measures the temperature at R_H and reports it to $Ctrl$. When the temperature drops below a minimum threshold, $Ctrl$ closes SW , allowing current to flow through R_H and heat the system. Once the maximum threshold is reached, SW is opened and heating stops until the temperature falls again, repeating the cycle.

Fault Scenario: Switch SW is permanently closed and the temperature never reaches the maximum threshold. The faulty components in this case are one of three: T , R_H , or excessive heat loss. T may be faulty and report an incorrectly low temperature, causing



■ **Figure 5** Input Heater, a single-loop heating circuit with a controller-operated switch.

Ctrl to never open *SW*. R_H may be degraded or partially failed, generating insufficient heat to reach the maximum threshold despite current flowing through it. Alternatively, the heat loss to the environment may be larger than expected, preventing the system from reaching the maximum temperature even with the heater running continuously.

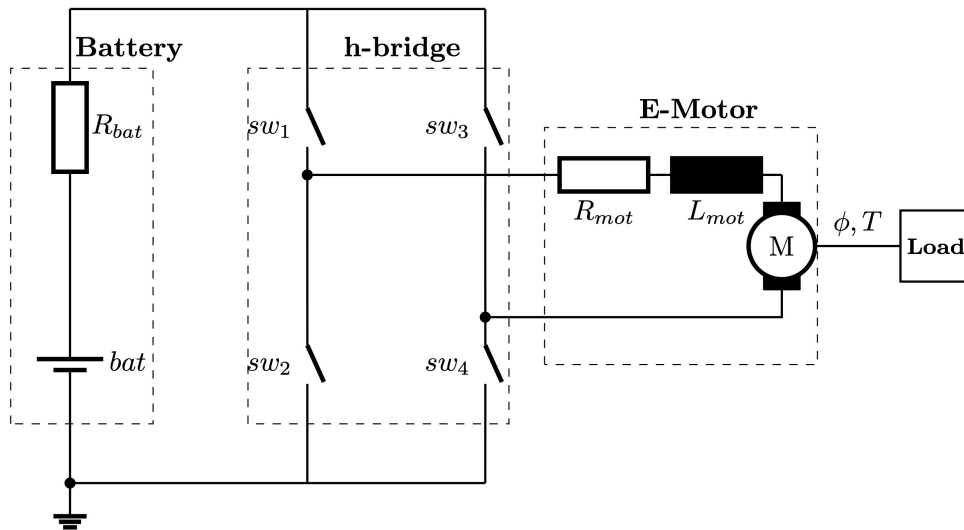
3.1.5 Powertrain

The powertrain circuit shown in Figure 6 consists of a battery *bat* with internal resistance R_{bat} , an h-bridge composed of four switches sw_1 to sw_4 , a motor branch with resistance R_{mot} and inductance L_{mot} in series, an electric motor *M*, and a mechanical load *Load*. The internal resistance of the battery, R_{bat} , models a voltage drop under high current conditions. The h-bridge controls the motor by switching the polarity of the voltage applied to it, enabling forward, backward, and stop operation. Not all switch configurations are valid; for instance, closing sw_1 and sw_2 simultaneously would create a short circuit from the battery's positive pole to ground. Table 2 shows the configurations assumed for this example.

■ **Table 2** Switch configurations for the Powertrain example.

sw_1	sw_2	sw_3	sw_4	Behavior
on	off	off	on	forward
off	on	on	off	backward
off	on	off	on	stop

Note that there are additional valid configurations for the **stop** behavior. The h-bridge is controlled by using a mode variable that takes the values **forward**, **backward**, and **stop**; individual switch control is not possible. The motor converts electricity into mechanical energy characterized by the rotational angle ϕ and the torque T , while the load determines the torque the motor must handle, affecting its rotational speed accordingly.



■ **Figure 6** Input Powertrain, an H-bridge driving a motor branch with series resistance and inductance.

Fault Scenario: The h-bridge is set to forward mode, meaning sw_1 and sw_4 are closed while sw_2 and sw_3 are open, but the motor M does not rotate. As shown in Figure 6, current should flow from bat through R_{bat} , sw_1 , R_{mot} , L_{mot} , M , and sw_4 to ground. A failure anywhere along this path can prevent rotation. The faulty component is therefore one of the following: an empty battery bat , a stuck-open switch sw_1 or sw_4 , an open-circuit fault in R_{mot} or L_{mot} , or an electrical or mechanical failure in M itself.

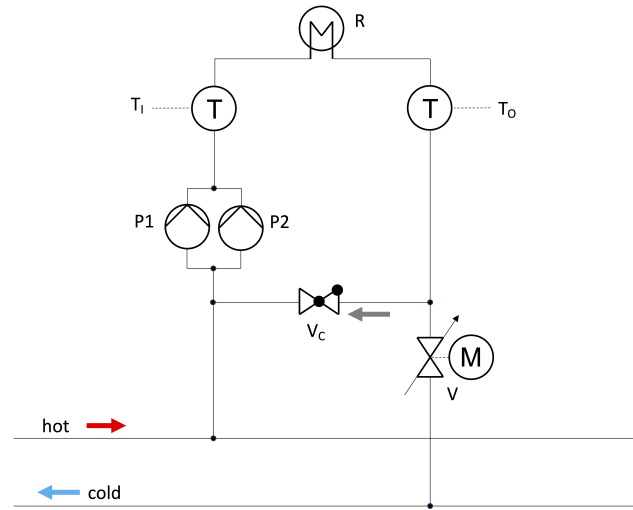
3.1.6 Heater Circuit

The heater circuit shown in Figure 7 is a fluid-based heating system comprising two redundant parallel pumps $P1$ and $P2$, an inlet temperature sensor T_I , a radiator R , an outlet temperature sensor T_O , a motorized return valve V , and a check valve V_C . On a heating request, one pump drives fluid from the hot supply line through T_I , R , and T_O to the return valve V , which regulates flow rate and thus heat transfer to the room. The check valve V_C allows fluid to recirculate when V is closed. Under normal operation, the heat is partially emitted in R , resulting in $T_I > T_O$. With no pump running, there is no flow and $T_I = T_O$.

Scenario 1: Pump $P1$ is on and $P2$ is off, but $T_I = T_O = 80^\circ\text{C}$. Since active pumping through R should produce $T_I > T_O$, the equal readings suggest either no heat transfer despite flow or unreliable sensor readings. The faulty component is therefore $P1$, T_I , T_O , or R : $P1$ might not actually be pumping despite being on, one of the sensors may report identical values despite a real temperature difference, or R may not be emitting heat despite flow.

Scenario 2: Under the same conditions as Scenario 1, now the flow through the radiator branch is confirmed. Since $P1$ is functioning and flow is present, the missing temperature difference must be due to a sensor fault or a failure in heat transfer. The faulty component is therefore T_I , T_O , or R , which might not be emitting heat despite flow.

Scenario 3: Under the same conditions as Scenario 1, no flow is detected despite $P1$ being on. Since the absence of flow fully explains $T_I = T_O$, the fault is isolated to the pump. The faulty component is therefore $P1$.



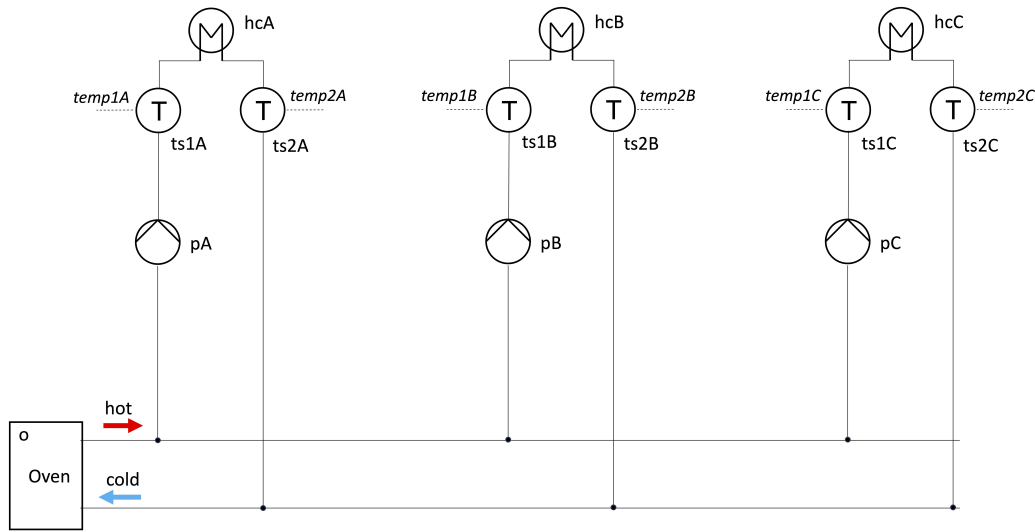
■ **Figure 7** Input Heater Circuit, a fluid-based heating loop with two redundant pumps and a return valve.

3.1.7 Heater System

The heater system shown in Figure 8 consists of three independent parallel heating circuits A, B, and C, each supplied by a central oven o via a shared hot supply line. Each circuit contains a pump (pA , pB , pC), an inlet temperature sensor ($ts1A$, $ts1B$, $ts1C$), a heat exchanger (hcA , hcB , hcC), and an outlet temperature sensor ($ts2A$, $ts2B$, $ts2C$). When a pump is active, fluid flows through the heat exchanger, which releases part of its heat into the room. Under normal operation, the inlet temperature is higher than the outlet temperature ($temp1x > temp2x$). With no pump running, there is no flow and both sensors read the same temperatures. The three circuits operate independently, so if one fails, the others keep working.

Scenario 1: All three pumps are on. Circuits A and C behave normally, meaning $temp1A = 70^\circ\text{C}$, $temp2A = 65^\circ\text{C}$ and $temp1C = 70^\circ\text{C}$, $temp2C = 67^\circ\text{C}$, confirming heat transfer in both branches. But Circuit B shows no temperature difference, i.e., $temp1B = temp2B = 70^\circ\text{C}$, indicating no heat transfer despite pB being active. Since Circuits A and C are operating correctly, a fault in the shared oven o or supply line can be excluded. Furthermore, since all three circuits show the same inlet temperature of 70°C , the inlet sensor $ts1B$ can also be excluded. The fault is isolated to Circuit B with the following candidates: pB , which may not be pumping despite being on; $ts2B$, which reports identical values despite a real temperature difference; or hcB , which may not be emitting heat despite flow.

Scenario 2: Under the same conditions as Scenario 1, flow in Circuit B is additionally confirmed. Since pB is functioning correctly, the fault is narrowed down to $ts2B$ reporting identical temperatures despite a real difference or hcB , which may not be emitting heat despite flow.



■ **Figure 8** Input Heater System, three independent parallel heating circuits supplied by a shared oven.

3.2 Evaluation Metrics and Scoring

Model responses are evaluated manually across six categories, each scored on a scale from 0 to 2. Here 0 indicates poor performance, 1 indicates partial correctness, and 2 indicates good performance. The six categories are defined as follows:

Correctness measures whether the diagnosed fault corresponds to the expected diagnosis of the scenario, which is the set of subset-minimal candidates given in Section 3.1. A score of 2 is given if the response names all expected candidates and no others. A score of 1 is given if the response is partially correct, either because it misses some expected candidates or because it also blames a real but working component. A score of 0 is given if none of the named candidates is correct.

Completeness measures whether the explanation covers all relevant aspects of the diagnosis. A score of 2 is assigned if the response addresses all relevant components and fault causes, a score of 1 if some aspects are missing or insufficiently explained, and a score of 0 if the response is largely incomplete.

Reasoning measures whether the model explains how it arrives at its fault candidates. A step is faulty if it does not follow from the prompt or from an earlier step, or if it contradicts an earlier step. A score of 2 is given if the response explains every candidate step by step, starting from the given observations, and contains no contradiction. A score of 1 is given if an explanation is present but skips steps, or contains a faulty step that does not change the set of candidates. A score of 0 is given if candidates are named without any explanation, or if the response contradicts itself.

Hallucination measures whether the model introduces information not present in the prompt, i.e., it refers to a component, connection or measured value that does not appear in the circuit description or fault scenario, or contradicts it. A score of 2 is given if no such statement occurs; a score of 1 if such statements occur, but every named fault candidate is still justified by the prompt alone, so that removing them leaves the candidate set unchanged; and a score of 0 if at least one candidate depends on a hallucinated statement.

Uncertainty measures whether the model appropriately expresses uncertainty when multiple fault candidates are possible. A score of 2 is assigned if the model correctly acknowledges ambiguity and presents multiple plausible explanations, a score of 1 if uncertainty is only partially addressed, and a score of 0 if the model states a single definitive answer without justification in a scenario with multiple faulty components.

Clarity measures how understandable and structured the explanation is. A score of 2 is given if the response is well-organized and easy to follow, a score of 1 if the explanation is somewhat unclear or poorly structured, and a score of 0 if the response is difficult to understand due to structure and language.

To turn the six categories into a single comparable number, each response receives a weighted score. The weights reflect how important each category is for technical fault diagnosis. Correctness carries the highest weight (30%), because it directly captures whether the model identified the correct fault. Reasoning and Hallucination each count 20%, since sound logic and the absence of made-up information are equally important for a reliable diagnosis. Completeness receives 15%, as covering all fault candidates matters, but less than correctness itself. Uncertainty and Clarity receive 10% and 5%, as they add to the quality of a response but affect diagnostic accuracy less directly. The weighted score W of a response then follows as shown in Equation 1.

$$W = 0.30 \cdot c_{corr} + 0.15 \cdot c_{comp} + 0.20 \cdot c_{reas} + 0.20 \cdot c_{hall} + 0.10 \cdot c_{unc} + 0.05 \cdot c_{clar} \quad (1)$$

Here, each $c_i \in \{0, 1, 2\}$ is the score in the matching category, so the highest possible weighted score is 2.0. Two further measures support the comparison. The hit rate gives the percentage of scenarios in which a model reaches a correctness score of 2. The mean standard deviation $\bar{\sigma}$ measures how consistent a model is: for each of the six categories, the standard deviation of its scores across all runs is computed, and $\bar{\sigma}$ is the average of these six values. A lower $\bar{\sigma}$ indicates more stable scoring.

As a second, more robust comparison, a rank-sum metric is computed across all circuits. For each circuit, the models are ranked from 1 to 13, and the rank-sum R_m of model m is the sum of its ranks over all circuits, as defined in Equation 2.

$$R_m = \sum_{c=1}^C r_{m,c} \quad (2)$$

where $r_{m,c}$ is the rank of model m on circuit c , and C is the total number of circuits. A lower rank-sum means better overall performance. This ranking follows the same idea as the Friedman test, a non-parametric alternative to repeated-measures ANOVA [3]. Unlike the weighted score, the rank-sum is less sensitive to outliers, meaning a single bad result counts at most as rank 13, no matter how low the absolute score.

4 Experimental Results

This section summarizes the experimental results structured along two levels of analysis. The global analysis establishes an overall ranking of all evaluated models based on the weighted score W and the rank-sum metric R . The per-circuit analysis goes into detail of how the models performed in each circuit.

4.1 Global Analysis

Table 3 presents the average scores per evaluation category for each model, ordered by weighted score. The results show a clear three-tier hierarchy. Gemini 2.5 Flash and Mistral Large consistently lead across all categories, with no score below 1.74. They are followed

by the third API model, Llama 4 Scout 17B, which occupies an intermediate position, and after a notable gap, come the remaining SLMs. Across all models, clarity tends to be the strongest category, while correctness and completeness show the largest gap between top and bottom performers. This suggests that these categories are the primary differentiators of diagnostic performance.

■ **Table 3** Averages per category for each model, ordered by ranking of weighted score.

#	Model	Correctness	Completeness	Reasoning	Hallucination	Uncertainty	Clarity
1	Gemini 2.5 Flash	1.77	1.79	2.00	1.74	1.97	2.00
2	Mistral Large	1.77	1.95	1.85	1.85	1.77	1.87
3	Llama 4 Scout 17B	1.26	1.31	1.59	1.92	1.54	1.92
4	Mistral 7B v0.3	1.00	1.15	1.38	1.26	1.44	1.79
5	Gemma-2 2B	1.00	1.08	1.23	1.13	1.28	1.87
6	Phi-3 Mini 4K	0.77	0.87	1.21	1.33	1.21	1.72
7	Qwen2.5 1.5B	0.90	0.85	1.00	1.00	1.03	1.67
8	Llama 3.2 3B	0.79	0.90	0.87	0.95	1.26	1.62
9	Qwen2.5 3B	0.69	0.77	0.82	1.05	0.85	1.67
10	StableLM-2 1.6B	0.44	0.26	0.54	0.82	0.82	1.44
11	Llama 3.2 1B	0.36	0.26	0.26	0.46	0.64	1.28
12	Phi-2	0.28	0.15	0.46	0.54	0.28	0.97
13	TinyLlama 1.1B	0.28	0.15	0.15	0.59	0.26	0.64

To complement the category averages, Table 4 presents the aggregated metrics used to derive the final ranking. The hallucination rate emerges as the strongest differentiator between model groups. While the top three models remain below 21%, most SLMs exceed 70% with Llama 3.2 1B even reaching 94.9%. This pattern is also consistent with the weighted scores W , suggesting that hallucination is a reliable indicator for the overall diagnostic quality. The mean standard deviation $\bar{\sigma}$ reinforces this grouping even further, with the top three models remaining below 0.47, while most SLMs exceed 0.6, indicating that SLM performance is not only lower on average, but also considerably less stable across circuits.

■ **Table 4** Hit rate, hallucination rate, standard deviation $\bar{\sigma}$ and the weighted score W that were computed.

#	Model	Hit %	Hallucination %	$\bar{\sigma}$	W
1	Gemini 2.5 Flash	79.5	20.5	0.264	1.846
2	Mistral Large	76.9	15.4	0.353	1.832
3	Llama 4 Scout 17B	38.5	7.7	0.467	1.526
4	Mistral 7B v0.3	25.6	61.5	0.653	1.235
5	Gemma-2 2B	20.5	71.8	0.581	1.155
6	Phi-3 Mini 4K	7.7	59.0	0.622	1.076
7	Qwen2.5 1.5B	17.9	74.4	0.708	0.982
8	Llama 3.2 3B	2.6	87.2	0.612	0.944
9	Qwen2.5 3B	7.7	71.8	0.668	0.865
10	StableLM-2 1.6B	2.6	76.9	0.632	0.595
11	Llama 3.2 1B	7.7	94.9	0.593	0.418
12	Phi-2	0.0	84.6	0.584	0.385
13	TinyLlama 1.1B	0.0	76.9	0.544	0.314

The rank-sum results in Table 5 place the models in the same order as the weighted score for all 13 models. This agreement shows that the ranking is robust and independent of the chosen aggregation method and that no single circuit distorts the overall picture.

■ **Table 5** Rank-Sum results across all circuits.

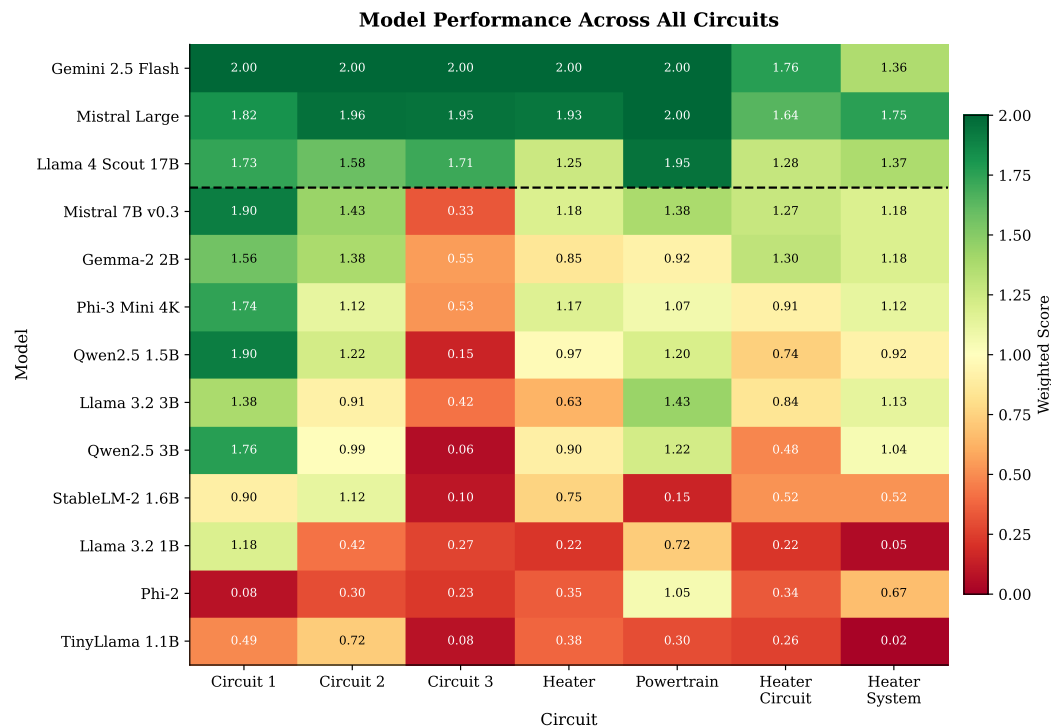
#	Model	C1	C2	C3	PT	H	HC	HS	Rank-Sum
1	Gemini 2.5 Flash	1	1	1	1	1	1	2	8
2	Mistral Large	4	2	2	2	2	2	1	15
3	Llama 4 Scout 17B	7	3	3	3	3	4	3	26
4	Mistral 7B v0.3	2	4	7	5	5	5	4	32
5	Gemma-2 2B	8	5	4	10	6	3	5	41
6	Phi-3 Mini 4K	6	8	5	7	4	6	6	42
7	Qwen2.5 1.5B	3	7	11	6	7	8	9	51
8	Llama 3.2 3B	9	10	6	4	9	7	7	52
9	Qwen2.5 3B	5	9	12	8	8	10	8	60
10	StableLM-2 1.6B	11	6	10	13	10	9	11	70
11	Llama 3.2 1B	10	11	8	11	12	12	12	76
12	Phi-2	13	13	9	9	13	11	10	78
13	TinyLlama 1.1B	12	12	13	12	11	13	13	86

4.2 Per-Circuit Analysis

Figure 9 shows the weighted score of every model on every system. On Circuit 1, the overall performance level is very high, which is consistent with the low complexity of the circuit. The gap between the API-based models and the SLMs is still small here, and several SLMs place ahead of Mistral Large and Llama 4 Scout 17B. On Circuit 2, which extends Circuit 1 with an additional parallel switch, this gap begins to widen. One observation is not visible in the scores: in Scenario 2 of Circuit 1, no model other than Gemini 2.5 Flash identified the theoretically valid, though unlikely, possibility of both L_1 and L_2 being faulty at the same time. All other models instead named only the most probable fault candidates.

At Circuit 3, the most complex of the three electrical circuits, the gap between the API-based models and the SLMs becomes dramatic. Gemini 2.5 Flash, Mistral Large, and Llama 4 Scout 17B all remain above 1.70, maintaining their strong performance from the previous circuits, while all remaining SLMs collapse to scores below 0.60. Model size alone does not explain this collapse: Mistral 7B v0.3, the largest of the locally executed models and fourth in both Circuit 1 and Circuit 2, drops behind Gemma-2 2B and Llama 3.2 3B. Qwen2.5 3B, TinyLlama 1.1B, and StableLM-2 1.6B achieve near-zero scores, indicating an almost complete failure to diagnose faults in this configuration.

On the domain-specific systems, the overall performance recovers noticeably across most models compared to Circuit 3. On the Heater, Gemini 2.5 Flash again achieves the maximum score, and the mid-range SLMs show a relatively gradual decline rather than a collapse. On the Powertrain, Llama 4 Scout 17B achieves its strongest result across all circuits, and Llama 3.2 3B and Mistral 7B v0.3 recover from their poor Circuit 3 performance. For the Heater Circuit, no model achieves a score of 2.00, with Gemini 2.5 Flash and Mistral Large leading at their weakest results so far. A notable result is Gemma-2 2B, which places ahead of both Llama 4 Scout 17B and Mistral 7B v0.3, suggesting that this model handles fluid-based diagnostic scenarios particularly well. On the Heater System, the most complex system evaluated, Mistral Large leads for the first time, while Gemini 2.5 Flash, which was not able to identify all faulty components, places third behind Llama 4 Scout 17B. Llama 3.2 1B and TinyLlama 1.1B achieve near-zero scores, indicating an almost complete failure to handle the complexity of this system.



■ **Figure 9** Weighted Score W of every model on every system. Systems are ordered by increasing complexity from left to right, models by overall weighted score from top to bottom.

4.3 Case Study: Phi-2 on Powertrain

The Powertrain is the only circuit in which Phi-2 reaches a weighted score above 1.0. Across all other circuits, the model scores between 0.08 and 0.67, but on Powertrain it achieves 1.05, performing better than several larger SLMs such as Gemma-2 2B and Llama 3.2 3B. This is particularly surprising given that the Powertrain circuit is structurally more complex than the simpler electrical circuits where Phi-2 failed almost completely.

A closer look at the model’s responses suggests that this strong performance is not the result of genuine reasoning, but rather a reflection of the training data. In all three runs, Phi-2 produces a list of plausible fault causes such as low battery voltage, open switches, faulty motor windings, and high motor resistance. While these match several of the correct fault candidates, the responses also contain clear signs that the model is not actually diagnosing the given scenario. In one run, the response is followed by a section titled “*Follow-up Exercises*” with five additional questions and detailed solutions involving multimeter measurements. In the other two runs, the response ends with an unrelated “*Exercise 2*” about designing a control algorithm for a robotic arm. None of this content was requested by the prompt.

H-bridge and motor control circuits are common examples in introductory electrical engineering, and the structure of Phi-2’s responses, a list of candidate faults followed by exercises and solutions, closely matches the typical layout of such textbooks. This is consistent with the training methodology of the Phi family, which explicitly relies on textbook-like training data [1]. In Phi-2’s case, the high Powertrain score does not indicate strong diagnostic capability, but rather a fortunate overlap between the test scenario and the model’s training data. This raises the question whether benchmark performance on familiar circuit types can be misleading, and reinforces the importance of including less common scenarios in evaluations of diagnostic capability.

5 Conclusions

In this paper, we present the results of an experimental evaluation of the diagnostic capabilities of large and small language models. In this study, we considered 13 different and freely available language models and 7 simple example systems from different domains. Well-known large language models performed best but still exhibit hallucinations and limitations in terms of correctness and completeness. This is worse for small language models. From our evaluation, we obtain that Gemini 2.5 Flash performs best in diagnosis, followed by Mistral Large and Llama 4 Scout 17B. The best small language model was Mistral 7B v0.3, followed by Gemma-2 2B and Phi-3 Mini 4K. Regarding the domains, large language models achieved better results for electric circuits than for heating system diagnosis. SLMs struggled to diagnose more complex electrical circuits.

There are some limitations of the study. First, we conducted the experiments only on smaller example systems. Second, we did the evaluation manually. Hence, there might be a bias in the obtained results. To prevent this, we defined the evaluation categories and applied them as closely as possible, trying to avoid any systematic bias that affects only one circuit or one set of language models. Finally, we limit ourselves to freely available models, not considering commercial ones. It is worth noting that, as shown in the example in the introduction (see Figure 1), ChatGPT was able to perfectly describe potential faults in the first electric circuit. Given the outcomes of other large language models, the reason might be the use of specialized implementations of diagnostic procedures. It would be worth conducting a specific study to determine how ChatGPT or similar large language models perform in diagnosis, considering parametrized circuits of larger sizes. We leave answering this question to future research.

Hence, future research has to cover: (i) extending the study considering more and larger systems from different domains, (ii) applying the study to commercial LLMs, and (iii) studying the effects of combining different diagnosis approaches like model-based diagnosis with different language models. This might be best for small language models that still offer textual communication but have limited diagnostic capabilities. Furthermore, such models can be installed locally and easily integrated into diagnostic processes without requiring communication with servers, thereby enabling data privacy to be more easily ensured.

References

- 1 Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacrose, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini,

- Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone, August 2024. arXiv:2404.14219 [cs.CL]. doi:10.48550/arXiv.2404.14219.
- 2 Deepak Bhaskar Acharya, Karthigeyan Kuppan, and B. Divya. Agentic ai: Autonomous intelligence for complex goals—a comprehensive survey. *IEEE Access*, 13:18912–18936, 2025. doi:10.1109/ACCESS.2025.3532853.
 - 3 Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.
 - 4 Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Physical impossibility instead of fault models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 331–336, Boston, August 1990. Also appears in *Readings in Model-Based Diagnosis* (Morgan Kaufmann, 1992). URL: <http://www.aaai.org/Library/AAAI/1990/aaai90-051.php>.
 - 5 Soodeh Hosseini and Hossein Seilani. The role of agentic ai in shaping a smart future: A systematic review. *Array*, 26:100399, 2025. doi:10.1016/j.array.2025.100399.
 - 6 Sarah Lukens, Matthew Bishof, Nadir Siddiqui, and Destiny West. An Evaluation Framework for Fault Diagnosis Using Technical Manuals in Retrieval-Augmented Large Language Models. *Annual Conference of the PHM Society*, 17(1), October 2025. doi:10.36001/phmconf.2025.v17i1.4549.
 - 7 Md Mahade Hasan, Muhammad Waseem, Kai-Kristian Kemell, Jussi Rasku, Juha Alaranta, and Pekka Abrahamsson. Assessing small language models for code generation: An empirical study with benchmarks. *Journal of Systems and Software*, 236:112815, June 2026. doi:10.1016/j.jss.2026.112815.
 - 8 Hamzah A. A. M. Qaid, Bo Zhang, Dan Li, See-Kiong Ng, and Wei Li. FD-LLM: Large Language Model for Fault Diagnosis of Machines, December 2024. arXiv:2412.01218 [cs.AI]. doi:10.48550/arXiv.2412.01218.
 - 9 Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, April 1987. doi:10.1016/0004-3702(87)90062-2.
 - 10 Peter Struss and Oskar Dressler. Physical negation — Integrating fault models into the general diagnostic engine. In *Proceedings 11th International Joint Conf. on Artificial Intelligence*, pages 1318–1323, Detroit, August 1989. URL: <http://ijcai.org/Proceedings/89-2/Papers/075.pdf>.
 - 11 Anna Sztyber-Betley, Elodie Chanthery, Louise Travé-Massuyès, Silke Merkelbach, Karol Kukla, Maxence Glotin, Alexander Diedrich, and Oliver Niggemann. Are Diagnostic Concepts Within the Reach of LLMs? In Marcos Quinones-Grueiro, Gautam Biswas, and Ingo Pill, editors, *36th International Conference on Principles of Diagnosis and Resilient Systems (DX 2025)*, volume 136 of *Open Access Series in Informatics (OASIs)*, pages 2:1–2:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIs.DX.2025.2.
 - 12 Huanting Wang, Jingzhi Gong, Huawei Zhang, Jie Xu, and Zheng Wang. Ai agentic programming: A survey of techniques, challenges, and opportunities, 2025. doi:10.48550/arXiv.2508.11126.
 - 13 Franz Wotawa. Model-Based Diagnosis—European Perspectives and Contributions. *The European Journal on Artificial Intelligence*, 39(3):557–576, 2026. doi:10.1177/30504554251403464.

Physics-Informed ML Diagnostic Engine: DXC'26 LiU-ICE Track Solution

Antoni Rogowski 

Warsaw University of Technology, Poland

Anna Sztyber-Betley 

Warsaw University of Technology, Poland

Abstract

The paper presents a solution to the DX'26 Competition LiU-ICE Track. The presented approach merges physics insights with data-driven residuals. The applied Grey-Box approach, involving the custom implementation of a recurrent LSTM network with state memory, is well-suited to environments with high inertia. The system does not directly estimate specific values of individual physical quantities, but focuses on determining their rate and direction of change, eliminating the problem of exploding residuals. Furthermore, this approach offers an advantage over classical LSTMs due to its physical representation of states. Structural analysis and the determination of the MSOs enabled the isolation of relationships between the observed variables, whilst simultaneously decoupling the training data from physical quantities unrelated to the variable under investigation. Data decoupling was leveraged to selectively incorporate faulty data into training. The solution implements four virtual sensors, each equipped with its own pair of neural networks. The proposed fault isolation approach is based on a fault diagnosis matrix, fixed alarm thresholds, and scalar product analysis. Additionally, adaptive covariate shift compensation mechanism was introduced to handle changes in external conditions.

The resulting algorithm proved to be highly resilient to measurement noise and capable of accurately analysing the process's variability, scoring True Detection Rate $TDR = 92.26\%$, False Alarm Rate $FAR = 7.86\%$, and True Isolation Rate $TIR = 79.62\%$ on hidden test data.

2012 ACM Subject Classification Computing methodologies → Causal reasoning and diagnostics

Keywords and phrases Diagnosis, Algorithms, Evaluation, LiU-ICE Benchmark, Gray-Box, Physics Informed Neural Networks

Digital Object Identifier 10.4230/OASICS.DX.2026.15

Category DX Competition

Funding This research was partially funded by an internal grant titled “Qualitative modeling of a technological process based on process data and partial models,” awarded in 2025 by the Scientific Council of the Discipline of Automation, Electronic, Electrical Engineering and Space Technologies at the Warsaw University of Technology.

Acknowledgements We extend our gratitude to Professor Janusz Piechna for his consultation and valuable insights on fluid dynamics and engine physics.

1 Introduction

The solution presented here was designed to address a problem that lies in the blind spot between two approaches. On the one hand, there are black-box models [3, 16, 20, 7, 18], which rely entirely on machine learning architectures and completely disregard the physical implementation of the underlying processes. On the other hand, there are PINNs (Physically Informed Neural Networks) architectures [17], which require detailed knowledge of the physical parameters of systems, such as their dimensions or the phenomena occurring within them, often described by complex physical equations, for example, the Navier-Stokes equations.



© Antoni Rogowski and Anna Sztyber-Betley;

licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 15; pp. 15:1–15:20

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The framework implemented in this paper bridges these two extremes by constructing Grey-Box estimators inspired by physical knowledge. Specifically, rather than allowing the neural networks to develop uninterpretable latent representations, we explicitly constrain the hidden states to reflect tangible physical quantities, such as mass and energy accumulation within control volumes. Consequently, the dimensionality of the state space ceases to be a tunable, arbitrary hyperparameter and becomes strictly dictated by the thermodynamic complexity of the monitored subsystem. This methodology aligns with and advances emerging hybrid approaches [8].

DXC competition has a long tradition of inspiring diagnostic solutions [13, 15]. This paper proposes a solution to the LiU-ICE Track (see Section 2), where the task is to diagnose a turbocharged gasoline engine in dynamic conditions under limited data and system knowledge [9]. Earlier versions of the competition were already held at Safeprocess'24 [10] and DX'25 [15]. The paper builds upon earlier solutions [10, 2, 8], introducing *Physics-Informed ML Diagnostic Engine*.

The main contributions of this paper include:

- **Decoupled Grey-Box Architecture (Section 3.1):** Development of parallel neural network-based estimators that take into account the laws of physics, ensuring mass conservation and modelling other real-world constraints. This approach reduces the unpredictability of traditional black-box artificial intelligence without the need for complex, analytical fluid dynamics models.
- **Physics-Informed Feature Engineering (Section 3.2):** Integration of aerodynamic domain knowledge directly into the preprocessing pipeline by deriving analytical features—such as the square root of pressure differentials based on the orifice flow equation.
- **Structurally-Aware Training Strategy (Section 3.3.2):** A novel training paradigm that explicitly exploits the analytical independence of the derived MSO sets. In contrast to conventional residual estimators trained exclusively on fault-free data, this methodology deliberately incorporates selected faulty data into the training pipelines of the decoupled nodes. This forces the neural networks to actively ignore specific cross-coupled disturbances, enhancing fault isolation robustness.
- **Adaptive Covariate Shift Compensation (Section 3.4):** Design of a dynamic mechanism active during a predefined starting phase. This approach neutralizes thermal variations arising from environmental conditions, ensuring robust inference and minimizing False Alarm Rates.
- **Robust Residual Processing and Isolation (Section 3.5 and Section 3.6):** A combination of exponential moving average filtering with vector space isolation logic based on cosine similarity. This ensures effective noise suppression and enables rapid fault isolation.

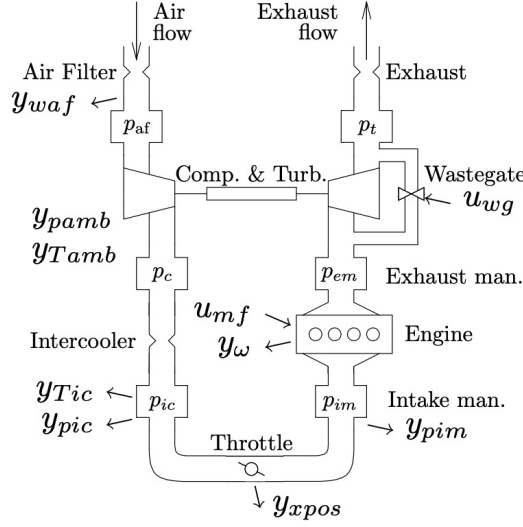
While several of the above aspects were raised in the diagnosis literature, the paper merges them with dedicated physics analysis into the consistent solution to the LiU-ICE benchmark.

2 Competition Description and Problem Formulation

The DX'26 Competition LiU-ICE Track requires developing a robust Fault Detection and Isolation (FDI) system for the air path of a turbocharged internal combustion engine (Figure 1) under strict operational constraints [9]. Table 1 explains the notation and lists symbols used in the paper.

■ **Table 1** Notation used throughout the paper.

Symbol	Description
<i>Measured sensor signals (known variables z)</i>	
y_{pic}	Intercooler pressure
y_{Tic}	Intercooler temperature
y_{pim}	Intake manifold pressure
y_{waf}	Mass flow through the air filter
y_{xpos}	Throttle actuator position
y_{ω}	Engine speed
y_{pamb}	Ambient pressure
y_{Tamb}	Ambient temperature
<i>Control signals</i>	
u_{mf}	Requested injected fuel mass
u_{wg}	Requested wastegate actuator position
<i>Internal (unknown) model variables x in MVEM mean-value engine model</i>	
p_{im}	Intake manifold pressure
p_{ic}	Intercooler pressure
p_{em}	Exhaust manifold pressure
T_{im}	Intake manifold temperature
T_{ic}	Intercooler temperature
T_{em}	Exhaust manifold temperature
V_{im}	Intake manifold volume (constant)
V_{ic}	Intercooler volume (constant)
V_{em}	Exhaust manifold volume (constant)
R	Gas constant of air (constant)
W_c	Compressor mass flow
W_{th}	Throttle mass flow
W_{cyl}	Cylinder mass flow
W_t	Turbine mass flow
W_{wg}	Wastegate mass flow
ω_{tc}	Turbocharger speed
P_c	Compressor power
P_t	Turbine power
J_{tc}	Turbocharger inertia (constant)
<i>Grey-box architecture</i>	
MSO	Minimally Structurally Overdetermined set
x_k	Hidden state vector at sample k
u_k	Input vector at sample k
x_{aug}	Augmented unmodeled state
N_x	Dimension of the hidden state of an MSO subsystem
$\mathbf{u}_{waf}$	Input feature vector for the WAF subsystem
r_k	Residual: $r_k = y_k - \hat{y}_k$
$g(x, u)$	State-transition subnetwork (estimated state derivative $\dot{x}$)
$g_{aug}(x_k, u_k)$	State-transition subnetwork for the augmented state x_{aug}
$h(x, u)$	Output subnetwork acting as virtual sensor
T_s	Euler integration step / sampling period ($T_s = 0.05$ s)



■ **Figure 1** Schematic diagram of the internal combustion engine air path. Connections of the physical components and the placement of key sensors [9].

2.1 Physical and Operational Challenges

The diagnostic architecture must cope with severe nonlinearities, thermal inertia, and actuator lags. A major obstacle is closed-loop fault propagation. Every single fault affects all physical states across the entire air intake system. Furthermore, since obtaining fault data on physical engines is potentially destructive, anomalous training samples are scarce. This forces reliance on nominal data and physical system knowledge. Solutions must also respect strict execution timeouts, necessitating computationally lightweight, two-stage (detection and isolation) architectures suitable for real-time inference.

2.2 Diagnosis Targets

The system must monitor the air flow and, when it detects the fault, it should classify the operating condition as one of the following:

- Multiplicative biases on sensors: air mass flow (f_{waf}), intercooler pressure (f_{pic}), and intake manifold pressure (f_{piim}).
- Physical air leakage in the intake manifold (f_{iml}).
- Unknown fault (f_x)

2.3 Evaluation Criteria

Performance is evaluated using three metrics (normalized 0–1):

- **FAR (False Alarm Rate)**: Percentage of false positive detections during nominal operation.
- **TDR (True Detection Rate)**: Percentage of correctly detected faults.
- **FIA (Fault Isolation Accuracy)**: Average probability assigned to the correct fault class upon detection.

The final ranking is determined by a composite objective function:

$$\text{Total Score} = \frac{(1 - FAR) + TDR + FIA}{3} \quad (1)$$

A naive system (e.g. one that never triggers alarms) achieves a baseline score of around 0.33.

3 Methods

3.1 Grey-Box State-Space Architecture

3.1.1 Hybrid modelling of system dynamics

To avoid the risks of overfitting and poor extrapolation, which are inherent in black-box models that cannot be interpreted from a physical perspective, we adopt a hybrid approach based on state space.

This work adopts the differential modelling framework proposed by [8], where the system dynamics are approximated using a discrete-time state-space Grey-Box model comprising two interconnected neural subnetworks. The state transition network, $g(x, u)$, estimates the system's gradient, and the hidden state is updated iteratively using Euler numerical integration (notation according to Table 1):

$$x_{k+1} = x_k + T_s \cdot g(x_k, u_k). \quad (2)$$

Subsequently, the output network, $h(x, u)$, acts as a virtual sensor, mapping the internal states and control inputs to the estimated measurement:

$$\hat{y}_k = h(x_k, u_k). \quad (3)$$

The raw diagnostic residual r_k , representing the deviation between the actual behavior of the physical system and the prediction of the model, is then calculated as:

$$r_k = y_k - \hat{y}_k. \quad (4)$$

Our architecture strictly adheres to the core structural principles of the Grey-Box methodology introduced by Jung [8]:

- **Structural Decentralization via MSO Sets:** Rather than utilizing a single global recurrent model, the architecture is strictly decentralized into four independent diagnostic tests (MSO sets, see Section 3.1.2).
- **Explicit Physical Constraining:** The hidden states are explicitly constrained to represent tangible physical quantities (e.g., mass and energy accumulation in control volumes). Consequently, the dimensionality of the state space is assigned to match the thermodynamic complexity of each specific MSO (ranging from 1 to 5 states) (details in Section 3.1.2).
- **Real-Time Hardware Synchronization:** The state update frequency (integration step T_s) is strictly dictated by the physical sensor sampling rate. Instead of using an arbitrary numerical step size, the integration process is closely tied to the frequency at which new data arrives from the sensor.

3.1.2 Structural analysis and selection of subsystems

3.1.2.1 Prerequisites

This solution uses structural analysis to find diagnostic tests. *Structural analysis* is a model-based diagnosis technique that abstracts a system of equations into an incidence matrix capturing only which variables appear in which equations, thereby enabling the identification of redundant subsets of equations and the derivation of diagnostic tests for large-scale systems without requiring their explicit analytical solution [1, 12, 6].

A Minimal Structurally Overdetermined (MSO) set, introduced in [12], is a concept important for deriving diagnostic tests. Consider a system described by a model $\mathcal{M}(z, x, f)$, where z denotes the vector of *known variables* (e.g., measured sensor outputs and controlled

inputs), x denotes the vector of *unknown variables* (internal states and algebraic variables), and f is a vector of fault parameters. For any subset of equations $\psi \subseteq \mathcal{M}(z, x, f)$, the *structural redundancy* ρ_ψ is defined as the difference between the number of equations in ψ and the number of unknown variables appearing in them.

A subset ψ is said to be an MSO set if $\rho_\psi = 1$, i.e., it contains exactly one more equation than unknown variables, and no proper subset of ψ is itself overdetermined. An MSO set carries the minimum amount of analytical redundancy needed to perform a consistency check between the model and the measurements [12, 1].

3.1.2.2 Structural model of the engine

The engine MSOs were determined using the Python implementation of the Fault Diagnosis Toolbox [5]. While the original LiU-ICE benchmark provides a comprehensive Differential-Algebraic Equation (DAE) system of 94 relations [9], we utilized a condensed core version of the Mean-Value Engine Model (MVEM) [4]. By focusing strictly on air-path dynamics and turbocharger energy balances, and omitting auxiliary algebraic constraints, the core model was reduced to 21 physical relations, 18 unknown variables, and 10 measurable signals. With a redundancy degree of 3, this formulation yielded 27 Minimally Structurally Overdetermined (MSO) sets. The incidence matrix for MVEM is presented in Table 2.

MSO 0, MSO 1, and MSO 10 were selected to achieve a full fault isolability while minimizing the computational complexity. However, because purely structural MSO subsets struggled to isolate the air mass flow fault, an additional, dedicated residual (r_{waf}) was added for the detection of f_{waf} . The complete computation pipeline for all four diagnostic residuals is summarized in Figure 2.

Although the MSO sets dictate the input variables for each subsystem, feeding raw sensor data directly into the neural estimators often results in degraded predictive performance due to the highly non-linear nature of the engine's fluid dynamics. To address this, Physics-Informed Feature Engineering was employed prior to inference. For example, the input data for the WAF subsystem was non-linearly transformed using logarithmic mappings of engine rotational speed and throttle position (e.g., $x_1 = \log(\omega + \epsilon)\dot{m}_{air}$) (details in Figure 2).

3.1.3 Selected diagnostic tests

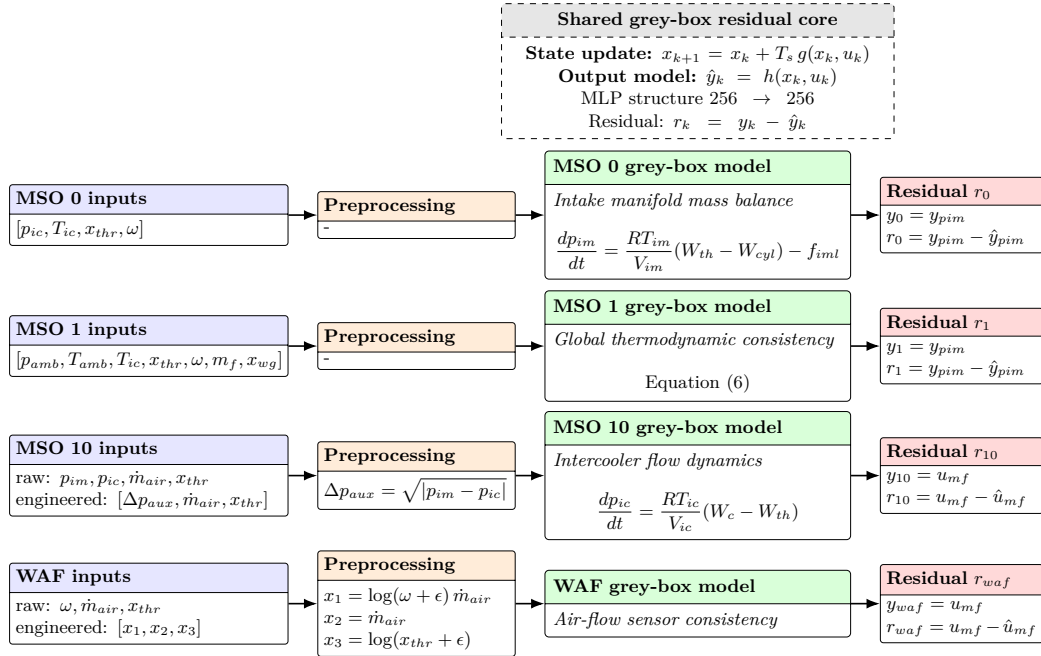
The MSO 0 subsystem ($N_x = 1$) models the local mass balance of the intake manifold. The fact that this subsystem is independent of the mass flow measurement (y_{waf}) and the turbine dynamics makes it robust to disturbances from the “cold side” of the system, allowing for a stable assessment of the manifold's tightness. The hidden state here follows the mass conservation equation:

$$\frac{dp_{im}}{dt} = \frac{R \cdot T_{im}}{V_{im}}(W_{th} - W_{cyl}) - f_{iml}. \quad (5)$$

The MSO 1 subsystem ($N_x = 5$) constitutes the most complex diagnostic node, providing a global thermodynamic consistency check. Notably, structural analysis dictates that this subsystem must remain strictly decoupled from the mass air flow (y_{waf}) and intercooler pressure (y_{pic}) sensors to ensure reliable fault isolation. Consequently, estimating the intake manifold pressure (y_{pim}) based solely on ambient conditions, injected fuel mass, and actuator positions requires the Grey-Box model to internally simulate the entire air-path dynamics. The five-dimensional state space perfectly reflects this physical propagation chain. It encompasses the four core differential variables of the Mean-Value Engine Model (intercooler pressure p_{ic} ,

■ **Table 2** Incidence Matrix of the reduced Mean-Value Engine Model (MVEM). Rows represent structural relations (R), while columns correspond to unknown variables (X), known variables (Z), and faults (F). A bullet ($\bullet$) denotes the incidence of a variable in a given relation.

	Unknown Variables (X)																	Known Variables (Z)							Faults (F)								
	W_{af}	W_c	W_{th}	W_{cyl}	W_t	W_{wg}	p_{ic}	$\dot{p}_{ic}$	p_{im}	$\dot{p}_{im}$	p_{em}	$\dot{p}_{em}$	T_{ic}	T_{em}	ω_{tc}	$\dot{\omega}_{tc}$	P_c	P_t	y_{waf}	y_{pic}	y_{pim}	y_{Tic}	y_{expos}	y_{ω}	y_{pamb}	y_{Tamb}	u_{wg}	u_{mf}	f_{waf}	f_{pic}	f_{pim}	f_{iml}	
R_0	•																		•												•		
R_1	•	•																															
R_2		•					•			•						•										•	•						
R_3		•		•																													
R_4							•			•																							
R_5							•																										
R_6														•																			
R_7				•						•																							
R_8					•					•				•																			
R_9				•	•						•																						
R_{10}										•																							
R_{11}										•																							
R_{12}					•	•						•			•																		
R_{13}											•	•																					
R_{14}				•											•																		
R_{15}					•						•					•																	
R_{16}						•					•					•										•							
R_{17}							•					•				•										•							
R_{18}		•																•								•							
R_{19}							•																			•	•						
R_{20}																•	•																



■ **Figure 2** Four implemented Grey-box model residuals.

manifold pressure p_{im} , exhaust pressure p_{em} , and turbocharger speed ω_{tc}), supplemented by an augmented state (x_{aug}) that explicitly captures unmodeled inertias, such as exhaust thermal capacity and wastegate mechanical delays:

$$\frac{d}{dt} \begin{bmatrix} p_{ic} \\ p_{im} \\ p_{em} \\ \omega_{tc} \\ x_{aug} \end{bmatrix} = \begin{bmatrix} \frac{R \cdot T_{ic}}{V_{ic}} (W_c - W_{th}) \\ \frac{R \cdot T_{im}}{V_{im}} (W_{th} - W_{cyl}) \\ \frac{R \cdot T_{em}}{V_{em}} (W_{cyl} + u_{mf} - W_t - W_{wg}) \\ \frac{P_t - P_c}{J_{tc} \cdot \omega_{tc}} \\ g_{aug}(x_k, u_k) \end{bmatrix}. \quad (6)$$

The MSO 10 subsystem ($N_x = 1$) focuses on intercooler dynamics, enabling rapid verification of flow continuity before the throttle:

$$\frac{dp_{ic}}{dt} = \frac{R \cdot T_{ic}}{V_{ic}} (W_c - W_{th}). \quad (7)$$

Input data preprocessing was described in Section 3.2.1.

Integrated WAF Subsystem. ($N_x = 1$). This diagnostic node differs slightly from the previously described MSO subsystems. Initially, the diagnostic architecture was designed to rely exclusively on the derived MSO subsets for full fault isolability. However, preliminary model evaluations revealed that purely structural residuals struggled to consistently isolate mass air flow faults under highly transient operating conditions due to the severe nonlinearities. To address this fundamental limitation, we explicitly adopted the physics-informed feature engineering approach proposed by Corrini et al. [10]. The input vector $\mathbf{u}_{waf}$ directly utilizes the logarithmic transformations validated by the authors—specifically the interaction between the mass air flow and the logarithm of engine speed, derived from the state-space equations of a turbocharged engine, to linearize the flow dynamics (see also Section 3.2.2):

$$\mathbf{u}_{waf} = \begin{bmatrix} \log(y_\omega + \epsilon) \cdot y_{waf} \\ y_{waf} \\ \log(y_{xpos} + \epsilon) \end{bmatrix}, \quad \epsilon = 10^{-6}. \quad (8)$$

3.2 Physics Based Data Preprocessing

The effectiveness of machine learning algorithms in the diagnosis of non-linear systems depends critically on the appropriate representation of input features.

3.2.1 Implementation of non-linear flow relationships

In accordance with the structure of the MSO 10 subsystem, as determined during the structural analysis phase, the key element is the flow dynamics through the throttle. Instead of forcing the neural network to learn the non-linear relationship between flow and pressure, an auxiliary feature Δp_{aux} was introduced:

$$\Delta p_{aux} = \sqrt{|p_{ic} - p_{im}|} \quad (9)$$

This transformation directly reflects the simplified equation for flow through a constriction. This reduces the complexity of the network's optimisation task, allowing it to focus on approximating the variable flow coefficient rather than the square root non-linearity itself.

3.2.2 Feature engineering for the WAF subsystem

For the flow sensor fault detection module, pre-processing based on the analysis of state variable interactions was applied, strictly following the methodology proposed by Corrini et al. [10]. The input vector was constructed using a combination of raw sensor data and engineered synthetic features:

- The non-linear interaction between engine speed and air mass flow: $\log(y_\omega + \epsilon) \cdot y_{waf}$.
- The raw, untransformed mass air flow signal: y_{waf} .
- The linearized throttle position signal: $\log(y_{xpos} + \epsilon)$.

The use of logarithmic transformations allows for the decoupling of the highly non-linear exponential relationships prevailing in the turbocharging system. This explicit linearization significantly improves the convergence and learning stability within the Grey-Box neural network framework.

3.3 Truncated Backpropagation Through Time Training Scheme

Soft sensors training pipeline was implemented in PyTorch [14].

3.3.1 Optimisation and time horizon

The process of training the parameters of the g and h subnetworks was carried out using the backpropagation algorithm. Due to the recursive nature of Grey-Box models and the need to maintain gradient stability, the Truncated BPTT (TBPTT) [19] technique was employed. The training data was organised into batches of size 16, with each batch comprising 600 samples. At a sampling rate of 20 Hz, this corresponds to a fixed time horizon of 30 s.

The Adam optimiser [11] was used as the optimisation algorithm with an initial learning rate of $5 \cdot 10^{-4}$. To fine-tune the weights in the advanced stages of training, dynamic step reduction was implemented, which reduced the learning rate by 3% every 10 epochs ($\gamma = 0.97$). Empirical evaluation confirmed that this specific scheduling yielded the most stable convergence and the lowest validation error across the entire provided benchmark dataset, preserving a non-vanishing step size even after 1000 epochs. Furthermore, to ensure robust optimization against high-frequency sensor noise and prevent weight oscillations induced by large outlier residuals, the Mean Absolute Error (MAE) was chosen as the training loss function in preference to the standard Mean Squared Error (MSE).

To prevent overfitting, the training process, scheduled for a maximum of 2000 epochs, was coupled with an Early Stopping mechanism. Training was stopped if the loss function value did not improve (by at least 10^{-6}) over a defined patience period of 80 epochs. The networks typically achieved convergence between the 1200th and 1800th epochs. The final convergence epochs and the corresponding loss values for all generated diagnostic nodes are summarized in Table 3. Throughout the entire process, the network state corresponding to the lowest loss value obtained for the training dataset suited specific to each MSO was monitored and recorded each time, ensuring that models with the highest achievable result were obtained.

3.3.2 Decoupled Dataset Composition

A critical aspect of training the decentralized diagnostic nodes was the deliberate composition of the training datasets. While standard residual generators are typically trained exclusively on nominal, fault-free data, this architecture leverages the structural independence properties derived from the MSO sets to enhance isolation robustness.

■ **Table 3** Training convergence statistics for the Grey-Box diagnostic subsystems. The training process utilized an Early Stopping mechanism with a patience of 80 epochs for all models. Note that the Final Loss is computed on normalized data uniformly scaled to the $[0, 1]$ range.

Subsystem	State Dim. (N_x)	Max Epochs	Stopping Epoch	Final Loss (MAE)
MSO 0	1	2000	1803	0.00685
MSO 1	5	2000	1485	0.00522
MSO 10	1	2000	1288	0.00940
WAF Subsystem	1	2000	1451	0.00697

If structural analysis dictates that a specific subsystem is perfectly decoupled from a particular sensor, feeding the network with datasets containing artificial faults from that sensor forces the Grey-Box model to actively learn to ignore those specific disturbances. Consequently, each sub-model was trained on a unique dataset combination, comprising both healthy operation data and selected faulty sequences. The exact composition of the training sets for each diagnostic node is detailed in Table 4.

■ **Table 4** Composition of training datasets for individual subsystems. Models were intentionally exposed to specific fault scenarios (f) from which they are structurally decoupled.

Subsystem	Nominal Data	Injected Fault Data	Structural Justification
MSO 0	NF	f_{waf}	Manifold dynamics structurally decoupled from mass air flow sensor.
MSO 1	NF	f_{waf}, f_{pic}	Global thermodynamic model forced to ignore cold-side sensor anomalies.
MSO 10	NF	f_{waf}, f_{iml}	Intercooler dynamics decoupled from mass air flow and downstream manifold leakage.
WAF Sub.	NF	f_{pic}, f_{pim}	Targeted flow estimator explicitly decoupled from pressure sensor deviations.

3.4 Environmental adaptation

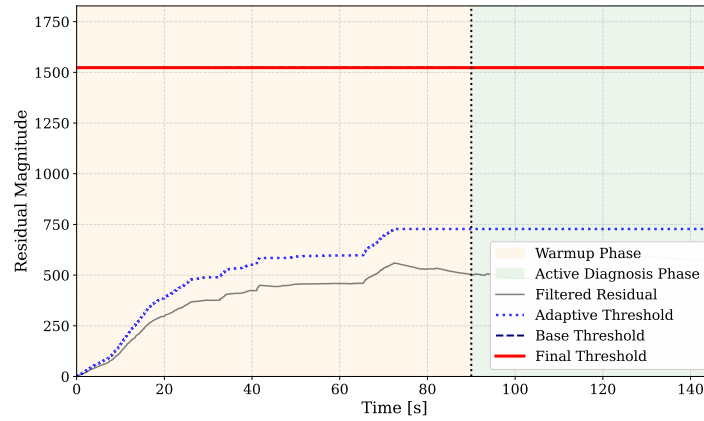
Each incoming sample is mapped on-the-fly to the required input vectors and subjected to the physics-informed transformations defined during the pre-processing phase. The estimated state values are then compared with the actual system readings to generate the diagnostic residual vectors (see Figure 4).

An innovative feature of this streaming workflow is the heuristic decision threshold adaptation mechanism applied specifically to the MSO 1 subsystem. In automotive air-path systems, ambient temperature (y_{Tamb}) and ambient pressure (y_{pamb}) act as absolute boundary conditions that fundamentally dictate intake air density. Consequently, they cannot be simply omitted, any fluctuation in these environmental factors cascades through the thermodynamic equations, significantly impacting baseline mass flows and pressure drops.

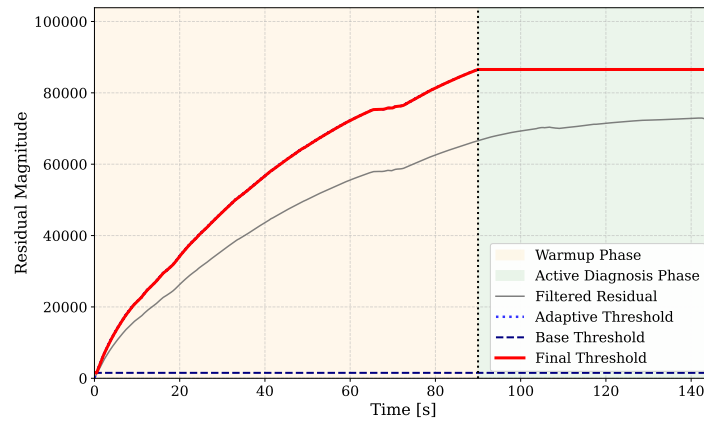
To compensate for the resulting Covariate Shift, the system suspends the generation of diagnostic alerts for the first 90 seconds (1800 samples) of the driving cycle. During this initialisation phase, it actively analyses the baseline prediction error induced by the current

environmental conditions. The highest recorded deviation during this window establishes the new baseline for the adaptive alarm threshold, supplemented by a 30% safety margin. The adaptive thresholding mechanism was illustrated in Figure 3.

Crucially, this adaptive mechanism was deliberately omitted for the remaining subsystems (such as the MSO 0). Since MSO 0 relies heavily on highly dynamic, flow-restricting components like the throttle, calibrating its threshold during a relatively steady-state initialisation phase would lead to a severe underestimation of transient errors. This would inevitably result in a flood of false alarms during sudden throttle openings later in the cycle due to the problem of lack of enforcement.



(a) Training conditions (FTP-75).

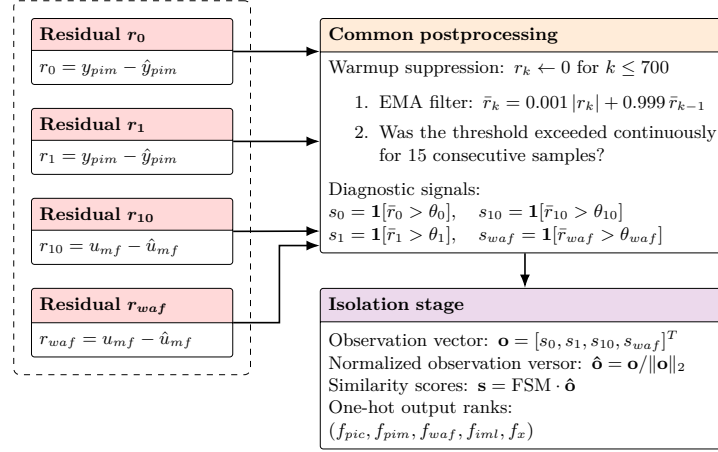


(b) New conditions (Synthetic drift).

■ **Figure 3** Comparison of the threshold adaptation logic: In standard training conditions (a), the system maintains the robust safety floor. Under significant covariate shifts (b), the adaptive candidate successfully supersedes the base floor, becoming the final active threshold to maintain diagnostic sensitivity without increasing false alarms. To simulate this covariate shift in ambient conditions, synthetic data featuring altered ambient temperature and pressure was generated, strictly mirroring realistic operational bounds.

3.5 Robust Detection Mechanism

To make the decision matrix independent of stochastic sensor noise and transients in fluid dynamics, a two-stage filtering architecture was implemented (Figure 4):



■ **Figure 4** Fault detection and isolation mechanism. The diagnostic residuals are computed according to the architecture detailed in Figure 2.

1. EMA (Exponential Moving Average) filter: Raw residuals from the model outputs are smoothed in real time using a low-pass filter with a heuristically determined time constant ($\alpha = 0.001$), which mitigates the effect of high-frequency measurement noise (see Figure 5).
2. Inertial Filter (Debounce Filter): The signal indicating that the filtered residual has exceeded the alarm threshold is verified by an error persistence counter. The alarm is only confirmed if the threshold is exceeded continuously for 15 consecutive samples (0.75 s).

Such a rigorous configuration eliminates alarms generated by micro-jumps and the phenomenon of chattering at the decision threshold. Although the use of an inertial filter slightly increases the time to detect, this is an acceptable trade-off resulting from the prioritisation of reducing the false alarm rate.

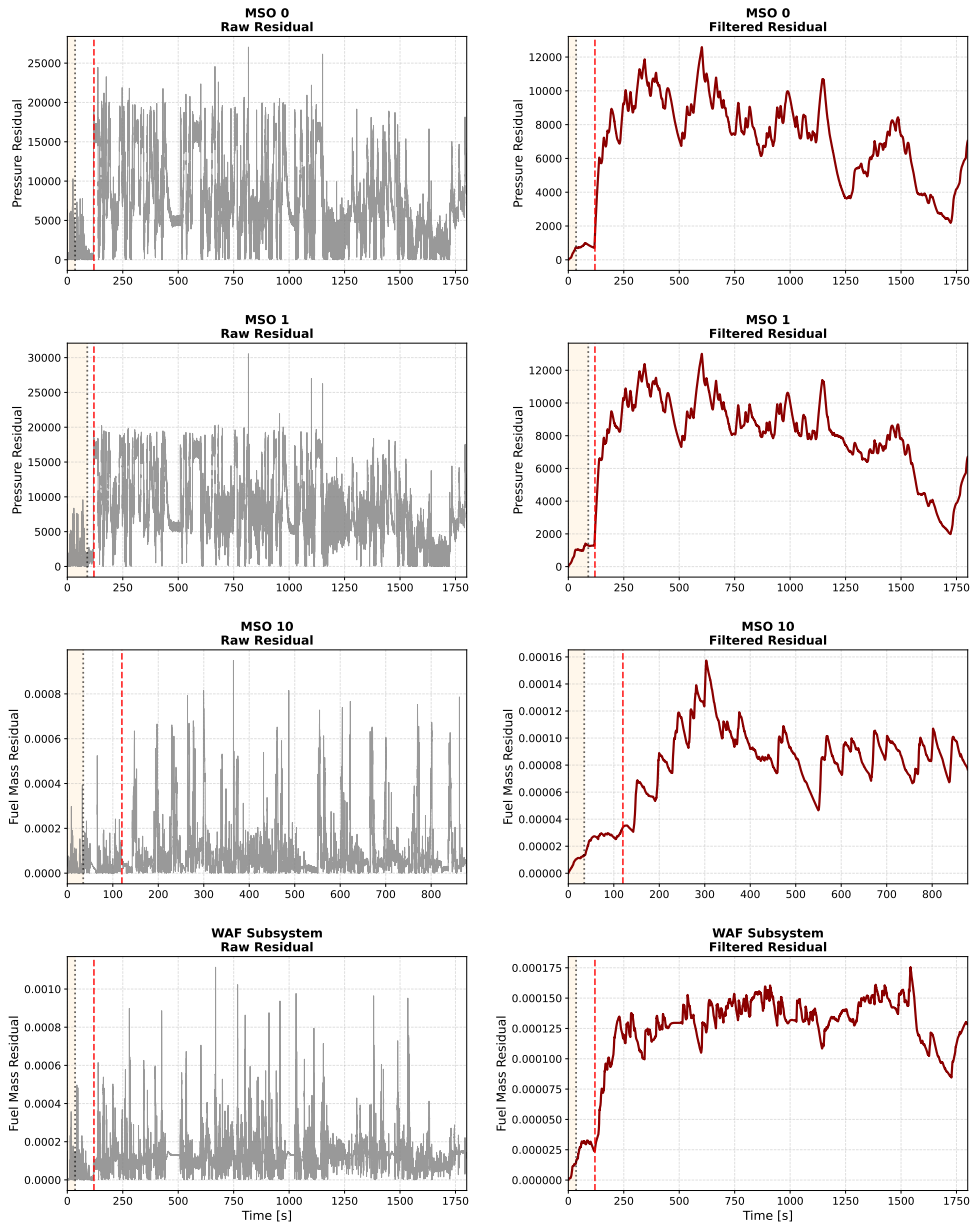
3.6 Fault Isolation and Vectorized Signature Resolution

The fault isolation strategy constitutes one of the most critical stages in the proposed architecture, directly driving the overall performance in the DX Competition. The official evaluation metric strictly balances the False Alarm Rate (FAR), True Detection Rate (TDR), and Fault Isolation Accuracy (FIA), as defined in Equation (1). To maximize the FIA, the diagnostic logic must resolve the boolean threshold flags from the four independent subsystems ($s_0, s_1, s_{10}, s_{waf}$) into a single definitive fault classification.

3.6.1 Cosine Similarity Resolution

The isolation mechanism is triggered only after the active alarms pass a strict inertial debounce filter, which requires the threshold violation to persist continuously for at least 15 samples (0.75 s) to prevent decision chatter. Once confirmed, the active flags form an observation vector $\mathbf{o} = [s_0, s_1, s_{10}, s_{waf}]^T$.

To compare this empirically observed symptom pattern against the theoretically encoded fault dependencies, a cosine similarity approach is employed. The observation vector is first L^2 -normalized into a unit vector ($\hat{\mathbf{o}} = \mathbf{o} / \|\mathbf{o}\|_2$) and subsequently projected onto the reference



■ **Figure 5** Comprehensive comparison of diagnostic residuals before (left column) and after (right column) Exponential Moving Average (EMA) filtration. The plots demonstrate the critical necessity of low-pass filtering in suppressing high-frequency sensor noise and transient spikes. MSO 0, MSO 1, and the WAF Subsystem are evaluated under an intake manifold leakage (f_{iml}) scenario, while MSO 10 highlights the response to an intake manifold pressure sensor bias (f_{pim}). A general initialization phase of 35 seconds (700 samples) is applied to all nodes to stabilize internal states and EMA filters, which is strictly necessary to prevent spurious false alarms during system startup. Furthermore, MSO 1 utilizes an extended 90-second environmental adaptation window to calibrate its dynamic threshold against ambient conditions. For each respective subsystem, the end of its required warmup phase is delineated by a vertical black dotted line. The red dashed line at $t = 120$ s marks the onset of the injected fault.

Fault Signature Matrix (FSM) via the dot product:

$$\mathbf{s} = FSM \cdot \hat{\mathbf{o}} \quad (10)$$

where $\mathbf{s}$ represents the array of similarity scores across the known fault classes.

3.6.2 Hardmax Optimization and Troubleshooting Trade-off

The final isolation output is strictly binarized by applying an *argmax* function to the score vector $\mathbf{s}$, assigning a value of 1.0 to the most probable candidate and zeroing out all remaining possibilities. From a practical engineering perspective, outputting a continuous probabilistic distribution of suspected faults would be highly beneficial for real-world mechanical troubleshooting, as it preserves critical information regarding secondary fault suspicions. However, the implementation of this definitive “hardmax” algorithm was a deliberate design choice tailored specifically to maximize the competition’s scoring function, which inherently rewards unambiguous isolation decisions.

3.6.3 Handling Unknown Faults

A crucial requirement of the benchmark is the ability to flag unknown anomalies (f_x). To prevent the system from forcibly assigning a novel anomaly to one of the four predefined classes, explicit structural safeguards were coded into the matrix logic. If the total similarity score evaluates to zero, or if a specifically unmodeled symptom pattern emerges (e.g., the simultaneous persistent activation of MSO 1, MSO 10, and the WAF subsystem without triggering MSO 0), the algorithm executes a case override. This explicitly routes the diagnosis to the unknown fault vector, ensuring safe and predictable behavior under unforeseen operating conditions.

■ **Table 5** Fault Signature Matrix (FSM) defining the theoretical isolation logic. A bullet (●) indicates that the diagnostic node is sensitive to a specific fault.

Diagnostic Node	Target Fault Scenarios (F)			
	MAF (f_{waf})	Leak (f_{iml})	IC Press (f_{pic})	IM Press (f_{pim})
r_0 (MSO 0)		●	●	●
r_1 (MSO 1)		●		●
r_{10} (MSO 10)			●	●
r_{waf} (WAF Sub.)	●	●		

4 Results

4.1 Computational Complexity and Real-Time Feasibility

The evaluation script was executed in a standard desktop environment. The hardware configuration consisted of an Intel Core i7-10850H CPU (2.70 GHz) paired with 32 GB of RAM, operating on Windows 11 Pro. Notably, the inference was conducted exclusively on the CPU without reliance on a dedicated GPU accelerator.

While the final inference phase of the trained Grey-Box models is exceptionally fast—relying purely on lightweight matrix multiplications and basic numerical integration—the offline training process using Truncated BPTT is computationally intensive. Training each diagnostic subnetwork to convergence required approximately 6 to 10 hours of computation time on the specified CPU setup. However, once trained, empirical measurements during the validation phase revealed an average processing time of 0.82 ms per sample. Because $t \ll 100$ ms, this performance validates the architecture’s compliance with the competition’s timeout limits and its readiness for real-time industrial deployment. This time is as much restricted because the timeouts limit the maximum processing time per sample and not the average. Max from these files is Max: 49.5 ms which is about half of the timeout.

4.2 Evaluation on the shared datasets

The proposed Grey-Box diagnostic architecture was evaluated against the official benchmark datasets provided by the competition organizers. The validation set encompasses two driving cycles: the FTP-75 (Federal Test Procedure) and the WLTP (Worldwide Harmonised Light Vehicles Test Procedure). The datasets contain both nominal fault-free operations (NF) and various fault scenarios injected at different magnitudes (e.g., sensor biases ranging from 0.85 to 1.15, and manifold leaks of 4 mm and 6 mm).

Table 6 presents the aggregated evaluation metrics, averaged across all specific fault magnitudes within each category. The results demonstrate the high robustness of the architecture, an achievement attributed mainly to the decentralized MSO formulation, supported by physics-informed constraints. The system achieved a global Total Score of 92.29%, driven primarily by a low average False Alarm Rate (FAR) of 0.68% and a near-perfect Fault Isolation Accuracy (FIA) of 98.97%. Notably, for the majority of the sensor bias faults (f_{pic} , f_{pim} , f_{waf}), the isolation accuracy reached a perfect 100%, validating the effectiveness of the Cosine Similarity isolation logic combined with the structural decoupling strategy.

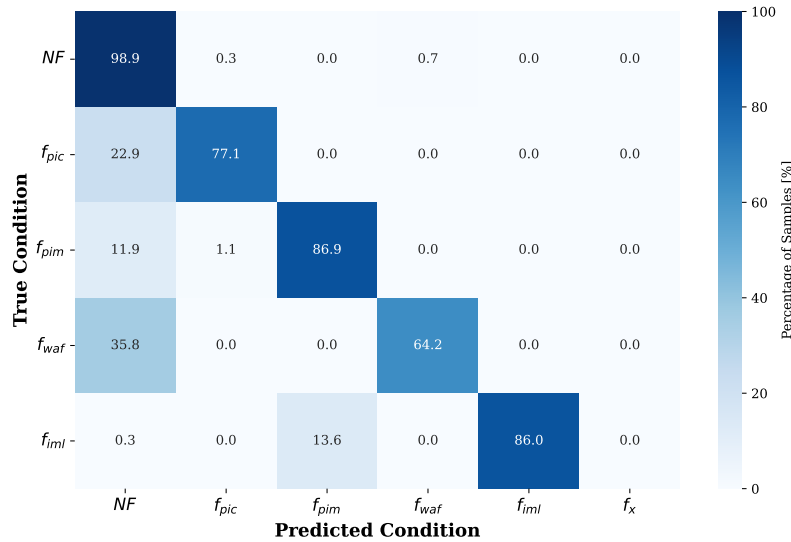


Figure 6 Confusion matrix.

Crucially, the proposed diagnostic architecture is entirely deterministic during the inference phase. The evaluation metrics (FAR, TDR, and FIA) remain perfectly consistent across multiple executions, demonstrating high reproducibility regardless of the underlying operating system or hardware architecture. This hardware-agnostic predictability is a significant advantage for potential real-world deployment. However, it should be noted that the final evaluation score exhibits a minor dependency on the Python runtime environment. As observed during testing, upgrading the interpreter from Python 3.11.8 distribution to version 3.11.12 yielded a slight improvement in the final score. This variation is strictly attributed to low-level differences in floating-point arithmetic handling and underlying library optimizations across Python releases.

■ **Table 6** Averaged evaluation results across the FTP-75 and WLTP driving cycles. Metrics are aggregated by fault class, encompassing all tested fault magnitudes. (Note: TDR for NF indicates the correct classification of nominal state).

Driving Cycle	Fault Class	FAR [%]	TDR [%]	FIA [%]	Score [%]
FTP-75	NF	0.00	100.00	100.00	100.00
	f_{pic}	0.00	78.15	100.00	92.72
	f_{pim}	0.00	85.20	100.00	95.08
	f_{waf}	0.00	57.68	100.00	85.92
WLTP	NF	1.80	100.00	100.00	98.20
	f_{pic}	0.00	76.87	100.00	92.28
	f_{pim}	4.07	89.65	97.53	94.37
	f_{waf}	0.00	67.35	100.00	89.12
	f_{iml}	0.00	99.65	86.30	95.30
Global Average (41 files):		0.68	78.75	98.97	92.29

The performance drop observed in the True Detection Rate (TDR) for specific faults (e.g., the 0.95 and 1.05 bias magnitudes for f_{waf}) is a direct consequence of the rigorous 0.75 s inertial debounce filter, EMA filter and the deliberate tuning of the alarm thresholds. By sacrificing detection sensitivity for micro-faults that closely mimic natural sensor noise, the architecture successfully prevented error propagation and achieved the primary industrial requirement: avoiding false positives under highly transient conditions.

4.3 Evaluation on the Competition Datasets

To rigorously validate the generalisation capabilities of the proposed Decoupled Grey-Box Architecture, the final algorithmic solution was submitted to the competition organisers for an independent, blind evaluation. The system was tested against a set of datasets withheld during the training and validation phases. These hidden datasets span varying driving cycles (WLTP, FTP75 City, FTP75 Highway) and introduce different fault magnitudes, including categorised small and large intake manifold leakages (f_{iml}).

The official performance report returned by the evaluating committee is summarised in Table 7.

4.3.1 Performance Analysis

As evidenced by the official feedback, the architecture exhibits exceptional robustness and accuracy in diagnosing mass air flow (f_{waf}) faults and intake manifold leakages (f_{iml}). Notably, the system handles leakages – which are traditionally difficult to decouple from

■ **Table 7** Official evaluation results provided by the competition organisers. High scores across most scenarios validate the robustness of the architecture, while specific degradation in two datasets highlights operational limits.

Driving Cycle	Fault & Size	TDR [%]	1-FAR [%]	FIA [%]	Score
FTP75 City 2	f_{iml} (large)	99.57	100.00	98.72	99.43
FTP75 City 2	f_{iml} (small)	97.13	100.00	99.33	98.82
FTP75 Highway	f_{iml} (large)	100.00	95.60	100.00	98.53
FTP75 Highway	f_{iml} (small)	96.66	100.00	98.87	98.51
FTP75 City 2	f_{waf} (-12%)	100.00	94.06	64.05	86.04
FTP75 City 2	f_{waf} (+12%)	96.83	100.00	100.00	98.94
FTP75 (Standard)	f_{pim} (-10%)	21.12	85.10	30.70	45.64
FTP75 (Standard)	f_{pim} (+10%)	99.93	100.00	91.25	97.06
FTP75 Highway	f_{pim} (-10%)	98.79	100.00	92.36	97.05
FTP75 Highway	f_{pim} (+10%)	99.22	100.00	100.00	99.74
WLTP	f_{pic} (-3%)	97.91	96.35	80.20	91.48
WLTP	f_{pic} (+8%)	100.00	34.61	0.00	44.87
Overall Average:		92.26	92.14	79.62	88.01

rapid throttle transients – with near-perfect scores (averaging over 98.8%) across both highly dynamic city driving and steady-state highway conditions. The dynamic thresholding and covariate shift adaptation mechanisms successfully maintained a 100% non-false-alarm rate ($1 - \text{FAR}$) in the majority of these independent test scenarios.

4.3.2 Boundary Limitations and Edge Cases

Despite the overall high performance, the blind evaluation exposed two specific boundary conditions where the diagnostic logic degrades. Firstly, a negative drift in the intake manifold pressure sensor (f_{pim} at -10%) under the standard FTP75 cycle resulted in a severely suppressed TDR (21.12%). Interestingly, the exact same fault magnitude was easily detected and isolated under Highway conditions (Score: 97.05). This suggests that the magnitude of a -10% pressure drop becomes indistinguishable from the model’s natural uncertainty bounds during the frequent, low-load deceleration phases characteristic of urban FTP75 driving.

Secondly, a positive drift in the intercooler pressure sensor (f_{pic} at +8%) under the WLTP cycle caused a complete collapse of the isolation logic ($\text{FIA} = 0.00\%$) alongside a high false alarm rate. A 100% TDR indicates that the residual generators correctly flagged an anomaly, but the resultant symptom vector fell outside the expected bounds of the Fault Signature Matrix. This independent finding highlights the extreme sensitivity of the air-path thermodynamics to positive pressure biases upstream of the throttle, identifying a clear direction for future structural improvements.

5 Discussion

Sensitivity to Mass Air Flow Faults

The most significant limitation observed during the evaluation was the relatively lower True Detection Rate (TDR) for the air mass flow sensor (f_{waf}). Despite incorporating specialized feature engineering and logarithmic transformations, the system struggled to consistently distinguish small fault magnitudes (e.g., 0.95 and 1.05) from the high-frequency measurement

noise and model mismatches during steep transients. This suggests that while the integrated WAF module enhances isolability, its detection threshold must remain high to avoid false positives, ultimately sacrificing sensitivity to low-magnitude sensor drifts.

Detection Latency vs. Robustness

To achieve a near-zero False Alarm Rate (FAR), a rigorous 0.75 s (15 samples) inertial debounce filter and high inertia filter were implemented. While this successfully suppressed noise-induced spikes and transient-related residuals, it introduced a significant activation delay. Future work should focus on developing adaptive filtering techniques that can reduce this latency without compromising the system's robustness against measurement noise.

Intake Manifold Leakage Isolation

The isolation of physical air leakage (f_{iml}) showed room for improvement. Enhancing the IML isolation would require a more granular analysis of the leakage-induced pressure drop characteristics.

Strategic Data Utilization and Synthetic Scenarios

To overcome the inherent data scarcity of the benchmark, the proposed methodology leveraged cross-fault training. By feeding diagnostic nodes with data containing faults from which they are structurally decoupled (e.g., exposing MSO 0 to f_{waf} anomalies), the models were trained to actively ignore overlapping disturbances. Furthermore, there is significant potential in artificially synthesizing new, physically plausible fault scenarios.

The Role of Warmup Periods

The implementation of a 90-second (1800 samples) initialization and warmup period was essential for handling covariate shift caused by environmental factors. This period allows the system to calibrate its baseline residuals and adapt its thresholds to the current operating environment. However, this approach introduces a critical diagnostic risk which is fault masking. If a fault—particularly a subtle sensor bias—is present from the moment the engine starts, the adaptation logic might erroneously incorporate this anomaly into its healthy baseline calibration. Furthermore, during this 90-second window, the system's sensitivity is intentionally reduced, creating a temporal blind spot where critical failures could go undetected.

By identifying these bottlenecks, we provide a clear roadmap for future iterations of the system, focusing on the refinement of the feature-driven diagnostic nodes and the optimization of the isolation logic to handle even more subtle fault manifestations.

References

- 1 J-Ph Cassar and M Staroswiecki. A structural approach for the design of failure detection and identification systems. *IFAC Proceedings Volumes*, 30(6):841–846, 1997.
- 2 Austin Coursey, Abel Diaz-Gonzalez, Marcos Quinones-Grueiro, and Gautam Biswas. Data-Driven Fault Detection and Isolation Enhanced with System Structural Relationships. In Marcos Quinones-Grueiro, Gautam Biswas, and Ingo Pill, editors, *36th International Conference on Principles of Diagnosis and Resilient Systems (DX 2025)*, volume 136 of *Open Access Series in Informatics (OASISs)*, pages 15:1–15:17, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASISs.DX.2025.15.

- 3 Miguel Cubero, Luis Ignacio Jiménez, Daniel López, Belarmino Pulido, and Carlos Alonso-González. Towards Predictive Maintenance in an Aluminum Die-Casting Process Using Deep Learning Clustering and Dimensionality Reduction. In Marcos Quinones-Grueiro, Gautam Biswas, and Ingo Pill, editors, *36th International Conference on Principles of Diagnosis and Resilient Systems (DX 2025)*, volume 136 of *Open Access Series in Informatics (OASICS)*, pages 6:1–6:16, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASICS.DX.2025.6.
- 4 Lars Eriksson. Modeling and control of turbocharged si and di engines. *Oil & Gas Science and Technology-revue de l'Institut Français Du Pétrole*, 62:523–538, 2007. URL: <https://api.semanticscholar.org/CorpusID:53476253>.
- 5 Erik Frisk, Mattias Krysander, and Daniel Jung. A toolbox for analysis and design of model based diagnosis systems for large scale models. *IFAC-PapersOnLine*, 50(1):3287–3293, 2017. 20th IFAC World Congress. doi:10.1016/j.ifacol.2017.08.504.
- 6 Maxence Glotin, Louise Travé-Massuyès, and Elodie Chanthery. MSO Sets and MTES for Dummies. In Ingo Pill, Avraham Natan, and Franz Wotawa, editors, *35th International Conference on Principles of Diagnosis and Resilient Systems (DX 2024)*, volume 125 of *Open Access Series in Informatics (OASICS)*, pages 13:1–13:15, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASICS.DX.2024.13.
- 7 Yuxin Huang, Austin Coursey, Marcos Quinones-Grueiro, and Gautam Biswas. Time-series few shot anomaly detection for hvac systems. *IFAC-PapersOnLine*, 58(4):426–431, 2024. 12th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2024. doi:10.1016/j.ifacol.2024.07.255.
- 8 Daniel Jung. Automated design of grey-box recurrent neural networks for fault diagnosis using structural models and causal information. In Roya Firoozi, Negar Mehr, Esen Yel, Rika Antonova, Jeannette Bohg, Mac Schwager, and Mykel Kochenderfer, editors, *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pages 8–20. PMLR, 23–24 June 2022. URL: <https://proceedings.mlr.press/v168/jung22a.html>.
- 9 Daniel Jung, Erik Frisk, and Mattias Krysander. The liu-ice benchmark – an industrial fault diagnosis case study. *arXiv preprint arXiv:2408.13269*, 2024. doi:10.48550/arXiv.2408.13269.
- 10 Daniel Jung, Erik Frisk, Mattias Krysander, Anna Sztyber-Betley, Francesco Corrini, Andrea Arici, Nicolas Anselmi, Mirko Mazzoleni, Jiamin Xu, Siwen Mo, Zixuan Xu, Chongpan Yang, Zhile Du, Hossein Safaeipour, Mehdi Forouzanfar, Vahid Mirahi, Anna Pinnarelli, Vicenç Puig, Qiao Deng, Yufei Liu, Jiakun Liu, Haobin Ke, Wanting Zhu, Silke Merkelbach, Maryam Ahang, and Hodayoun Najjaran. A fault diagnosis benchmark of technical systems with incomplete data — six solutions. *Control Engineering Practice*, 164:106427, 2025. doi:10.1016/j.conengprac.2025.106427.
- 11 Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. arXiv:1412.6980.
- 12 Mattias Krysander, Jan Åslund, and Mattias Nyberg. An efficient algorithm for finding minimal overconstrained subsystems for model-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 38(1):197–206, 2008. doi:10.1109/TSMCA.2007.909555.
- 13 T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, L. Kuhn, J. de Kleer, A. van Gemund, and A. Feldman. First international diagnosis competition-DXC'09. *Proc. Int. Workshop on Principles of Diagnosis DX*, 9:383–396, 2009.
- 14 Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: an imperative style,

- high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8024–8035, Red Hook, NY, USA, 2019. Curran Associates Inc.
- 15 Ingo Pill, Daniel Jung, Eldin Kurudzija, Anna Sztyber-Betley, Michał Syfert, Kai Dresia, Günther Waxenegger-Wilfing, and Johan de Kleer. The DX Competition 2025 and Its Benchmarks. In Marcos Quinones-Grueiro, Gautam Biswas, and Ingo Pill, editors, *36th International Conference on Principles of Diagnosis and Resilient Systems (DX 2025)*, volume 136 of *Open Access Series in Informatics (OASICS)*, pages 14:1–14:19, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASICS.DX.2025.14.
- 16 Kélian Poujade, Louise Travé-Massuyès, Jérémy Pirard, and Laure Vieilleveigne. Unsupervised Multimodal Learning for Fault Diagnosis and Prognosis - Application to Radiotherapy Systems. In Marcos Quinones-Grueiro, Gautam Biswas, and Ingo Pill, editors, *36th International Conference on Principles of Diagnosis and Resilient Systems (DX 2025)*, volume 136 of *Open Access Series in Informatics (OASICS)*, pages 16:1–16:17, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASICS.DX.2025.16.
- 17 M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi:10.1016/j.jcp.2018.10.045.
- 18 Henrik Sebastian Steude, Lukas Moddemann, Alexander Diedrich, Jonas Ehrhardt, and Oliver Niggemann. Diagnosis driven anomaly detection for cyber-physical systems**this research paper is funded by dtec.bw and the european union. *IFAC-PapersOnLine*, 58(4):13–18, 2024. 12th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2024. doi:10.1016/j.ifacol.2024.07.186.
- 19 Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501, 1990. doi:10.1162/neco.1990.2.4.490.
- 20 A. Youssef, H. Noura, A. El Amrani, E. El Adel, and M. Ouladsine. A survey on data-driven fault diagnostic techniques for marine diesel engines. *IFAC-PapersOnLine*, 58(4):55–60, 2024. 12th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2024. doi:10.1016/j.ifacol.2024.07.193.

Counterfactual Fault Responsibility in a Dynamic Structural Causal Model of a DC Motor

Emir Mujić 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Franz Wotawa¹ 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Abstract

Model-based diagnosis can identify fault hypotheses that are compatible with observed abnormal behaviour. Still, it does not necessarily determine which fault is responsible for a particular symptom when several faults are active simultaneously. We propose a counterfactual framework for post-diagnostic fault responsibility analysis in dynamical systems. The physical process is represented as a time-indexed structural causal model, while faults are encoded as interventions on individual system mechanisms. Responsibility is evaluated by comparing the factual system trajectory with counterfactual trajectories in which candidate faults are removed while the operating conditions and control structure are kept fixed. We instantiate the framework on an electro-thermo-mechanical model of a DC motor and evaluate it in two simulated multi-fault scenarios. The results show that counterfactual fault removal can separate the contributions of different faults to individual symptoms and distinguish severity contribution from necessity for a failure event. The proposed approach therefore complements conventional diagnosis with symptom-level causal explanations of faulty system behaviour.

2012 ACM Subject Classification Computing methodologies → Causal reasoning and diagnostics

Keywords and phrases Counterfactual Reasoning, Fault Responsibility, Structural Causal Models, Dynamic Systems, Model-Based Diagnosis

Digital Object Identifier 10.4230/OASICS.DX.2026.16

Category Short Paper

Funding This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/COE12.

1 Introduction

Fault diagnosis in dynamical systems commonly addresses two related tasks: detecting that the system has departed from nominal behaviour and isolating the fault or set of faults that is compatible with the observed measurements. Standard fault detection and isolation methods typically achieve this through residuals, analytical redundancy relations, observers, or structural relations between system variables. These methods are well suited to determining whether a fault is detectable or distinguishable from other faults. However, once several faults are simultaneously active, identifying the active fault set does not necessarily explain which fault is responsible for a particular symptom or failure event. This distinction is important because compatibility, detectability, and causal responsibility are different notions. A fault may produce a nonzero residual or contribute to abnormal behaviour without being necessary for a specified event. Conversely, a fault may be largely hidden in the measured output by feedback control while still being responsible for increased actuator effort, thermal

¹ Authors appear in the alphabetic order.



stress, or the crossing of a safety threshold. Residual magnitudes alone therefore do not generally establish whether a failure would still have occurred had one of the active faults been absent.

We address this problem using counterfactual reasoning. Rather than asking only whether a fault hypothesis is consistent with the observations, we ask whether the observed symptom would still have occurred under an intervention that removes the fault while preserving the remaining operating conditions. This requires a model in which faults modify the physical mechanisms that generate the system trajectory. Structural causal models (SCMs) provide such a representation through modular structural assignments and explicit interventions.

In this work, a dynamical system is represented as a time-indexed structural causal model derived from its physical equations. Electrical and mechanical faults are encoded as interventions on individual mechanisms rather than as direct changes to measured variables. Starting from a factual multi-fault trajectory, we generate counterfactual trajectories in which candidate faults are removed individually while the initial conditions, reference task, load profile, and controller remain fixed. The resulting comparisons provide a post-diagnostic explanation of fault responsibility. They can distinguish whether a fault contributes to the magnitude of a symptom, is mainly responsible for one symptom rather than another, or is necessary for a failure event to occur. The proposed method therefore complements conventional FDI. FDI supplies candidate fault mechanisms, while counterfactual analysis explains their symptom-specific consequences after diagnosis.

We demonstrate the approach using a DC motor model. The model contains electrical, mechanical, and thermal feedback mechanisms and therefore provides a suitable setting for studying fault propagation under closed-loop control. Two simulated multi-fault experiments are used to evaluate whether counterfactual fault removal can separate responsibility for angle-synchronous disturbances and overheating.

The main contributions of this paper are (1) the formulation of a physical DC motor model as a dynamic SCM, (2) the representation of electrical and mechanical faults as interventions on physical mechanisms, (3) a counterfactual fault-removal procedure for assigning symptom-specific responsibility.

2 Related Work

In Reiter's theory of diagnosis [13], a diagnosis is a set of components assumed to be abnormal such that the system description is consistent with the observations. De Kleer and Williams extended this framework to multiple-fault diagnosis through the General Diagnostic Engine [5], while later work studied minimal and kernel diagnoses and structured system descriptions [4, 3]. These approaches provide rigorous methods for generating candidate fault explanations, but primarily ask which abnormality assumptions are consistent with the observations rather than whether a fault was responsible for a particular failure event.

Abductive diagnosis places greater emphasis on explanation by selecting hypotheses that imply the observed behaviour. Poole proposed an early general framework for abductive explanation and prediction [12], while Console and Torasso integrated models of correct and faulty behaviour [1]. Friedrich, Gottlob, and Nejdl further connected diagnosis with decisions and therapy [6]. Cordier et al. [2] relate AI-based model-based diagnosis to control-based fault detection and isolation (FDI). Both traditions identify faults through violations of healthy-system relations, although AI diagnosis commonly uses symbolic component models and logical conflicts, whereas FDI typically uses numerical dynamic models and residuals.

Structural methods extend FDI to larger dynamical systems by reasoning about the connections between variables and equations, sometimes without requiring a complete numerical model. They have been applied to fault detectability and isolability, sensor placement, and residual-generator selection [14, 15]. In contrast, our work assumes that relevant fault mechanisms are available and uses counterfactual simulation to determine which fault is responsible for a specified symptom.

Causal models provide a natural representation of faults as changes to system mechanisms. Pearl's structural causal model framework formalizes interventions and counterfactuals [10], while Halpern and Pearl define actual causality in terms of whether an event remains necessary under suitable counterfactual changes [8]. This perspective is directly relevant to fault responsibility analysis, as shown in the work of Weiblich et al. [16], where they construct counterfactual trajectories and score them to indicate whether the failure was removed. We employ a similar idea, however using an SCM constructed .

3 Methods

In this section, we describe the dynamic SCM used for the DC motor. A DC motor is a useful example of a causal dynamical system because its main mechanisms are known from physics: voltage affects armature current, current generates torque, torque changes angular velocity and rotor angle, and the resulting current and motion generate heat. Following the view of causal dynamical systems in [11], we model the plant dynamics as deterministic structural equations and add noise only at the measurement level.

As is the practice in causal reasoning, we make an assumption on the world and the model around the level of abstraction needed for the task. We do not explicitly model microscopic transient effects inside the motor, but rather focus on second-order dynamics. We also assume that a model of the motor is available, as well as the knowledge of how certain faults affect the motor.

The structure of the causal graph and variable ordering is defined by the physical direction of the mechanisms when considering a motor operating regime. The input voltage affects the armature current through the electrical dynamics. The current affects the produced torque, which affects the angular velocity and rotor angle. The angular velocity also causes back-EMF and opposes changes in the current, creating a primary loop in the dynamics. The current also generates heat through resistive losses, and angular velocity generates heat through friction, causing the winding temperature to change the armature resistance. The winding temperature then affects the armature resistance, closing an electro-thermal feedback path in the causal graph.

3.1 Structural Causal Models

A Structural Causal Model (SCM) [10, 8, 7] represents a system in terms of causal mechanisms for each variable. The central property of an SCM is modularity. Each structural assignment represents an autonomous mechanism that can be modified without changing other mechanisms in the model. An intervention in an SCM $do(X_k = x)$, replaces the original structural assignment for variable X_k by the constant assignment $X_k := x$. This distinguishes intervention from conditioning: observing $X_k = x$ does not change the data-generating mechanism, whereas $do(X_k = x)$ does.

SCMs support counterfactual reasoning. A counterfactual asks what would have happened under a different intervention, given the same factual situation. Conceptually, this is evaluated by first inferring the relevant background conditions from the factual observation, then

modifying the intervened mechanism, and finally evaluating the modified model under the same background conditions. In this paper, we use a domain-specific counterfactual fault removal criterion to distinguish severity contribution from necessity for a symptom event.

To correctly define a counterfactual on a DC motor, we use an SCM. Since the DC motor is a dynamical system, we use a time-indexed SCM. Instead of a classic static assignment where any variable X_i in an SCM is defined by a function of its parents Pa_i and exogenous inputs U_i , $X_i := f_i(\text{Pa}_i, U_i)$, the model specifies how the variable X_i evolves from one time step to the next:

$$X_{t+\Delta t}^i := f(X_t^i, \text{Pa}_t^i, U_t^i), \quad (1)$$

where now the variable is dependent on its previous state as well as the time-indexed states of the parents and exogenous variables. This connects SCMs with classical state-space and differential-equation models; the equations describe physical time evolution, while the causal interpretation specifies which mechanisms affect how and which each variable is affected.

3.2 A causal dynamical model of a DC motor

As our plant model, we develop a custom model of a DC electric motor. The model is a dynamic, nonlinear, multivariate system that falls into the category of dynamic structural causal models with noise added to the measured values, as detailed in [11]. As a starting point, we use well-known descriptions of the basic DC motor behaviour:

$$L \frac{di_a(t)}{dt} = u_{in}(t) - R_a(t)i_a(t) - k\omega(t), \quad (2)$$

$$J \frac{d\omega(t)}{dt} = ki_a(t) - \tau(\omega) - M_L(t). \quad (3)$$

where $u_{in}[V]$ is the input voltage, $R_a[\Omega]$ is the coil armature resistance, $i_a[A]$ is the armature current, $k[V \cdot s]$ ² is the back-EMF constant, $\omega[rad/s]$ is the angular velocity, and $L[H]$ is the inductance. $J[kgm^2]$ is the moment of inertia of the rotor, $\tau[Nm]$ is the friction torque and $M_L[Nm]$ is the load torque.

We model the friction $\tau(\omega)$ as a Stribeck curve. It consists of three parts: the startup, mixed, and running friction. General mechanical friction is given by the following formula as opposing torque:

$$\tau(\omega) = \left(\tau_C + (\tau_{max} - \tau_C) \cdot e^{-\left(\frac{|\omega|}{\omega_C}\right)^2} \right) \cdot \tanh\left(\frac{\omega}{\epsilon}\right) + k_r \cdot \omega. \quad (4)$$

In Equation 4, τ_C is the Coulomb friction, τ_{max} the maximum friction value, ω is the angular velocity, ω_C is a value of ω at which we switch from the startup to mixed friction, k_r is the viscous friction coefficient, and ϵ is a small value used to saturate the $\tanh(\cdot)$ function to make sure that the friction is always opposing rotation.

The thermal part of the model is added because the winding resistance is not constant, but changes with the winding temperature. This allows the model to account for overheating. The temperature-dependent armature resistance is modelled as

$$R_a(t) = R_0 (1 + \alpha_{Cu}(T_w(t) - T_{ref})), \quad (5)$$

² The back-EMF constant k in both the electrical and mechanical equations is equivalent in SI units: $\left[\frac{Nm}{A}\right] = \left[\frac{J}{C}\right] = \left[\frac{J}{C}\right][s] = [Vs]$.

where R_0 is the nominal armature resistance, $\alpha_{Cu}[1/K]$ is the copper temperature coefficient and T_{ref} is the reference temperature. The copper losses are modelled as $P_{Cu}(t) = i_a^2(t)R_a(t)$.

The winding temperature is then described as

$$C_w \frac{dT_w(t)}{dt} = P_{Cu}(t) - \frac{T_w(t) - T_h(t)}{R_{wh}}, \quad (6)$$

where $C_w[J/K]$ is the winding thermal capacitance, $T_w(t)[K]$ is the winding temperature, $T_h[K]$ is the housing temperature, and $R_{wh}[K/W]$ is the thermal resistance between the winding and the housing. The housing temperature is modelled as

$$C_h \frac{dT_h(t)}{dt} = P_h(t) + \frac{T_w(t) - T_h(t)}{R_{wh}} - \frac{T_h(t) - T_a}{R_{ha}}, \quad (7)$$

where C_h is the housing thermal capacitance, R_{ha} is the thermal resistance between the housing and ambient, and T_a is the ambient temperature. The term $P_h(t)$ represents the heat generated by mechanical losses. Power lost to mechanical movement is proportional to viscous friction, so we approximate it by a scaling factor $P_h(t) = \eta_h k_r \omega^2(t)$, where η_h is the scaling factor.

The full electro-thermo-mechanical model is given by the state vector:

$$x(t) = [i_a(t) \quad \omega(t) \quad \theta(t) \quad T_w(t) \quad T_h(t)]^T, \quad (8)$$

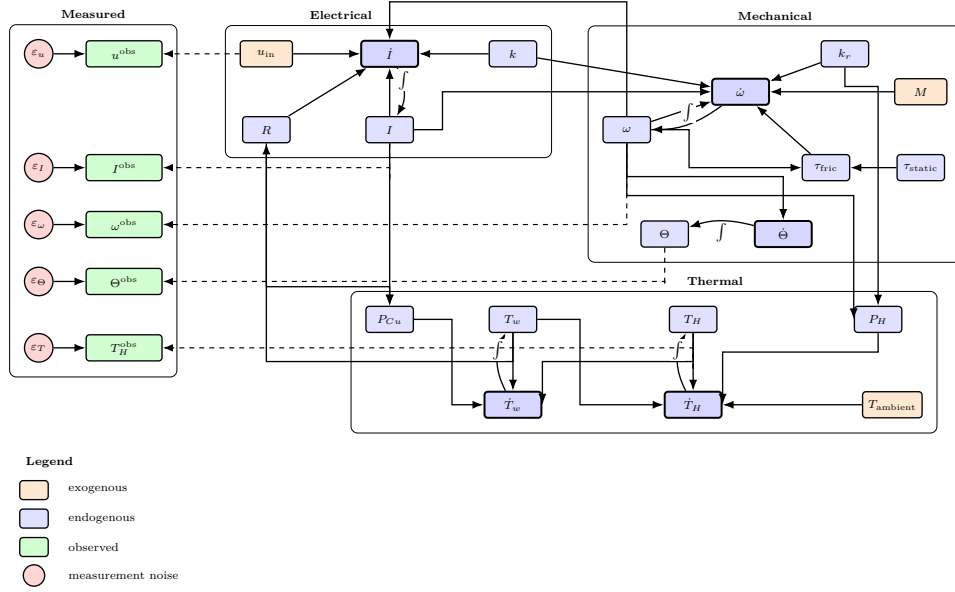
equations describing the causal mechanisms for the state endogenous variables written in Equations 2, 3, 6, 7 and the expression $\omega(t) = \dot{\theta}(t)$. To complete the causal model, we define the exogenous variables in a vector $u(t) = [u_{el}(t) \quad M(t) \quad T_{ambient}(t)]^T$. $u_{el}(t)$ is the effective input voltage, $M(t)$ is the outside torque required for the task and $T_{ambient}(t)$ is the ambient temperature. A graph of the dynamic SCM can be seen in Figure 1. The figure shows the model's causal relations, but leaves out some mechanisms since the graph would be too crowded. A graph like the one in Figure 1 is meant to easily show the causal relationships and help with determining how interventions (and with that counterfactual) would propagate through the model.

Each fault is represented as an intervention on one structural mechanism of the motor model. Thus, a fault does not directly assign a value to an observed variable. Instead, it modifies the equation that generates the affected variable. The observed symptoms then arise through the propagation of this mechanism change through the dynamical system. The electrical faults in the model are defined as follows. A resistance fault increases (FEF1) the effective armature resistance $R_0 \rightarrow R_0 + \Delta R$. This affects both the electrical voltage equation and the copper losses. A brush or commutator voltage drop fault (FEF2) introduces an additional voltage loss to the input $u_{in}(t) = u_{el}(t) - u_{drop}(t)$. A localized commutator fault (FEF3) introduces a voltage drop only at a specific rotor angle $u_{drop}(t) = \Delta V_s \cdot g(\theta(t))$.

Here, $g(\theta)$ is an angular gate function:

$$g(\theta) = \begin{cases} 1, & |\text{wrap}_\pi(\theta - \theta_0)| \leq \delta, \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

This means that the fault is only active when the rotor passes through a defective angular sector. A demagnetization fault (FEF4) is modelled by reducing the motor constant $k \rightarrow k - \Delta k$. Since k appears in both the electrical and mechanical equations, this fault simultaneously changes the back-EMF and the produced motor torque.



■ **Figure 1** A dynamic SCM representing a DC electric motor model. For clarity, this graph does not contain all variables of the full model.

The mechanical faults are modelled similarly. Increased viscous friction fault (FMF1) modifies the viscous friction coefficient $k_r \rightarrow k_r + \Delta k_r$. An increased Coulomb friction (FMF2) fault modifies the Coulomb friction level: $\tau_C \rightarrow \tau_C + \Delta \tau_C$. An inertia fault (FMF3) changes the effective inertia of the system $J \rightarrow J + \Delta J$. Finally, a periodic mechanical disturbance (FMF4) is represented as an angle-dependent torque $M_p(t) = \Delta M_p \cdot g(\theta(t))$.

Finally, we write the SCM model in a compact form. At every time step, the previous state x_t , the input voltage u_t , the load torque $M_{L,t}$ and the active fault vector F_t determine the next state $x_{t+\Delta t}$:

$$x_{t+\Delta t} = f_{SCM}(x_t, u_t, M_{L,t}, F_t). \quad (10)$$

The measured values are then generated from the true simulated states by adding measurement noise: $y_t = h(x_t) + \varepsilon_t$. The variable y_t denotes the measured variables, $h(\cdot)$ is the measurement function, and ε_t is measurement noise. The measurement function $h(\cdot)$ represents a sampling algorithm from simulation sampling time to sensor sampling. To mimic reality, we keep the states of the motor hidden and only observe certain variables that would be easily measurable with sensors, i.e., the voltage, current, angle, rotational velocity, and the housing temperature.

The motor is simulated by numerically integrating the structural equations. For this, we use a fourth-order Runge–Kutta method. During one integration step, the input voltage, load torque, and active fault configuration are kept constant. Faults can therefore be scheduled by changing the corresponding fault variable at a chosen time.

3.3 Counterfactual fault responsibility analysis

An SCM can also be used to explain which fault mechanisms are responsible for specific observed symptoms. This is particularly useful when multiple faults are active at the same time. In such cases, identifying the set of active faults does not necessarily explain which fault caused which part of the faulty behaviour. We use counterfactual fault removal to assign responsibility to individual mechanisms.

Let the factual fault configuration set be denoted by $F = \{F_1, F_2, \dots, F_k\}$,

where multiple fault mechanisms may be active, that is $F_i > 0, i = 1, \dots, k$. This is the set of all faults we predefine to be possible or probable in a system. The factual trajectory is generated by simulating the SCM under a fault configuration containing at least a single active fault F^{fact} :

$$x_{t+\Delta t}^{fact} = f_{SCM} \left(x_t^{fact}, u_t, M_{L,t}, do(F = F^{fact}) \right). \quad (11)$$

This represents a real scenario where a system is detected to be faulty by an arbitrary measure, but it is unknown what exactly the root cause is; that is, we know that there is at least a single fault active $F_j > 0$, but don't know which fault exactly is to blame such that $F_j \in F$.

To evaluate the responsibility of any particular fault $F_i \in F$, we construct a counterfactual world in which this fault is removed while the remaining background conditions are kept fixed. This corresponds to a counterfactual trajectory with a single fault F_i removed:

$$x_{t+\Delta t}^{-i} = f_{SCM} \left(x_t^{-i}, u_t, M_{L,t}, do(F_i = 0, F_{\setminus i} = F_i^{fact}) \right). \quad (12)$$

Here, the same initial state and torque profile are used as in the factual case; the controller input is recomputed in each counterfactual branch since the system is closed in a control loop.

Responsibility score is then measured by comparing the symptom measure between the observed and the counterfactual trajectories. Let $S_s(x)$ denote a scalar measure for any symptom s . The responsibility of fault F_i for symptom s is defined as

$$R_{i \rightarrow s} = S_s(x^{fact}) - S_s(x^{-i}). \quad (13)$$

A large positive value means that removing fault F_i reduced the symptom s and therefore that F_i is responsible for that symptom. A value close to zero means that the symptom is largely unchanged when the fault is removed.

This analysis allows the diagnosis to be refined from a fault-level statement to a symptom-level explanation. For example, a localized electrical defect may be responsible mainly for angle-synchronous current and tracking ripple, while increased Coulomb friction may be responsible mainly for persistent high current and thermal load. Counterfactual removal therefore explains how different fault mechanisms contribute to different aspects of the observed faulty behaviour.

4 Experiments

We simulate the model parameters found in a datasheet for an industrial DC motor, MAXON RE40 (model number 148867) motor [9].

We evaluate the counterfactual responsibility procedure on two simulated fault-assignment experiments. The purpose of these experiments is not to identify an unknown fault, but to explain which fault mechanism best explains the observed symptom. Each experiment follows the same procedure: simulate the faulty branch, remove each candidate fault by intervention, compute symptom-specific scores, and assign contribution or necessity by comparing factual and counterfactual branches.

All experiments use the previously described SCM, simulated with an integration step of $\Delta t = 10^{-4}$ s. The faults are present from the start of the simulation, so the system never evolves under nominal conditions. The motor is controlled by the same closed-loop position

controller in every factual and counterfactual branch. The full control loop consists of a PI controller for the positional error, $e(t) = \theta_{ref}(t) - \theta(t)$, voltage saturation of 24 V, and a small feedforward term to compensate for the motor's own inertia.

The electrical and mechanical quantities are sampled at $f_{el} = 10$ kHz and $f_{mech} = 1$ kHz, respectively, with added Gaussian noise. The same mechanical sampling is used for the controller. Position is measured with a 10-bit relative encoder. Noise is added to all measurements with standard deviations of $\sigma_u = 0.05$ V, $\sigma_{i_a} = 0.02$ A, $\sigma_\omega = 0.5$ rad/s, $\sigma_\theta = 0.002$ rad and $\sigma_{T_h} = 1$ °C. The values of both the sampling frequencies and measurement noise are chosen as plausible for sensors accompanying a DC motor in use.

Unlike the real setting where the initial states of the system would come from an abduction step (filtering), we use the simulated values directly as we have access to them. To verify that the method works with a mismatch between the simulated motor and the model, we run $N = 100$ tests with random parameter values uniformly sampled around nominal values (from [9]) by 25%.

4.1 Angle-synchronous ripple

The first experiment tests the method on two simultaneously active faults: FEF3, an angle-gated localized electrical voltage drop, and FMF2, an increase in Coulomb friction. Hypothetical scenario represents a damaged brush-commutator contact for $\delta = 10^\circ$ where the voltage drops by 2 V, and worn bearings increasing the load torque by $\Delta\tau_C = 0.1$ Nm at angular velocity around 0 rad/s.

The reference angle repeatedly passes through the damaged angular sector, making the localized electrical defect observable. At the same time, the trajectory contains low-speed reversals and loaded crawling phases. The motor is simulated for 120 s.

The simulated world faults FEF3 = 1 and FMF2 = 1 active is compared to three different counterfactual worlds: $do(\text{FEF3} = 0)$, $do(\text{FMF2} = 0)$ and $do(\text{FEF3} = 0, \text{FMF2} = 0)$.

These counterfactual worlds are chosen to show whether the method can correctly assign the blame and separate the effects of faulty behaviour from individual faults and their combined effect. In a more realistic setting, all possible faults would be included as possible, or some subset chosen by a heuristic. The expected assignment is that FEF3 is responsible mainly for angle-synchronous current and tracking ripple, while FMF2 is responsible mainly for persistent high current and overheating.

To evaluate the angle-localized symptom, we define a diagnostic angular sector around the defective commutator position:

$$g_{\text{diag}}(\theta) = \mathbf{1}(|\text{wrap}_\pi(\theta - \theta_0)| \leq \delta_{\text{diag}}), \quad (14)$$

with $\theta_0 = 0$ and $\delta_{\text{diag}} = 15^\circ$.

This diagnostic sector is slightly wider than the physical fault sector to capture the dynamic response around the defective angular region. Given the information from the encoder, it's possible to estimate at which angles the faults appear and therefore estimate the diagnostic sector δ_{diag} .

For each counterfactual trajectory, we compute a set of scores, each pointing toward a symptom. To calculate the scores, we only use the measurements we plausibly could have in the system already. We record tracking RMSE, RMS and peak current, RMS and peak voltage, voltage-saturation time (above $0.98u_{max}$) and peak temperatures. The thermal symptom is evaluated using the winding and housing temperature. A winding-overheating event is declared if $\max_t T_w(t) \geq T_{w,\text{lim}}$. A housing-temperature event is defined analogously

using $T_{h,\text{lim}} = 70^\circ\text{C}$. To measure not only whether the threshold is crossed but also how severe the violation is, we use the thermal excess cost.

$$J_{\text{thermal}} = \frac{1}{N} \sum_k (\max(0, T_w(k) - T_{w,\text{lim}}))^2 + 0.25 \frac{1}{N} \sum_k (\max(0, T_h(k) - T_{h,\text{lim}}))^2, \quad (15)$$

which grows quadratically if they exceed the limits and is zero if the respective temperatures stay in the safe region below. We also say that overheating of the winding is four times as harsh, hence the 0.25 factor in the expression.

To evaluate the localized electrical fault symptom, we compute the differences in the angle tracking values and current in the damaged commutator sector (denoted by $\in$) and outside of it ($\notin$). A diagnostic angular gate at the damaged commutator is defined in Equation 14. To describe the angle-synchronous tracking ripple we write $R_\theta = \text{RMSE}_{e_\theta, \in} - \text{RMSE}_{e_\theta, \notin}$ and for the angle-synchronous current ripple, $R_I = \text{RMS}_{I, \in} - \text{RMS}_{I, \notin}$.

An angle-ripple event is detected if either of these quantities exceeds its threshold of 0.05 rad and 0.75 A respectively. The corresponding angle-ripple cost is

$$J_{\text{angle}} = \left(\frac{\max(0, R_\theta)}{R_{\theta, \text{lim}}} \right)^2 + \left(\frac{\max(0, R_I)}{R_{I, \text{lim}}} \right)^2. \quad (16)$$

Finally, the overall severity score is a weighted sum of mean tracking error, mean current, voltage saturation, thermal excess, and angle-synchronous ripple:

$$J_{\text{total}} = w_{\text{track}} J_{\text{track}} + w_I J_I + w_{\text{sat}} J_{\text{sat}} + w_{\text{thermal}} J_{\text{thermal}} + J_{\text{angle}}. \quad (17)$$

In our experiments, all weights were set to 1. The individual score components are chosen to match the expected physical effects of the faults we model and are aware of the effects. The localized electrical fault FEF3 is angle-dependent, so its main symptom should appear as an increase in current or tracking error when the rotor passes through the damaged angular sector. Increased Coulomb friction FMF2 primarily increases the required motor current and therefore the winding and housing temperatures. Voltage saturation is included because severe faults may force the controller to operate at its actuator limit.

Counterfactual responsibility is evaluated by comparing these scores to the SCM factual branch. For a removed fault F_i , the reduction in a symptom score is $\Delta J_s^{-i} = J_s^{\text{fact}} - J_s^{-i}$.

A large positive value means that removing F_i reduces symptom s , and therefore that the fault contributes to that symptom. If an event occurs in the factual branch but disappears under $do(F_i = 0)$, then F_i is treated as necessary for that event. This separates event prevention from severity reduction: a fault may be responsible for making a symptom worse even if it is not individually necessary for the event threshold to be crossed.

4.2 Overheating responsibility

The second experiment evaluates responsibility for winding overheating in the presence of partial demagnetization (FEF4) and increased viscous friction (FMF1). This combination represents an aged motor with weakened permanent magnets and worn bearings. The factual configuration is $\text{FEF4} = 1, \text{FMF1} = 1$, with $\Delta k = 0.05k_0$ and $\Delta k_r = 30k_{r0}$. The motor is simulated for 1200s to capture its slower thermal dynamics. The reference and load profiles are chosen to excite viscous friction through sustained moderate-to-high-speed operation. The motor follows a constant-speed trajectory while the load torque increases in steps from 120 mNm to the nominal value of 177 mNm.

The factual world is compared to five counterfactual worlds: $\text{do}(\text{FEF2} = 0)$, $\text{do}(\text{FEF4} = 0)$, $\text{do}(\text{FMF2} = 0)$, $\text{do}(\text{FMF1} = 0)$ and $\text{do}(\text{FEF4} = 0, \text{FMF1} = 0)$. FEF2 and FMF2 are inactive in the factual world and are therefore included as negative controls. FEF4 and FMF1 are the candidate mechanisms for explaining the observed overheating.

We use the tracking, current, voltage-saturation, and thermal scores defined in Section 4.1, omitting the angle-ripple term. The primary event is winding overheating:

$$E_{\text{thermal}} = \mathbf{1} \left[\max_k T_w(k) \geq T_{w,\text{lim}} \right], \quad (18)$$

where $T_{w,\text{lim}} = 90^\circ\text{C}$. Thermal severity is quantified using J_{thermal} from Equation 15, and the total score is the equally weighted sum of the remaining score components.

A fault contributes to thermal severity if its removal decreases the peak winding temperature by more than 5°C , a heuristically selected threshold. The reduction is measured as

$$\Delta T_{w,\text{max}}^{-i} = \max_k T_w^{\text{fact}}(k) - \max_k T_w^{-i}(k).$$

A stronger condition is necessity: fault F_i is necessary for overheating when the event occurs in the factual trajectory but disappears after its removal,

$$E_{\text{thermal}}^{\text{fact}} = 1 \quad \text{and} \quad E_{\text{thermal}}^{-i} = 0. \quad (19)$$

5 Results

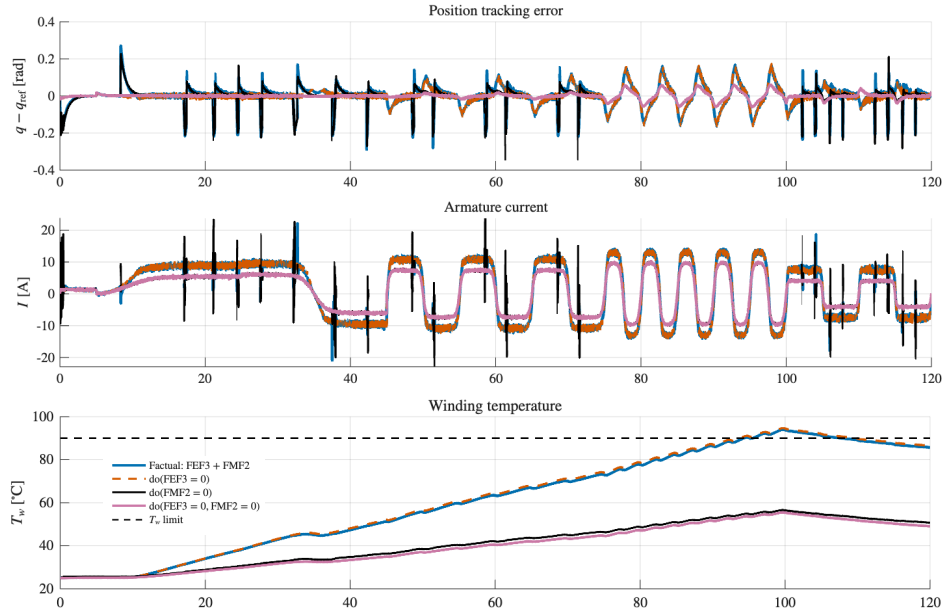
5.1 Fault blame assignment - Angle-synchronous ripple

Figure 2 shows the position tracking error, armature current, and winding temperature for the factual and counterfactual trajectories. The upper panel reveals angle-localized disturbances in branches where FEF3 remains active. These disturbances are strongly suppressed when FEF3 is removed, supporting the interpretation that FEF3 is responsible for the angle-synchronous tracking symptom.

The middle panel shows that branches with FMF2 active require a persistently higher armature current, while FEF3 additionally produces localized current spikes. Removing FMF2 lowers the current magnitude throughout the experiment, whereas removing FEF3 primarily suppresses the angle-synchronous spikes.

The winding-temperature trajectories in the lower panel show that FMF2 is necessary for the overheating event. Every branch in which FMF2 is removed remains below the winding-temperature limit, while removing FEF3 alone does not prevent overheating.

In Table 1, we show numerical results of our experiments, the values of score functions from Section 4.1, and winding temperature values for each simulated world. From the table, we can conclude that intervention $\text{do}(\text{FEF3} = 0)$ always removes the angle ripple event, while the intervention $\text{do}(\text{FMF2} = 0)$ always removes the overheating event. The results indicate that the effects of the two faults are interaction-dependent. Removing one fault does not necessarily reduce every cost component, because the intervention changes the system trajectory and, consequently, the conditions under which the remaining fault propagates. For example, removing only FMF2 eliminates the thermal event but increases (J_{ang}). This increase is associated with a change in the angle-dependent current behaviour: the RMS current inside the affected angular sector becomes higher than outside it. This suggests that, in the absence of FMF2, the effect of FEF3 on the current becomes more pronounced within



■ **Figure 2** Counterfactual results for the FEF3+FMF2 scenario: position tracking error, armature current, and winding temperature. Removing FEF3 suppresses angle-synchronous disturbances, while removing FMF2 reduces current demand and prevents overheating.

the affected angular sector. We therefore do not interpret the total score alone as a global responsibility measure. Instead, responsibility is assigned symptom-wise: FMF2 is necessary for the thermal event, while FEF3 is necessary for the angle-ripple event.

■ **Table 1** Counterfactual fault-assignment results for the combined FEF3 and FMF2 experiment. The factual branch contains both the angle-gated electrical fault FEF3 and the increased Coulomb-friction fault FMF2. A dash indicates that the winding-temperature limit was not reached.

Intervention	Removed fault	max T_w [°C]	RMSE $_q$ [rad]	RMSE $_{q,sec}$ [rad]	J_T	J_{ang}	J_{tot}	T_w event	Angle event
Factual: FEF3 + FMF2	none	94.00	0.0606	0.1145	0.5090	1.585	83.631	yes	yes
do(FEF3 = 0)	FEF3	94.70	0.0430	0.0121	0.8496	0.000	83.472	yes	no
do(FMF2 = 0)	FMF2	56.63	0.0444	0.1186	0.0000	4.537	43.124	no	yes
do(FEF3 = 0, FMF2 = 0)	FEF3+FMF2	55.48	0.0163	0.0049	0.0000	0.000	36.038	no	no

In the randomized-parameter repetitions, the method assigned FEF3 as necessary for the angle-ripple event and FMF2 as necessary for the thermal event in all 100 runs.

5.2 Fault blame assignment - Overheating responsibility

Figure 3 shows the winding temperature and armature current for the factual branch and the counterfactual branch in which FMF1 is removed. With FMF1 active, the motor requires a persistently higher armature current throughout the loaded operating phases. After the intervention do(FMF1 = 0), the current demand is substantially reduced, indicating that increased viscous friction primarily appears as an increase in the actuation effort required to follow the same task.

The upper panel shows the thermal consequence of this increased current demand. In the factual branch, the winding temperature crosses the 90°C limit and reaches approximately 106°C. When FMF1 is removed, the winding temperature remains below the limit throughout the experiment. FMF1 is therefore necessary for the winding-overheating event under the considered operating conditions.

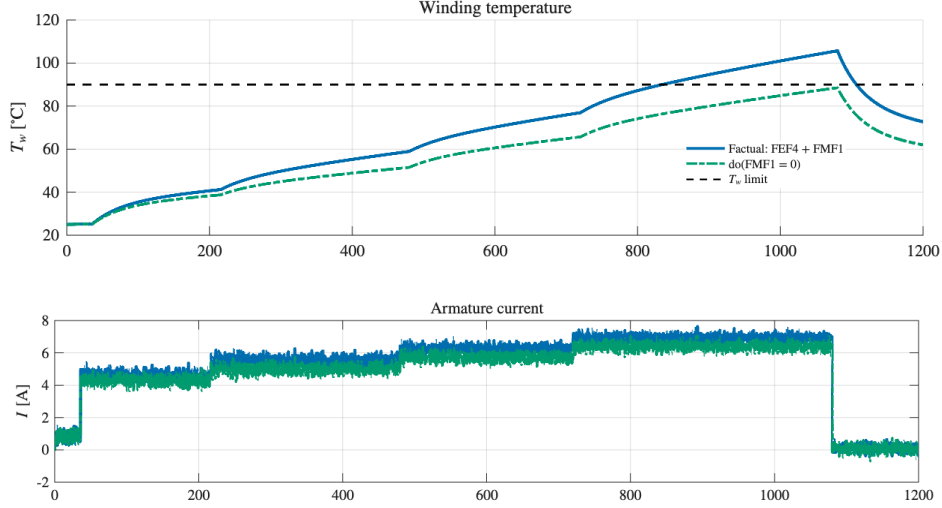


Figure 3 Counterfactual winding temperature and armature current for the FEF4+FMF1 scenario. Removing FMF1 reduces current demand and prevents overheating.

Table 2 reports the numerical results for all counterfactual branches. The counterfactual results indicate that the observed overheating is mainly attributable to FMF1, while FEF4 has a smaller but non-negligible contribution. Removing FEF4 reduces the peak winding temperature and decreases (J_T), but it does not prevent the winding-temperature event. In contrast, removing FMF1 reduces the peak winding temperature, eliminates the thermal cost, and prevents the overheating event. Removing both faults produces a further reduction in temperature and total cost, suggesting that FEF4 additionally increases the thermal severity under the conditions created by FMF1. The interventions on FEF2 and FMF2 serve as negative controls; their results are identical to the factual trajectory, as expected because the faults are not active in the simulation. This supports the specificity of the counterfactual fault-assignment procedure.

To evaluate robustness to parameter mismatch, we repeated the experiment using the randomized parameter offsets described in Section 4. The method correctly assigned responsibility to FMF1 in 84% of the 100 randomized repetitions. The failed repetitions occurred when parameter mismatch changed the model’s thermal behaviour, and it became less clearly attributable to FMF1 under the chosen threshold.

6 Discussion and Limitations

The experiments demonstrate that counterfactual fault removal can separate the responsibilities of simultaneously active mechanisms. In the first experiment, removing FEF3 eliminates angle-synchronous ripple, whereas removing FMF2 prevents overheating. In the second,

Table 2 Counterfactual fault-assignment results for the combined FEF4 and FMF1 experiment. The factual branch contains both partial demagnetization FEF4 and increased viscous friction FMF1. A dash indicates that the winding-temperature limit was not reached.

Intervention	Removed fault	$\max T_w$ [°C]	$\max T_h$ [°C]	RMSE_q [rad]	J_T	J_{tot}	T_w event	Prevents T_w event
Factual: FEF4 + FMF1	none	105.70	72.53	0.0057	19.4562	51.7240	yes	–
do(FEF2 = 0)	FEF2	105.70	72.53	0.0057	19.4562	51.7240	yes	no
do(FEF4 = 0)	FEF4	96.52	67.36	0.0057	1.6265	30.7509	yes	no
do(FMF2 = 0)	FMF2	105.70	72.53	0.0057	19.4562	51.7240	yes	no
do(FMF1 = 0)	FMF1	88.51	61.72	0.0056	0.0000	27.1003	no	yes
do(FEF4 = 0, FMF1 = 0)	FEF4+FMF1	81.30	57.63	0.0055	0.0000	24.4604	no	yes

removing FEF4 reduces thermal severity without preventing the event, while removing FMF1 prevents winding overheating. The method therefore distinguishes symptom responsibility, severity contribution, and event necessity even when the controller masks faults at the position level.

Counterfactual comparisons are performed under shared background conditions. The initial state, input trajectory, load torque, controller, and reference task remain fixed, so differences between trajectories can be attributed to the removed mechanism rather than to unrelated changes in operating conditions. This is particularly useful in closed-loop systems, where the controller may hide faults in some observed variables.

The randomized-parameter repetitions suggest that the method is not tied to one exact numerical realization. The FEF3/FMF2 assignment remained stable in all runs, whereas the FMF1 assignment succeeded in (84%) of the repetitions. This shows that counterfactual explanations depend on the accuracy of the model, as well as the intervention set and the chosen event thresholds.

The main limitation is that the experiments are simulation-based and use the same model class to generate and evaluate the counterfactual branches. Real motors contain unmodelled effects such as complex commutator behaviour, contact transients, magnetic saturation, temperature-dependent friction, changing loads, sensor delays, and actuator nonlinearities. The method also assumes that the relevant fault types and their effects on system mechanisms are known. In addition, the current experiments initialize counterfactual worlds using the true simulated state. In a real diagnostic setting, this state would need to be inferred from noisy observations, and estimation errors would affect the counterfactual trajectories and responsibility scores. Future work therefore includes a complete abductive pipeline in which hidden states and uncertain parameters are estimated from measured trajectories before evaluating counterfactual interventions.

7 Conclusion

We presented a dynamic structural causal model of a DC motor for counterfactual fault responsibility analysis. Faults are represented as interventions on physical mechanisms, allowing faulty behaviour to be explained through propagation in the electro-thermo-mechanical dynamics.

The experiments show that counterfactual fault removal can distinguish symptom responsibility, severity contribution, and necessity for a failure event. In the first experiment, FEF3 explains the angle-synchronous ripple while FMF2 explains overheating. In the second, FMF1 is necessary for winding overheating, while FEF4 mainly contributes to severity. Future work will focus on real-data validation, state and parameter estimation, and robustness under unmodelled fault mechanisms.

References

- 1 Luca Console and Pietro Torasso. Integrating models of correct behavior into abductive diagnosis. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 160–166, Stockholm, August 1990. Pitman Publishing.
- 2 M-O Cordier, Philippe Dague, François Lévy, Jacky Montmain, Marcel Staroswiecki, and Louise Travé-Massuyès. Conflicts versus analytical redundancy relations: a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2163–2177, 2004. doi:10.1109/TSMCB.2004.835010.
- 3 Adnan Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222, 1998. doi:10.1613/JAIR.462.
- 4 Johan de Kleer, Alan K. Mackworth, and Raymond Reiter. Characterizing diagnosis and systems. *Artificial Intelligence*, 56, 1992. doi:10.1016/0004-3702(92)90027-U.
- 5 Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987. doi:10.1016/0004-3702(87)90063-4.
- 6 Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Hypothesis classification, abductive diagnosis and therapy. In *International Workshop on Expert Systems in Engineering*, pages 69–78. Springer, 1990. doi:10.1007/3-540-53104-1_32.
- 7 Joseph Y Halpern. Defaults and normality in causal structures. In *KR*, pages 198–208, 2008. URL: <https://aaai.org/papers/KR08-020-defaults-and-normality-in-causal-structures/>.
- 8 Joseph Y Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part i: Causes. *The British journal for the philosophy of science*, 2005.
- 9 MAXON. Re40, part number 148867. <https://www.maxongroup.de/maxon/view/product/motor/dcmotor/re/re40/148867>. Accessed: 05-08-2026.
- 10 Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge university press, 2009.
- 11 Jonas Peters, Stefan Bauer, and Niklas Pfister. *Causal Models for Dynamical Systems*, pages 671–690. Association for Computing Machinery, New York, NY, USA, 1 edition, 2022. doi:10.1145/3501714.3501752.
- 12 David Poole. Explanation and prediction: an architecture for default and abductive reasoning. *Computational Intelligence*, 5(2):97–110, 1989.
- 13 Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987. doi:10.1016/0004-3702(87)90062-2.
- 14 M. Staroswiecki and G. Comtet-Varga. Analytical redundancy relations for fault detection and isolation in algebraic dynamic systems. *Automatica*, 37(5):687–699, 2001. doi:10.1016/S0005-1098(01)00005-X.
- 15 Christofer Sundström, Erik Frisk, and Lars Nielsen. Selecting and utilizing sequential residual generators in fdi applied to hybrid vehicles. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(2):172–185, 2013. doi:10.1109/TSMC.2013.2248147.
- 16 Juliane Weillbach, Sebastian Gerwinn, Karim Barsim, and Martin Fränzle. Counterfactual-based root cause analysis for dynamical systems. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 303–319. Springer, 2024.

When Dynamic Slicing Helps and Hurts Spectrum-Based Fault Localization: Evidence from an Expanded TCAS Benchmark

Iulia Nica ✉ 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Franz Wotawa ✉ 

Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Austria

Abstract

Spectrum-based fault localization and dynamic program slicing are two established approaches to automated software debugging. A recent study provided a first experimental evaluation of a hybrid approach that combined both techniques, reporting that the combination improves the wasted effort metric for programs with multiple output variables. However, as stated in the study, further experiments are required for a final judgment of the introduced debugging methodology. In this paper, we review and extend that evaluation through three rounds of experiments, all carried out under a strict single-fault assumption. First, we replicate the third experiment from the original study to validate the obtained findings. Second, we expand the TCAS benchmark from 2 to 11 faulty variants by systematically generating nine new mutants spanning six distinct mutation operator families, and verify their quality through a dedicated kill-ability assessment. Eleven of the twelve mutants pass our quality filter and are used in two independent rounds of extended experiments, each evaluated with three spectrum-based fault localization coefficients (Ochiai, Tarantula, Sarhan-Beszédes) in both the plain and hybrid configurations. Our results confirm that the slicer's behavior has a decisive impact on the quality of the hybrid approach and that improvements from slicing are far from uniform across mutation types. In particular, bugs that suppress the execution of entire program paths present a fundamental challenge for dynamic slicing, providing concrete evidence for the limitations reported in the original study.

2012 ACM Subject Classification Software and its engineering → Software testing and debugging; Computing methodologies → Causal reasoning and diagnostics

Keywords and phrases software fault localization, dynamic slicing, spectrum-based fault localization, mutation testing, automated debugging, TCAS

Digital Object Identifier 10.4230/OASICS.DX.2026.17

Category Short Paper

Funding *Franz Wotawa*: The work was supported by the Austrian Science Fund (FWF) Cluster of Excellence Bilateral AI under contract number 10.55776/COE12.

1 Introduction

Automated software debugging remains one of the most practically relevant open problems in software engineering. Among the approaches proposed to support fault localization, spectrum-based fault localization (SFL) [6, 2] and dynamic program slicing [8] have each attracted substantial research attention. SFL assigns suspiciousness scores to program statements based on their co-occurrence with passing and failing test outcomes, while dynamic slicing identifies statements that causally contribute to a given output value for a specific test execution.

Schleich and Wotawa [11] recently provided the first experimental evaluation of a hybrid approach that integrates dynamic slicing into SFL, implemented within the open-source JSR framework [12]. Their study considered three sets of Java programs, including the well-known



© Iulia Nica and Franz Wotawa;

licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 17; pp. 17:1–17:15

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

TCAS collision avoidance benchmark. A key finding was that for programs with more than one observable output variable, the hybrid approach consistently improves the wasted effort metric compared to plain SFL, while the impact of the underlying slicer tool was identified as a significant source of variability.

However, the evaluation of the hybrid approach on TCAS programs was limited to only two faulty variants. Given that the TCAS program has a well-documented history as a mutation testing benchmark [4], a broader evaluation covering more diverse mutation operator families would be valuable for assessing the generality of the findings.

This paper makes the following contributions:

- We *replicate* the third experiment of [11], confirming that the published results are reproducible under our independent infrastructure (Section 3).
- We *construct an expanded TCAS mutation benchmark* by adding *nine new faulty variants* to the original two, spanning six distinct mutation operator families drawn from the spreadsheet mutation testing literature [1]. Each variant is exercised by a parameterized JUnit test suite of *40 test cases*, structured as eight input combinations $\times$ five output variable assertions. The full benchmark comprises 480 instrumented test executions and 2,400 individual output assertions per round of experiments (Section 4).
- We perform a *quality assessment* of the generated mutants through a dedicated kill-ability study and select eleven variants for the SFL evaluation (Section 4.4).
- We *conduct two independent rounds of experiments* on the expanded benchmark (EE1 and EE2), each evaluating three SFL coefficients (Ochiai, Tarantula, Sarhan-Beszédes) in both plain-SFL and hybrid (sliced) configurations (Section 5).
- We *identify a structural explanation* for the cases in which dynamic slicing degrades SFL accuracy: mutations that suppress execution of entire conditional blocks systematically defeat the slicer (Section 6).

Throughout all experiments we adopt the *single-fault assumption*, i.e., each faulty program variant contains exactly one intentionally introduced defect.

The remainder of this paper is structured as follows. Section 2 briefly recalls the foundations of SFL and dynamic slicing. Section 3 presents the replication of the third experiment from the original study. Section 4 describes the construction of the expanded TCAS benchmark, including mutant generation, test suite design, and kill-ability assessment. Section 5 reports the two rounds of extended experiments. Section 6 discusses the findings and their implications, Section 7 addresses threats to validity, and Section 8 concludes.

2 Background

We briefly review the foundational concepts relevant to this paper, for a more detailed discussion, we refer the reader to [11].

2.1 Spectrum-based Fault Localization

SFL uses the *program spectrum*, a matrix recording for each test case and each program statement whether the statement was executed, together with an error vector recording whether each test passed or failed. From these data, four aggregate counts $a_{nm}(i)$ are computed for each statement i , where $n \in \{0, 1\}$ indicates whether the statement was executed and $m \in \{0, 1\}$ indicates whether the test passed (0) or failed (1). These counts are combined into a suspiciousness coefficient. We evaluate three coefficients: Tarantula [6], Ochiai [2], and Sarhan-Beszédes [9].

Debugging quality is measured by the *rank* of the faulty statement and the *wasted effort* (WE), defined as

$$WE = |\{s : \text{score}(s) > \text{score}(s_f)\}| + 0.5 \cdot (|\{s : \text{score}(s) = \text{score}(s_f)\}| - 1), \quad (1)$$

where s_f is the faulty statement. Lower WE is better. We also report Hit Ratio@ k (Hit@ k), the fraction of programs for which the fault is ranked within the top k positions.

2.2 Dynamic Slicing and the Hybrid Approach

Dynamic slicing [8] computes, for a given execution and a chosen output variable, the set of statements that causally influenced that variable's value. The hybrid approach of [11] decomposes each test case into one sub-test per output variable and uses the dynamic slice for each output to restrict which statements are considered in the SFL computation for that sub-test. This allows the approach to exploit data-dependency information that plain SFL ignores.

The extended version of the JSR framework [10] implements this hybrid approach for Java programs, employing JaCoCo [5] for line coverage analysis and Slicer4J [3] for dynamic slicing. This implementation builds upon the original JSR framework developed by Stracke [12, 7].

3 Round 1: Replication of the Third Experiment

The third experiment of [11] used the same Java programs as the initial experiment, but increased the number of observable output variables per program. For the TCAS program, five output variables were considered: `alim`, `inhibit_biased_climb`, `non_crossing_biased_climb`, `non_crossing_biased_descend`, and `return_value`. Each input combination thus yields five assertions, and the JUnit test suite consists of 40 parameterized test cases per TCAS variant (5 assertions $\times$ 8 input combinations).

3.1 Replication Procedure

We executed the third experiment using the original `smallProject` provided with the JSR repository, retaining the two original TCAS variants (`Tcas`, `Tcas2`) alongside the remaining programs of the benchmark. The JSR pipeline was driven through the IntelliJ plugin (with *Coverage Metric: checked*, *Reduction Algorithm: Greedy HGS*, *Keep 0 coverage*, *Deactivate redundant test cases*) and the subsequent SFL matrix computation was triggered through the JSR command-line interface. The post-processing was performed with the original `SFL.py` script.

3.2 Replication Outcome

For the TCAS rows of *Tables 13* and *14* of [11], our independently obtained ranks and wasted-effort values coincide with those reported in the original study, confirming the reproducibility of the published experimental setup. This replication serves two purposes: it establishes the baseline against which the extended experiments in Section 5 can be compared, and it validates that the JSR tool chain behaves consistently across installations.

It is worth noting that the replication required two minor adjustments to the build environment. First, the dependency `soot-inflow-2.10.0-mandoline.jar` was not available in the standard Maven repository and had to be installed manually, with the corresponding version entry in `pom.xml` updated accordingly. Second, a compile error in `ThreadCalls.java`

had to be resolved: the method `MethodSubSignature(subsig)` is no longer present in newer versions of Soot, requiring a small adaptation of the `ThreadCall.java` source file. Neither of these adjustments affected the experimental results; they reflect the natural evolution of third-party dependencies over time and are documented here to assist future replicators of this work.

4 An Expanded TCAS Mutation Benchmark

The original study relied mostly on programs with fewer than five output variables, the single example with five outputs being the TCAS, as defined in Section 3.3 of [11]. To address this, we constructed an expanded benchmark of eleven TCAS variants by generating *nine new mutants* and adding them to the two preexisting ones. This section documents the construction process, the test suite that exercises each variant, and the quality assessment used to filter weak mutants before the SFL experiments.

4.1 Existing TCAS Variants

The JSR repository ships with two families of TCAS variants: a set of single-output programs and a set of five-output programs. We work exclusively with the five-output family (`Tcas`, `Tcas2`, `Tcas3`), which is the one relevant to the third experiment of [11]. A close inspection of the `classes` folder used by the `testall.py` script reveals two inconsistencies that must be acknowledged before any extension can be made:

1. **Inconsistent fault-marker placement.** The `//Error` comment marking the faulty line is placed in `Tcas.java` (the buggy original), but in the `Tcas2Altered` and `Tcas3Altered` files for the second and third variants. The non-annotated files in those cases are the correct versions. All three correct (non-annotated) files are textually identical; only the altered versions differ.
2. **Compound mutation in `TcasAltered`.** The first altered variant introduces two simultaneous changes to line 59: `+NOZCROSS` becomes `-NOZCROSS` (an AOR-style operator replacement) *and* a division by 1 is added. Technically, this violates the single-fault assumption. We retain only the AOR-style operator replacement in our further experiments.

For all new variants we adopt the convention that each `TcasN.java` ($N \in \{4, \dots, 12\}$) is the correct version with no `//Error` annotation, and the corresponding `TcasNAltered.java` contains exactly one fault marked with a `//Error` comment.

4.2 Mutation Operator Selection

To ensure a systematic and reproducible mutant generation process, we adopt the mutation operator taxonomy of Abraham and Erwig [1], originally introduced for end-user spreadsheet mutation testing but directly applicable to imperative Java code. Table 1 summarizes the sixteen operators of this taxonomy.

From this taxonomy we selected six operator families for our new mutants: AOR, CRP, LCR, ROR, FRC, and UOI. The selection criterion was that the operator should be meaningfully applicable to the imperative control- and data-flow constructs of the TCAS code, so that resulting faults are realistic representatives of typical programmer mistakes (sign errors, incorrect constants, swapped predicates, miss-negations, missing operands, etc.). Range-oriented operators (CRS, NRS, CRE, NRE, RRR) are not directly applicable to the TCAS code, which does not manipulate ranges; we therefore omit them from the mutant generation.

■ **Table 1** Mutation operator taxonomy of Abraham and Erwig [1] used as a reference for selecting diverse fault types.

Operator	Description
ABS	ABSolute value Insertion
AOR	Arithmetic Operator Replacement
CRP	Constants RePlacement
CRR	Constants for Reference RePlacement
LCR	Logical Connector Replacement
ROR	Relational Operator Replacement
RCR	Reference for Constant Replacement
FDL	Formula DeLetion
FRC	Formula Replacement with Constant
RFR	ReFERENCE Replacement
UOI	Unary Operator Insertion
CRS	Contiguous Range Shrinking
NRS	Non-Contiguous Range Shrinking
CRE	Contiguous Range Expansion
NRE	Non-Contiguous Range Expansion
RRR	Range Reference Replacement

4.3 Generated Mutants

Table 2 summarizes all twelve TCAS variants used in our study, including the three from the original work [11] (*TcasAltered*, *Tcas2Altered*, *Tcas3Altered*) and the nine *new* variants generated for this work (*Tcas4Altered*–*Tcas12Altered*). For each mutant the table indicates the mutation operator family, the modified source line, and the exact textual change applied.

■ **Table 2** The twelve TCAS mutants used in this study. Mutants in italics (*Tcas4*) are excluded from the SFL experiments based on the kill-ability assessment of Section 4.4.

File	Line	Operator	Mutation detail
TcasAltered.java	59	AOR	+ NOZCROSS → - NOZCROSS
Tcas2Altered.java	89	CRP	upward_preferred = 1 → = 0
Tcas3Altered.java	130	CRP	UPWARD_RA → UNRESOLVED
<i>Tcas4Altered.java</i>	132	CRP	DOWNWARD_RA → UNRESOLVED
Tcas5Altered.java	125	LCR	Own_Below_Threat() → Own_Above_Threat()
Tcas6Altered.java	126	LCR	Own_Above_Threat() → Own_Below_Threat()
Tcas7Altered.java	117	ROR	High_Confidence != 0 → == 0
Tcas8Altered.java	117	ROR	Own_Tracked_Alt_Rate <= OLEV → >=
Tcas9Altered.java	117	ROR	Cur_Vertical_Sep > MAXALTDIFF → <
Tcas10Altered.java	59	FRC	Up_Separation + NOZCROSS → Up_Separation + 50
Tcas11Altered.java	125	UOI	Non_Crossing_Biased_Climb() → !Non_Crossing_Biased_Climb()
Tcas12Altered.java	59	UOI	Up_Separation + NOZCROSS → -Up_Separation + NOZCROSS

The nine new mutants distribute across the operator families as follows: two LCR mutations (*Tcas5*, *Tcas6*) targeting logical connectors in predicates that determine the climb/descent direction, three ROR mutations (*Tcas7*, *Tcas8*, *Tcas9*) targeting relational operators in the guard expression of the main resolution-advisory block, one CRP mutation (*Tcas4*) substituting a constant value, one FRC mutation (*Tcas10*) replacing a formula sub-expression with a constant literal, and two UOI mutations (*Tcas11*, *Tcas12*) intro-

ducing unary negation operators in different program locations. Together with the three inherited mutants this yields six distinct operator families across twelve fault locations, distributed over the three core decision procedures of TCAS: `Inhibit_Biased_Climb()`, `Non_Crossing_Biased_Descend()`, and `alt_sep_test()`. This represents a four-fold increase in mutant diversity over the original study on TCAS programs.

4.4 Kill-ability Assessment

A mutation testing benchmark is only useful if its mutants are actually *kill-able*, i.e., distinguishable from the correct program by at least one test case in the suite. We assessed kill-ability using the `testall.py` script from the JSR repository, which executes each test case against both the original and the altered program and counts passing and failing test cases. Table 3 reports the results for all twelve mutants.

■ **Table 3** Kill-ability assessment of the twelve TCAS mutants. Each mutant is run against all 40 test cases of the parameterized JUnit suite. A higher number of failing test cases corresponds to a stronger (more easily killed) mutant. The kill rate is the fraction of failing test cases. Tcas4 is flagged as a weak mutant and excluded from the SFL experiments.

Variant	Operator	Passing	Failing	Kill rate
Tcas	AOR	35	5	12.5%
Tcas2	CRP	38	2	5.0%
Tcas3	CRP	38	2	5.0%
<i>Tcas4</i>	<i>CRP</i>	<i>39</i>	<i>1</i>	<i>2.5% (weak, excluded)</i>
Tcas5	LCR	36	4	10.0%
Tcas6	LCR	38	2	5.0%
Tcas7	ROR	29	11	27.5% (strong)
Tcas8	ROR	29	11	27.5% (strong)
Tcas9	ROR	23	17	42.5% (strong)
Tcas10	FRC	37	3	7.5%
Tcas11	UOI	37	3	7.5%
Tcas12	UOI	33	7	17.5%

4.4.1 Quality Observations

Three observations follow from Table 3. First, the three ROR mutants (Tcas7, Tcas8, Tcas9) yield the highest number of failing test cases (11, 11, and 17 respectively), which is consistent with the fact that they affect the guard expression governing the main decision branch of `alt_sep_test()`: any flipping of the guard inverts the decision behavior for a large fraction of input combinations. These are the *strongest* mutants in the benchmark. Second, Tcas4 has only one failing test case (2.5%), which is below our acceptance threshold for a useful SFL benchmark mutant - a single failing test provides very little signal for any spectrum-based ranking method. We therefore exclude Tcas4 from the subsequent SFL experiments. Third, the remaining eleven mutants exhibit failure rates between 5% and 42.5%, providing a healthy spread of difficulty levels for the SFL evaluation.

4.4.2 Final Benchmark

After applying the kill-ability filter, the final benchmark used in the extended experiments consists of the eleven variants: Tcas, Tcas2, Tcas3, Tcas5, Tcas6, Tcas7, Tcas8, Tcas9, Tcas10, Tcas11, Tcas12.

4.5 Test Suite Design and Scale

For each of the eleven TCAS variants we maintain a parameterized JUnit test class that exercises the five observable outputs (`alim`, `inhibit_biased_climb`, `non_crossing_biased_climb`, `non_crossing_biased_descend`, and `return_value`) over eight chosen input combinations. With five assertions per input combination this yields *40 test cases per variant*. Across the eleven-mutant benchmark, this corresponds to:

- $11 \times 40 = 440$ instrumented test executions per experimental round;
- $440 \times 3 \times 2 = 2,640$ ranking computations for three SFL coefficients in two configurations (sliced/not sliced) per experimental round and 5,280 across the two extended experiments EE1 and EE2.

5 Rounds 2 and 3: Extended Experiments

We performed two independent rounds of extended experiments on the eleven selected TCAS variants. The two rounds differ in how the new variants are integrated into the JSR test infrastructure: in EE1 we isolate them in a dedicated project to obtain a compact, TCAS-only spectrum matrix, while in EE2 we integrate them alongside the preexisting programs of the original `smallProject`, replicating the matrix structure used in [11]. Both rounds compute the rank and wasted effort of the faulty statement for the three SFL coefficients in both the plain-SFL and the hybrid (sliced) configurations.

All artifacts required to reproduce the extended experiments (EE1 and EE2) - the expanded TCAS benchmark, the analysis scripts, and the generated SFL matrices - are publicly available in a replication package ¹.

5.1 First Extended Experiment (EE1)

5.1.1 Motivation and Setup

The original `smallProject` accumulated benchmarks from all experiments conducted in the JSR project since 2021, with the consequence that the resulting SFL coverage matrices are very large. In particular, the plain SFL coverage matrix exported by JSR for the original `smallProject` has dimensions $557 \times 1,752$ (statements $\times$ test cases). Such a large matrix is unnecessarily expensive to process when the goal is to evaluate only the eleven TCAS variants, and it mixes signal from unrelated programs into the SFL summary tables.

To address this, we created a dedicated project, `smallProject2026`, containing only the eleven selected TCAS variants and their corresponding JUnit test classes (the `Test` companion of each TCAS variant). The SFL coverage matrices generated by JSR for this project have dimensions 451×716 , a reduction of approximately 35% in test-case rows and 60% in statement columns relative to the original project, while preserving full coverage of the eleven TCAS variants.

5.1.2 Build and Execution Procedure

The experiment was carried out in four steps:

1. **JSR rebuild against `smallProject2026`.** The JSR core was rebuilt against the new project so that the IntelliJ plugin would point to the new directory layout.

¹ <https://doi.org/10.5281/zenodo.21431782>

2. **Test Suite Reduction (TSR) through the IntelliJ plugin.** We launched the JSR IntelliJ plugin and started a TSR run with the following configuration: *Coverage Metric: checked, Reduction Algorithm: Greedy HGS, Keep 0 coverage, deactivate redundant test cases*. The plugin emits an **ERROR** log entry indicating that it fails to create the `coverageMatrix_checked_coverage` directory under the CLI output tree even when no SFL matrix was explicitly requested. The TSR itself completes successfully; we discuss this anomaly in Section 7.
3. **SFL run via CLI.** We invoked the SFL command of `JSR-CLI-1.0-SNAPSHOT.jar` with the source, classes, and test directories of `smallProject2026`, the Slicer4J location, the output directory `JSR-CLI/build/jsr/cliTest03`, and the package filter `at.tugraz`. This run completed without errors.
4. **Post-processing.** We modified the original `SFL.py` script to operate only on the eleven TCAS variants and to emit the LaTeX tables in the required format. The modified script (`SFL_new_v2.py`) produces the per-coefficient rank and WE tables, the summary `Hit@k` tables, and the hybrid-vs-plain comparison tables.

5.1.3 EE1 Results

Tables 4-6 present the EE1 results. Table 4 gives the aggregate `Hit@1`, `Hit@5`, and average WE across the eleven variants for both configurations. Table 5 reports per-variant ranks and WE for plain SFL. Table 6 reports per-variant ranks and WE for the hybrid sliced approach, with $\uparrow$ indicating improvement and $\downarrow$ indicating degradation relative to the corresponding plain-SFL entry.

■ **Table 4** EE1: Aggregate summary of findings for all eleven TCAS variants. WE stands for wasted effort.

	Ochiai			Tarantula			Sarhan-Beszédes		
	Hit@1	Hit@5	WE	Hit@1	Hit@5	WE	Hit@1	Hit@5	WE
SFL	0.45455	1.0	12.77273	0.27273	1.0	17.68182	0.45455	1.0	12.22727
Hybrid	0.09091	0.72727	18.0	0.09091	0.81818	20.0	0.09091	0.90909	17.63636

■ **Table 5** EE1: Per-variant rank and wasted effort for plain SFL (not sliced).

Variant	Ochiai		Tarantula		Sarhan-Beszédes	
	Rank	WE	Rank	WE	Rank	WE
Tcas	4	16.5	4	16.5	2	11.5
Tcas2	2	2.5	2	2.5	2	2.5
Tcas3	1	0.0	1	0.0	1	0.0
Tcas5	1	8.5	3	17.5	1	8.5
Tcas6	4	14.0	4	15.0	3	13.0
Tcas7	1	22.5	1	22.5	1	22.5
Tcas8	1	22.5	1	22.5	1	22.5
Tcas9	1	22.5	2	48.5	1	22.5
Tcas10	2	9.5	4	17.5	2	9.5
Tcas11	2	9.5	4	17.5	2	9.5
Tcas12	2	12.5	4	14.5	2	12.5

■ **Table 6** EE1: Per-variant rank and wasted effort for the hybrid sliced approach. ↑ marks an improvement, ↓ a degradation, both with respect to Table 5.

Variant	Ochiai		Tarantula		Sarhan-Beszédes	
	Rank	WE	Rank	WE	Rank	WE
Tcas	4	4.0↑	3↑	4.0↑	4↓	4.0↑
Tcas2	9↓	43.0↓	10↓	43.0↓	4↓	43.0↓
Tcas3	1	0.0	1	0.0	1	0.0
Tcas5	2↓	10.5↓	2↑	13.5↑	3↓	11.5↓
Tcas6	6↓	15.0↓	7↓	21.0↓	4↓	10.0↑
Tcas7	3↓	34.0↓	3↓	34.0↓	3↓	34.0↓
Tcas8	3↓	34.0↓	3↓	34.0↓	3↓	34.0↓
Tcas9	8↓	47.5↓	5↓	47.5↑	8↓	47.5↓
Tcas10	2	1.5↑	2↑	1.5↑	2	1.5↑
Tcas11	2	4.5↑	4	15.5↑	2	4.5↑
Tcas12	4↓	4.0↑	2↑	6.0↑	4↓	4.0↑

5.1.4 EE1 Observations

For the eleven-variant TCAS benchmark, plain SFL achieves a Hit@1 of 0.45455 with Ochiai and Sarhan-Beszédes, while the hybrid approach collapses to a Hit@1 of only 0.09091 across all three coefficients. The average WE increases from roughly 12.8 (Ochiai) and 12.2 (Sarhan-Beszédes) for plain SFL to 18.0 and 17.6 respectively for the sliced configuration. At the per-variant level the picture is more nuanced: slicing yields a clear WE improvement for Tcas, Tcas10, Tcas11, and Tcas12, no change for Tcas3, and a substantial degradation for Tcas2 and the three strong ROR mutants Tcas7, Tcas8, and Tcas9. We return to the structural cause of these degradations in Section 6.

5.2 Second Extended Experiment (EE2)

5.2.1 Setup and Execution

For EE2 we integrate the eleven TCAS variants *directly into the original `smallProject`*. Thus, the resulting project replicates the matrix structure used in the original study while incorporating the new variants.

The plugin-driven TSR run produces the same SFL matrix path anomaly as in EE1, with the `coverageMatrix_checked_coverage` directory failing to materialize. We then ran the SFL CLI command, which completed cleanly and exported the SFL outcome matrices to the corresponding `outcomeMatrix.csv` files. Finally, we updated `SFL_new.py` to use the single set of paths corresponding to the unified `smallProject` layout and re-ran the analysis.

5.2.2 EE2 Results

Tables 7–9 report the EE2 results following the same structure as EE1.

■ **Table 7** EE2: Aggregate summary of findings for all eleven TCAS variants. WE stands for wasted effort.

	Ochiai			Tarantula			Sarhan-Beszédes		
	Hit@1	Hit@5	WE	Hit@1	Hit@5	WE	Hit@1	Hit@5	WE
SFL	0.45455	1.0	12.77273	0.27273	1.0	17.68182	0.45455	1.0	12.22727
Hybrid	0.18182	0.72727	18.27273	0.18182	0.81818	20.18182	0.18182	0.90909	17.81818

17:10 When Dynamic Slicing Helps and Hurts

■ **Table 8** EE2: Per-variant rank and wasted effort for plain SFL (not sliced).

Variant	Ochiai		Tarantula		Sarhan-Beszédes	
	Rank	WE	Rank	WE	Rank	WE
Tcas	4	16.5	4	16.5	2	11.5
Tcas2	2	2.5	2	2.5	2	2.5
Tcas3	1	0.0	1	0.0	1	0.0
Tcas5	1	8.5	3	17.5	1	8.5
Tcas6	4	14.0	4	15.0	3	13.0
Tcas7	1	22.5	1	22.5	1	22.5
Tcas8	1	22.5	1	22.5	1	22.5
Tcas9	1	22.5	2	48.5	1	22.5
Tcas10	2	9.5	4	17.5	2	9.5
Tcas11	2	9.5	4	17.5	2	9.5
Tcas12	2	12.5	4	14.5	2	12.5

■ **Table 9** EE2: Per-variant rank and wasted effort for the hybrid sliced approach. ↑ marks an improvement, ↓ a degradation, both with respect to Table 8.

Variant	Ochiai		Tarantula		Sarhan-Beszédes	
	Rank	WE	Rank	WE	Rank	WE
Tcas	4	4.0↑	3↑	4.0↑	4↓	4.0↑
Tcas2	9↓	43.0↓	10↓	43.0↓	4↓	43.0↓
Tcas3	1	0.5↓	1	0.5↓	1	0.5↓
Tcas5	2↓	10.5↓	2↑	13.5↑	3↓	11.5↓
Tcas6	6↓	16.0↓	8↓	21.0↓	4↓	10.0↑
Tcas7	3↓	34.0↓	3↓	34.0↓	3↓	34.0↓
Tcas8	3↓	34.0↓	3↓	34.0↓	3↓	34.0↓
Tcas9	8↓	47.5↓	5↓	47.5↑	8↓	47.5↓
Tcas10	1↑	1.0↑	1↑	1.0↑	1↑	1.0↑
Tcas11	2	6.5↑	4	17.5	2	6.5↑
Tcas12	4↓	4.0↑	2↑	6.0↑	4↓	4.0↑

5.2.3 EE2 Observations

The plain-SFL columns of Table 8 coincide exactly with those of EE1 (Table 5), which is expected because both experiments use the same test suite and the same fault locations. The hybrid configuration, however, behaves slightly differently between EE1 and EE2: in EE2 the sliced Hit@1 rises from 0.09091 to 0.18182 across all three coefficients. The increase is driven by Tcas10, which moves from rank 2 in EE1 to rank 1 in EE2.

6 Discussion

6.1 Plain-SFL Results are Stable across Experiments

The plain-SFL results (Tables 5 and 8) are identical between EE1 and EE2, which is expected since both experiments use the same eleven TCAS variants and the same parameterized JUnit test suite. Tcas3 achieves perfect localization (rank 1, WE 0.0) with all three coefficients, while Tcas9 stands out with an exceptionally high Tarantula WE of 48.5, reflecting the difficulty of localizing a fault whose mutation is on the guard of the main resolution-advisory block.

6.2 Slicing Degrades Results for the Majority of Variants

Contrary to the expectation raised by Finding 3 of [11] - that multiple output variables lead to WE improvements with slicing - the hybrid approach *worsens* the average WE for the eleven-variant TCAS benchmark in both EE1 and EE2. Only Tcas10, Tcas11, and Tcas12 show consistent WE improvements, while Tcas2, Tcas7, Tcas8, and Tcas9 show severe degradations of up to +34 in WE (Tcas7, Tcas8) and +47.5 in WE (Tcas9). The expanded benchmark thus reveals that the favorable picture obtained in the original study was not representative of the broader behavior of the hybrid approach.

6.3 Bug Location Explains the Slicer Failures

The most problematic variants share a common structural characteristic: their faults affect the computation of the Boolean variable `enabled` (Tcas7, Tcas8, Tcas9) or of a local control-flow variable (Tcas2). When `enabled` evaluates to `false` as a consequence of the mutation, the entire conditional block in `alt_sep_test()` is never entered, so the dynamic slice of the output variable `return_value` never reaches the faulty line. The slicer therefore *excludes the buggy statement* from the slice, and SFL subsequently assigns it a low suspiciousness score, leading to a high rank and a large WE. This generalizes the slicer-related concern raised in [11] with concrete structural evidence: dynamic slicing is fundamentally unable to reach faults that suppress the execution of the very code paths that would expose them.

6.4 EE1 vs. EE2 Comparison

The single notable difference between EE1 and EE2 is the sliced Hit@1, which improves from 0.09091 (EE1) to 0.18182 (EE2) for all three coefficients. This improvement is essentially attributable to Tcas10, which achieves rank 1 after slicing in EE2 but only rank 2 in EE1. We think that the larger coverage matrix in EE2 (which includes all original programs, including the one-output variants) provides the slicer with a richer execution context for computing slices of the five output variables, leading to slightly different slice boundaries for this particular variant.

6.5 Computational Overhead

We did not record execution times during the experiments because the objective of this study was to evaluate fault localization effectiveness rather than runtime performance. Nevertheless, the computational overhead of the hybrid approach is substantially higher than that of plain SFL, since dynamic slices must be computed for every output variable and every test execution. While this overhead was acceptable for the TCAS benchmark used in our experiments, it may become a practical concern for larger programs with extensive test suites and numerous output variables. A systematic empirical evaluation of runtime overhead therefore represents an important direction for future work.

6.6 Summary Findings

We summarize the extended evaluation in five findings that refine the conclusions of [11].

Finding 1: The slicer has a decisive impact on the hybrid approach, consistent with Finding 1 of [11]. In particular, for ROR mutations on the `enabled` guard of `alt_sep_test()`, dynamic slicing is fundamentally unable to reach the faulty line, rendering the hybrid approach ineffective regardless of the number of output variables.

Finding 2: Furthermore, the effectiveness of the hybrid approach is constrained by the limitations of the underlying slicing framework. As also stated in [11], Slicer4J uses Jimple [13], a three-address intermediate representation, as the internal representation of the program. This transformation may remove important parts of the program, preventing the slicer from always returning a correct output. Notably, dynamic slicers do not always return a correct slice, which motivated extensions such as relevant slicing [14], where a slice must contain the faulty statement.

A detailed analysis of the generated execution traces and dynamic slices was performed in the original study [11] for the two TCAS variants. For the extended evaluation presented in this paper, a similar trace- and slice-level analysis is a natural next step. Such an analysis would help determine whether the observed behaviour is an inherent consequence of dynamic slicing or is influenced by implementation-specific aspects of Slicer4J.

Finding 3: Not all mutations with multiple outputs benefit from slicing. The benefit is limited to mutations where the faulty statement lies on a data-dependency path that is actually traversed by failing test cases. This qualifies Finding 3 of [11]: having multiple outputs is a necessary but not sufficient condition for the hybrid approach to improve over plain SFL.

Finding 4: We observe that the magnitude of degradation when slicing fails (up to +43.0 WE) tends to exceed the magnitude of improvement when it succeeds (at most ~8 WE), suggesting that identifying in advance the fault types for which slicing is beneficial is an important direction for making the hybrid approach practically robust. Can a developer know beforehand whether slicing is a good idea? Our analysis in Section 6.3 suggests that structural program characteristics may provide useful indicators. In particular, programs in which execution is dominated by a small number of highly influential control predicates may be more susceptible to the path-suppression effect observed in this study, whereas programs with richer data dependencies or more distributed control flow may benefit more consistently from slicing. Identifying predictive metrics that characterize when dynamic slicing is likely to improve or degrade fault localization represents an interesting direction for future research.

Finding 5: Based on our analysis, we hypothesize that an additional condition is that the faulty computation must remain on an executed data-dependency path in the failing execution. In such cases, dynamic slicing can exploit the output-specific data dependencies to eliminate statements that are irrelevant to the observed failure, thereby improving the discrimination of spectrum-based fault localization. Conversely, if the fault affects a control predicate whose evaluation determines whether the faulty computation is executed at all, the execution path containing the fault may be suppressed. As a result, the faulty statement is excluded from the dynamic slice, and the combined approach may perform worse than plain SFL. Thus, multiple outputs appear to be a necessary prerequisite, whereas the preservation of the faulty computation on an executed data-dependency path may be an additional condition for the hybrid approach to be beneficial. This hypothesis, however, requires validation on a broader range of benchmark programs and fault types.

7 Threats to Validity

We acknowledge the following threats to the validity of our experimental evaluation.

7.1 Tooling Anomalies

During both EE1 and EE2, we observed that the JSR IntelliJ plugin produces an `ERROR` log entry of the form `SFLMatrixCsvExporter.writeFiles() - Error while creating directory ../coverageMatrix_checked_coverage` even when the SFL export is not explicitly requested. The TSR phase nevertheless completes successfully and the subsequent CLI-based SFL run produces all expected matrices. We also observed that, regardless of the `-o` output-directory flag passed to the CLI, certain intermediate matrices are written to `cliTest01/coverageMatrix_line_coverage/outcomeMatrix.csv`. These anomalies suggest that the output-path configuration in JSR's matrix exporter is partially hard-coded and not fully override-able through the public CLI options. While neither anomaly affected our final numerical results (which are derived from the explicitly inspected matrices), they constitute a documented limitation of the current JSR release.

7.2 Benchmark Generality

Although we have increased the TCAS variant count from 2 to 11 and covered six mutation operator families, all variants share the same underlying program. Findings should therefore be interpreted as specific to the TCAS benchmark and not generalized to arbitrary Java programs without further empirical evidence.

It is also worth noting that TCAS's structural characteristics may influence the observed behaviour of the combined dynamic slicing and spectrum-based fault localization approach. In particular, TCAS relies heavily on interconnected control predicates, such that mutations affecting them can suppress entire execution paths and consequently reduce the discriminatory power of dynamic slicing, as discussed in 6. Whether this is specific to TCAS or also occurs in other software systems remains an open question. Programs with richer data dependencies, where incorrect outputs propagate through arithmetic computations rather than being dominated by a single predicate, may be less susceptible to this suppression effect, since a mutation is then more likely to alter a value along a path that still executes rather than eliminate the path entirely. Similarly, programs whose behaviour is distributed across several independent decisions, instead of a small number of dominant predicates, may be less prone to the same all-or-nothing failure. Evaluating the proposed approach on such structurally diverse benchmark suites remains a key objective for future research.

7.3 Slicer Dependence

The hybrid approach depends on the choice of dynamic slicer (Slicer4J [3] in our case). A slicer with different internal heuristics could in principle reach different conclusions about which statements are excluded from a slice when the faulty path is suppressed. Re-running our experiments with alternative slicers is left to future work.

7.4 Single-fault Assumption

Our evaluation follows the single-fault assumption commonly adopted in spectrum-based fault localization research and in the original study we replicate in the first place. This enables controlled experimentation and direct comparison with prior work. However, real-world software systems may contain multiple faults, which could interact in ways that affect both spectrum-based suspiciousness rankings and the computed dynamic slices. Evaluating the proposed approach in multi-fault scenarios is therefore an important direction for future work.

7.5 Raw Wasted Effort

Since all evaluated subjects are variants of the same TCAS benchmark, the number of executable statements is constant across experiments. Therefore, raw Wasted Effort (WE) values are directly comparable in our setting. For studies involving programs of substantially different sizes, a normalized WE metric would facilitate cross-benchmark comparisons.

8 Conclusion

In this work, we present a replication and expansion of the experimental evaluation of the hybrid dynamic-slicing SFL approach of Schleich and Wotawa [11]. The contribution comprises three rounds of experiments. In the first round, we replicated the third experiment of the original study and confirmed its reproducibility. In the second and third rounds (EE1 and EE2), we extended the TCAS benchmark from two to eleven faulty variants by generating nine new mutants across six mutation operator families, and evaluated the three SFL coefficients in both the plain-SFL and hybrid sliced configurations.

The expanded benchmark of 11 retained variants $\times$ 40 parameterized JUnit test cases $\times$ 5 output assertions per test corresponds to 2,200 individual output-value checks per SFL coefficient per configuration, or 5,280 ranking computations across the two extended experiments and three coefficients in both configurations. This represents a substantial extension of the empirical evidence base for the hybrid approach.

Our central finding is that improvements from dynamic slicing are highly dependent on the structural relationship between the fault location and the dynamic slices computed for the output variables. The Slicer4J tool fails to include the faulty statement in its output whenever the fault prevents the execution of the very code path that would reveal it - a fundamental limitation of dynamic slicing that manifests precisely for the strongest mutations (those with the highest number of failing test cases). This provides new, concrete structural evidence for the slicer-related limitations identified in the original study and suggests that fault localization approaches combining slicing and SFL should prioritize addressing slicer accuracy on control-flow-dominated faults.

Future work should investigate whether static slicing or hybrid static/dynamic slicing can overcome this limitation, whether the findings generalize to larger programs and more diverse fault types, and whether alternative relevant slicers exhibit different behavior on the same benchmark.

References

- 1 Robin Abraham and Martin Erwig. Mutation operators for spreadsheets. *IEEE Transactions on Software Engineering*, 35(1):94–108, 2009. doi:10.1109/TSE.2008.73.
- 2 Rui Abreu, Peter Zoetewij, and Arjan J. C. van Gemund. On the accuracy of spectrum-based fault localization. In *Proceedings of the Testing: Academic and Industrial Conference – Practice and Research Techniques (TAIC PART)*, pages 89–98, 2006. doi:10.1109/TAIC.PART.2006.256442.
- 3 Khaled Ahmed, Mieszko Lis, and Julia Rubin. Slicer4J: A dynamic slicer for Java. In *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2021.
- 4 Hyunsook Do, Sebastian Elbaum, and Gregg Rothmel. SIR: Software infrastructure repository. <https://sir.csc.ncsu.edu>, 2005.
- 5 Eclemma Team. JaCoCo: Java Code Coverage Library. <https://www.eclemma.org/jacoco/>, 2025. Last accessed: April 2026.

- 6 James A. Jones and Mary Jean Harrold. Empirical evaluation of the Tarantula automatic fault-localization technique. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 273–282, 2005. doi:10.1145/1101908.1101949.
- 7 Roxane Koitz-Hristov, Thomas Sterner, Lukas Stracke, and Franz Wotawa. On the suitability of checked coverage and genetic parameter tuning in test suite reduction. *Journal of Software: Evolution and Process*, 36(8):e2656, 2024. doi:10.1002/smr.2656.
- 8 Bogdan Korel and Janusz Laski. Dynamic program slicing. *Information Processing Letters*, 29:155–163, 1988. doi:10.1016/0020-0190(88)90054-3.
- 9 Qusay Idrees Sarhan and Arpad Beszedes. Experimental evaluation of a new ranking formula for spectrum based fault localization. In *Proceedings of the 22nd IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 276–280, 2022. doi:10.1109/SCAM55253.2022.00038.
- 10 Jonas Schleich. JSR: Extended Version of the Java Test Suite Reduction Framework. <https://github.com/SchleichJonas/JSR>, 2025. Fork of the original JSR framework by Lukas Stracke; GitHub repository, version used in this study, accessed: 2026-05-20.
- 11 Jonas Schleich and Franz Wotawa. Combining dynamic slicing and spectrum-based fault localization – A first experimental evaluation. In *Proceedings of the 36th International Conference on Principles of Diagnosis and Resilient Systems (DX 2025)*, OASICS, pages 3:1–3:18, 2025. doi:10.4230/OASICS.DX.2025.3.
- 12 Lukas Stracke. JSR: Java Test Suite Reduction Framework. <https://github.com/Lms24/JSR>, 2024. GitHub repository, accessed: 2026-05-20.
- 13 Raja Vallée-Rai and Laurie J. Hendren. Jimple: Simplifying java bytecode for analyses and transformations. Technical Report 1998-4, Sable Research Group, McGill University, July 1998. URL: <https://api.semanticscholar.org/CorpusID:10529361>.
- 14 Xiangyu Zhang, Haifeng He, Neelam Gupta, and Rajiv Gupta. Experimental evaluation of using dynamic slices for fault localization. In *Proceedings of the Sixth International Symposium on Automated and Analysis-Driven Debugging (AADEBUG)*, pages 33–42, 2005. doi:10.1145/1085130.1085135.

Summary Of: Modelling Cyber-Physical Systems for Fault Diagnosis

Alexander Diedrich ✉ 

Helmut-Schmidt-University, Hamburg, Germany

Mattias Krysander ✉ 

Linköping University, Sweden

René Heesch ✉ 

Helmut-Schmidt-University, Hamburg, Germany

Oliver Niggemann ✉ 

Helmut-Schmidt-University, Hamburg, Germany

Abstract

This is an extended abstract of the manuscript 'Modelling Cyber-Physical Systems for Fault Diagnosis' [4] that was published in the journal IEEE Transactions on Systems, Man, and Cybernetics in December 2025.

2012 ACM Subject Classification Computing methodologies → Knowledge representation and reasoning

Keywords and phrases Fault Diagnosis, SMT, Multimode Systems

Digital Object Identifier 10.4230/OASICS.DX.2026.18

Category Extended Abstract

Related Version *Full Version:* <https://ieeexplore.ieee.org/abstract/document/11194731>

Funding *René Heesch:* This research as part of the project EKI is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr, which we gratefully acknowledge.

1 Extended Abstract

The complexity of cyber-physical systems, paired with significant availability requirements, require prompt reactions when faults occur [13]. To ensure timely fault handling, it is necessary to compare data from the system to a pre-existing model of nominal working behaviour and reason about possible faults [20]. The required model, that we will refer to as a *system description* (SD), is an essential but difficult-to-obtain part of the diagnostic process.

Overall, the system description models the structure and behaviour of technical systems [20]. It expresses for each individual component how it is connected to other components (its connections) and how its output is calculated from its inputs (its behaviour model) [8]. For a two-bit Boolean AND-gate, for example, the system description is defined as $\neg ab(c_{AND}) \rightarrow out = in_0 \wedge in_1$ [10], where the behaviour is given through the Boolean operator $\wedge$ and the connection is defined through relating the inputs $in_0, in_1 \in \{0, 1\}$ to $out \in \{0, 1\}$. The predicate $\neg ab(c_{AND})$ defines the healthiness (non-abnormality [20]) of the AND-gate. Such a formulation of the system description is called a weak-fault model, i.e., it models the component's normal behaviour and does not specify component behaviour under faults [23].

Consistency-based fault diagnosis [20] algorithms use such logical weak-fault system descriptions to infer faults by comparing the system description to actual observations of the system. If a fault exists in the system, the observations together with the system description



© Alexander Diedrich, Mattias Krysander, René Heesch, and Oliver Niggemann;
licensed under Creative Commons License CC-BY 4.0

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 18; pp. 18:1–18:6

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

lead to an inconsistency. Using non-monotonic reasoning [21], a fault diagnosis algorithm [22] is able to infer potential faulty components (also known as diagnoses or root causes) from this inconsistency. Until now, system descriptions have usually been used for diagnosing Boolean circuits [20, 10], while only a few authors have extended propositional logic system descriptions beyond systems with only Boolean circuits [11, 5, 17].

What is so far missing from current literature is a unified formulation of system descriptions that allows reasoning over sufficiently complex systems, such as those generating continuous signals and operating in different modes. Obtaining such a more elaborate system description is highly desirable for performing fault diagnosis of cyber-physical systems [19, 15]. These systems are used in most modern applications and combine intelligent monitoring and control with classical mechanical system components. Because modern applications require constant adjustments and updates, those systems must adapt over time, contain proprietary components with complicated behaviour models, operate in several modes, and generate Boolean, integer, and continuous signals.

To tackle these challenges we pose the following three research questions within the article.

RQ1: *Is there a consistent logical theory to extend classical Boolean consistency-based fault diagnosis to work with operating modes and continuous signals?*

RQ2: *How is it possible to diagnose faults, provided a consistent logical system description for modes and continuous signals exists?*

RQ3: *Can the approach be adapted to reasoning approaches requiring less expert knowledge?*

To answer all three research questions and obtain a consistent logical theory for diagnosing faults in cyber-physical systems, the article presents a novel formulation in SMT over nonlinear real arithmetic (SMT- $\mathcal{NRA}$) [3] for consistency-based fault diagnosis. We show how to use established fault diagnosis algorithms adapted to an SMT- $\mathcal{NRA}$ system description, and introduce the novel HySD_ENHANCED algorithm for a direct approach to consistency-based fault diagnosis. Overall, this article presents the following contributions:

- We present a novel SMT- $\mathcal{NRA}$ formulation that includes operating modes, continuous signals, and behaviour models.
- Using the SMT- $\mathcal{NRA}$ formulation, we adapt classical consistency-based fault diagnosis theory [10, 16] and introduce the novel algorithm HySD_ENHANCED to perform fault diagnosis.

We evaluated our approach on two benchmarks of multiple-tank systems [9, 1], two Boolean circuits, and a use-case of dynamically configurable battery packs. While we use the two benchmarks and the battery pack example to evaluate the capabilities and limitations of our approach, we use the Boolean circuits to show that our method also generalizes to classical Boolean circuit fault diagnosis. With our experiments we are able to diagnose 97% of the injected faults.

Underlying our approach are some assumptions that we need to state explicitly: i) While in previous work we presented methods to learn and validate propositional logic system descriptions [6, 17, 7] and there is ongoing research to learn SMT expressions from data [14], but there is so far no approach to learn the system expressions in this article in their entirety. ii) While parts of the presented system description have been proven to be learnable, for example, behaviour models or state estimations, there is still some work required to fit these into our presented system description. iii) We assume a cyber-physical system can be subdivided into a set of operating modes that is significantly smaller than the true operating modes in the data. Earlier work showed that through the use of discretisation functions, such a subdivision is well attainable [17, 18].

The main benefit of our approach is that the presented type of system description is much more general than traditional propositional logic system descriptions [17, 6], i.e., has broader variable domains, is able to handle operating modes, and supports inference from symptoms to faulty components.

The main idea behind using SMT- $\mathcal{NRA}$ for consistency-based fault diagnosis is to formulate the system description in such a way that components are *suitably* associated to relevant observations. This means we first require a subset $COMPS' \subset COMPS$ that influences the health of a single observation. For this article it is reasonable to assume that such a set $COMPS'$ exists [17, 6]. The general form of these system descriptions is

$$\bigwedge_{obs_j \in OBS} \left(\bigwedge_{c_i \in COMPS_k} \neg ab(c_i) \rightarrow obs_j \right) \quad (1)$$

where $COMPS_k \subset COMPS$, with $k \in \mathbb{N}$. In other words, changes in the health state of a component are assumed to have a direct influence on a particular observation. The goal in this article is to use this architecture of formulating logical expressions and significantly increase its generality. The intuition is to associate component health states with discretisations of the expected and actual state and the expected and actual observations.

$$\bigwedge_{c_j \in COMPS'} \neg ab(c_j) \rightarrow ((d_s(\mathcal{V}) = d_s(\mathcal{V}')) \wedge (|g_m(\mathcal{V}) - \alpha_i| \leq \epsilon)) \quad (2)$$

$COMPS' \subset COMPS$ is a set of components relevant for the current formula (e.g., their behaviour has an influence on the mode and measurements of the system). Function $d_s(\cdot)$ computes the expected ($d_s(\mathcal{V})$) and actual mode ($d_s(\mathcal{V}')$) through state estimation methods, where $\mathcal{V} \neq \mathcal{V}'$. This means the expected mode is calculated from variables $\mathcal{V}$ describing how the system should behave, while variables $\mathcal{V}'$ describe actual measurements from the system. Function $g_m(\mathcal{V})$ computes the expected measurement given mode $m \in \mathcal{N}$, while α_i is a single actual measurement. $\epsilon \in \mathbb{R}$ denotes a noise threshold. We use the $\neg ab$ -predicates to denote non-abnormal behaviour [20].

To perform fault diagnosis, it is necessary to compare the SMT- $\mathcal{NRA}$ system description with current observations. The comparison is done by computing an expected value of the measurement, i.e., the expectation of normal behaviour $g(\cdot)$, given the current operating mode. To compare the prediction with the actual measurement α_i it is possible to calculate a residual value $r_i \in \mathbb{R}$ through

$$r_i = \begin{cases} |g(\mathcal{V}) - \alpha_i| \leq \epsilon; \text{ no fault} \\ |g(\mathcal{V}) - \alpha_i| > \epsilon; \text{ else} \end{cases} \quad (3)$$

to account for noise $\epsilon \in \mathbb{R}$. Discretising the residual into a Boolean value by only computing the rightmost side of Equation (2) then leads to a computable interpretation. In other words, an expert-defined physical model calculates the expected value for some measurement. The difference between the expected and the actual value form the residual. The residual is then discretised to obtain a Boolean value indicating the fault symptom. SMT- $\mathcal{NRA}$ provides the advantage that all necessary calculations can be performed within the same logical calculus and can be proven through solvers such as Z3 or MathSAT.

A fault that is detected due to a non-zero residual is called a symptom. Symptoms occur in one of four cases:

1. The behaviour model outputs too much noise or the model error is too large. This is a system description design error and must be accounted for.

2. The behaviour model does not match to the current operating mode. This is a system description design error and must be accounted for.
3. Some sensor fails and the residual becomes non-zero. In this case, fault diagnosis is required to determine the minimum cardinality diagnosis.
4. Some component fails and due to deviating observations, the residual becomes non-zero. In this case, fault diagnosis is required to determine the minimum cardinality diagnosis.

To diagnose faults that are due to changes in the operating mode, the operating mode needs to be made explicit within the system description [17, 2]. To compare the expected mode to the actual mode using SMT- $\mathcal{NRA}$ several computations need to take place: i) Function $d_s(\mathcal{V})$ needs to calculate the expected discretised observable modes from the weak-fault behaviour model and the known variables of the system. ii) Function $d_s(\mathcal{V}')$ needs to compute the actual discretised observable modes from the known variables and the current observations of the system. Again, SMT- $\mathcal{NRA}$ allows the formulation of non-linear functions directly within the logical calculus.

We define a mode fault as a fault that, given two discretised model variables $m, m_a \in \mathbb{Z}$ computed by $d_s(\mathcal{V})$ and $d_s(\mathcal{V}')$, respectively, a mode fault occurs when the actual mode differs from the predicted mode $m \neq m_a$.

In the article we evaluate, whether we can find faults in several cyber-physical systems, with the SMT logic system description presented in Equation (2). Within the evaluation we use a representative set of several exemplary use-cases from the process industry, a set of configurable batteries, and a set of Boolean circuits. For this abstract, we only present a subset of the results. Please note that in tables with empirical results bold symbols denote diagnosis of the true root cause (i.e. the injected fault). Diagnoses in italics denote wrong diagnoses. Diagnoses that are neither bold, nor in italics are diagnoses allowed by the model due to partial observation.

For all experiments we used a threshold value ϵ of 0.001. This was sufficient as we used analytical models to implement function $g_m(\cdot)$, such that the difference between measurement and prediction is close to zero in the nominal case. In practice, ϵ may be adjusted by comparing the variance in the prediction of g_m and the variance of α in the fault-free mode and over a representative set of historical data.

■ **Table 1** Empirical results for two well-known electronic circuits.

System	Fault	Mode	Diagnosis
Full-Adder	f0	Normal	{f0} , {f5,f6}, {f3}
	f1	Normal	{f1}
	f5	Normal	{f5} , {f0}, {f3}
	None	Normal	$\emptyset$
Polybox	m0	Normal	{m0} , {m1}, {a0}
	m1	Normal	{a0,a1}, {m1} , {a0}, {m2,m0}
	a0	Normal	{m0} , {m1}, {a0}
	None	Normal	$\emptyset$

Our approach was evaluated on a half-adder circuit and the well-known polybox example. The results are presented in Table 1. Results are similar to the process industrial use-cases above. However, the systems do only contain one operating mode and exhibit a lower observability. Still, in each experiment to correct fault diagnosis was computed. But due to the limited observability the diagnosis sets were larger.

Table 2 presents the diagnostic results of investigating all single faults for a multi-mode battery pack. The columns from left to right show the actuated mode, the injected fault, whether the fault directly affects an observed variable, and the single fault diagnoses. An

■ **Table 2** Empirical results for a multi-mode battery pack.

Mode	Fault	Diagnosis
Bypass	NF	$\emptyset$
	f_i_pack	$\emptyset$
	f_i_1	{f_i_1}
	f_v_1	{f_v_1}, {f_cell_1}
	f_cell_1	{f_v_1}, {f_cell_1}
	mode = Forward	$\emptyset$
	mode = Backward	$\emptyset$
Forward	NF	$\emptyset$
	f_i_pack	{f_i_pack}
	f_i_1	{f_i_1}
	f_v_1	{f_v_1}, {f_cell_1}
	f_cell_1	{f_v_1}, {f_cell_1}
	mode = Backward	{f_i_pack}, {f_mode}
	mode = Bypass	{f_mode}
Backward	NF	$\emptyset$
	f_i_pack	{f_i_pack}
	f_i_1	{f_i_1}
	f_v_1	{f_v_1}, {f_cell_1}
	f_cell_1	{f_v_1}, {f_cell_1}
	mode = Forward	{f_i_pack}, {f_mode}
	mode = Bypass	{f_mode}

empty set diagnosis indicates that the system is fault-free, i.e., no fault is detected. An isolability analysis study is presented in [12], covering all faults except mode faults. The results here are consistent with the analysis results there. In non-ideal cases where the fault is either not detected or not uniquely isolated, it is due to a lack of information, that is the diagnosis inference is complete. Furthermore, the proposed diagnostic computational framework also handles fault-changing modes.

References

- 1 Kaja Balzereit, Alexander Diedrich, Jonas Ginster, Stefan Windmann, and Oliver Niggemann. An ensemble of benchmarks for the evaluation of ai methods for fault handling in cpps. In *19th IEEE International Conference on Industrial Informatics*, November 2021.
- 2 Andreas Bunte, Benno Stein, and Oliver Niggemann. Model-based diagnosis for cyber-physical production systems based on machine learning and residual-based diagnosis models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33(01), pages 2727–2735, 2019. doi:10.1609/AAAI.V33I01.33012727.
- 3 Alessandro Cimatti, Alberto Griggio, Ahmed Irfan, Marco Roveri, and Roberto Sebastiani. Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions. *ACM Transactions on Computational Logic (TOCL)*, 19(3):1–52, 2018. doi:10.1145/3230639.
- 4 Alexander Diedrich, Mattias Krysander, Rene Heesch, and Oliver Niggemann. Modeling cyber-physical systems for fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2025.
- 5 Alexander Diedrich, Alexander Maier, and Oliver Niggemann. Model-based diagnosis of hybrid systems using satisfiability modulo theory. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*. Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19), 2019.
- 6 Alexander Diedrich, Lukas Moddemann, and Oliver Niggemann. Learning system descriptions for cyber-physical systems. In *Proceedings of 12th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, 2024.

- 7 Alexander Diedrich, Lukas Moddemann, and Oliver Niggemann. On validating propositional logic system descriptions for fault diagnosis. *Engineering Applications of Artificial Intelligence*, 165:113379, 2026. doi:10.1016/J.ENGAPPAI.2025.113379.
- 8 Alexander Diedrich and Oliver Niggemann. On residual-based diagnosis of physical systems. *Engineering Applications of Artificial Intelligence*, 109:104636, 2022. doi:10.1016/j.engappai.2021.104636.
- 9 Jonas Ehrhardt, Malte Ramonat, Rene Heesch, Kaja Balzereit, Alexander Diedrich, and Oliver Niggemann. An ai benchmark for diagnosis, reconfiguration & planning. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2012.
- 10 Alexander Feldman, Gregory Provan, and Arjan Van Gemund. Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research*, 38:371–413, 2010. doi:10.1613/JAIR.3025.
- 11 Alban Grastien. Diagnosis of hybrid systems with smt: opportunities and challenges. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, pages 405–410. IOS Press, 2014. doi:10.3233/978-1-61499-419-0-405.
- 12 Fatemeh Hashemniya, Arvind Balachandran, Erik Frisk, and Mattias Krysander. Structural diagnosability analysis of switched and modular battery packs. In *2024 Prognostics and System Health Management Conference (PHM)*, pages 362–369. IEEE, 2024.
- 13 Rolf Isermann and Peter Balle. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control engineering practice*, 5(5):709–719, 1997.
- 14 Samuel Kolb, Stefano Teso, Andrea Passerini, and Luc De Raedt. Learning smt (lra) constraints using smt solvers. In *IJCAI International Joint Conference on Artificial Intelligence*, volume 2018, pages 2333–2340. ijcai. org, 2018. doi:10.24963/IJCAI.2018/323.
- 15 Ion Matei, Maksym Zhenirovskyy, Johan de Kleer, and Alexander Feldman. A hybrid qualitative and quantitative diagnosis approach. In *Annual Conference of the PHM Society*, volume 11(1), 2019.
- 16 Amit Metodi, Roni Stern, Meir Kalech, and Michael Codish. A novel sat-based approach to model based diagnosis. *Journal of Artificial Intelligence Research*, 51:377–411, 2014. doi:10.1613/JAIR.4503.
- 17 Lukas Moddemann, Henrik Sebastian Steude, Alexander Diedrich, and Oliver Niggemann. Discret2di - deep learning based discretization for model-based diagnosis. In *Proceedings of 12th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, 2024.
- 18 Phillip Johann Overlöper, Lukas Moddemann, Nemanja Hranisavljevic, Alexander Windmann, and Oliver Niggemann. Discretization of cps time series with neural networks. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2024. doi:10.1109/ETFA61755.2024.10710901.
- 19 André Platzer. *Logical foundations of cyber-physical systems*. Springer, 2018. doi:10.1007/978-3-319-63588-0.
- 20 Raymond Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987. doi:10.1016/0004-3702(87)90062-2.
- 21 Raymond Reiter. Nonmonotonic reasoning. In *Exploring artificial intelligence*, pages 439–481. Elsevier, 1988.
- 22 Patrick Rodler. How should i compute my candidates? a taxonomy and classification of diagnosis computation algorithms. In *33rd International Workshop on Principle of Diagnosis – DX 2022*, 2022.
- 23 P Struss. Model-based on-board diagnosis and tools for the developer of on-board systems-vmdb and idd: Two projects of the european car industries. *MONET Newsletter Issue 2 - Automotive Supplement*, 2002.

Uncertainty-Aware Fault Diagnosis of Unknown Faults Using Ensemble-Based NODE Residuals

Daniel Jung   

Linköping University, Sweden

Theodor Westny   

Linköping University, Sweden

Abstract

This extended abstract presents an uncertainty-aware approach to data-driven fault diagnosis using ensembles of neural ordinary differential equations (NODE). The proposed method leverages physical insights in the design of NODE models for residual generation and employs bootstrap ensembles to quantify prediction uncertainty and adapt detection thresholds. The adaptive thresholds enable robust detection and isolation of faults in scenarios with limited training data and reduces the risk of false alarms due to out-of-distribution data. The methodology is validated on real-world data from the LiU-ICE benchmark, providing insights into how training data quality and residual design impact uncertainty-aware fault diagnosis.

2012 ACM Subject Classification Computing methodologies

Keywords and phrases Data-driven diagnosis, model-based diagnosis, structural methods, neural ordinary differential equations, ensemble models

Digital Object Identifier 10.4230/OASICS.DX.2026.19

Category Extended Abstract

Funding *Daniel Jung*: D. Jung is partially funded by the Swedish excellence center ELLIIT.

Theodor Westny: T. Westny was with the Division of Vehicular Systems at Linköping University.

1 Introduction

This is an extended abstract based on the journal paper:

Jung, D., & Westny, T. (2026). Uncertainty-aware fault diagnosis of unknown faults using ensemble-based NODE residuals. *Mechanical Systems and Signal Processing*, 242, 113599.

Machine learning (ML) and data-driven modeling for fault diagnosis use training data to model the relationship between different signals. The purpose of the models is to detect inconsistencies between the model predictions and observations (fault detection) and compute diagnoses (fault isolation). Still, data-driven fault diagnosis is complicated by the fact that most faults are rare events, thus, collecting relevant training data is challenging, expensive, and sometimes infeasible. Because of the lack of representative training data in data-driven fault diagnosis, one challenge is to distinguish whether a detected anomaly is caused by a faulty component or model inaccuracies caused by out-of-distribution (OOD) data.

An ensemble-based NODE approach is proposed to address the challenges of OOD data in data-driven fault diagnosis. The solution applies data-driven residuals for fault detection as well as adaptive thresholds that capture aleatoric and epistemic uncertainties. The adaptive threshold is used to reduce the risk of false alarms when the model accuracy is low. While neural networks possess the capacity to learn complex relationships between signals, they are



© Daniel Jung and Theodor Westny;

licensed under Creative Commons License CC-BY 4.0

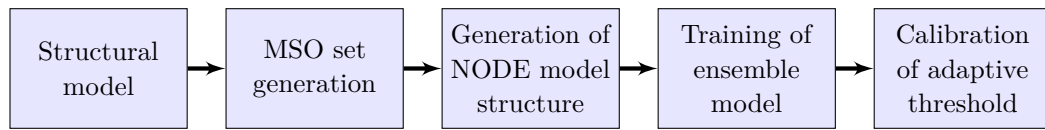
37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).

Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 19; pp. 19:1–19:6

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** A summary of the NODE-based residual generator design process.

inherently susceptible to overfitting to the training domain, which can limit their ability to generalize under distribution shifts or operating conditions that are insufficiently represented in the training data.

The use of data-driven residuals based on Neural Ordinary Differential Equations (NODE) models derived from physical insights has been successfully proposed before, e.g., in [7]. The data-driven residuals can be applied in a consistency-based diagnosis framework to reason about faults even though training data from faults is scarce. However, fault isolation is complicated by false alarms caused by OOD data which motivated the need to capture the model validity or prediction accuracy of the data-driven residuals in relation to training data.

The focus of this work has been to avoid false alarms by using ensemble models instead of single NODE models to estimate model inaccuracies. The motivation is the advantage of ensemble models to simplify the calibration of an adaptive threshold since the ensemble model approximates prediction uncertainties even for complex models. A summary of the design process is illustrated in Figure 1. The proposed method is evaluated using the LiU-ICE benchmark [5] as a case study and several important conclusions have been identified for systematic design of data-driven residuals and efficient use of training data.

2 Design and Evaluation of NODE-Based Residuals

In model-based diagnosis, residual generators are constructed based on mathematical models derived based on physical insights about the system to be monitored. A structural representation of the mathematical model can be used to perform diagnosability analysis and to find redundant equation sets for residual generation using the Dulmage-Mendelsohn's decomposition [6]. Based on the structural model of the system and available measurements, a number of Minimally Structurally Over-determined (MSO) candidate sets are derived using the Fault Diagnosis Toolbox [3]. From the redundant equation sets, e.g., MSO sets, computational graphs describing the relationship between different signals describe how to compute the residual generators. Based on these computational graphs, the NODE network model structures are derived. Here, a subset of the MSO sets is selected based on three criteria:

- i) It must be possible to construct a residual generator in state-space form.
- ii) The target signal must have sufficient excitation for the NODE model to learn the system dynamics.
- iii) When there are multiple MSO sets that have the same target signal, select the MSO set with the largest set of input signals.

The first two criteria are needed for efficient implementation and training of the NODE models. The first criterion is fulfilled if the MSO set can be realized with integral causality [2]. The second criterion is determined by selecting NODE model candidates where the target signal span a wide range of values with respect to the operating range. For the LiU-ICE model, residual candidates are selected where pressure or air mass flow signals are used as target signals, since they vary significantly during a driving cycle, but not, e.g., temperatures, since they vary slowly during a driving cycle. The third criterion is added to maximize prediction accuracy by providing a maximum number of informative input signals. Note

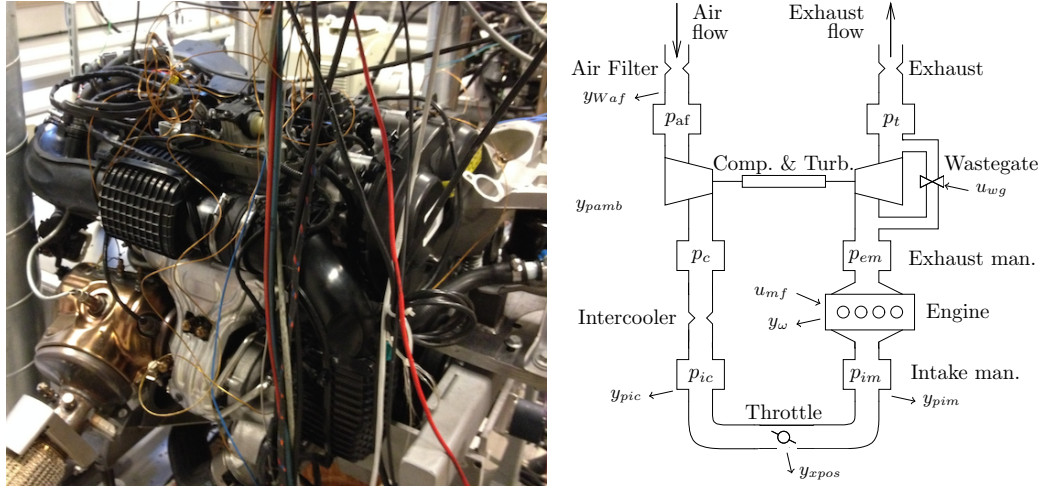


Figure 2 The LiU-ICE benchmark lab setup showing the engine test bench in the left picture and a schematic of the air path of the internal-combustion engine in the right picture.

that there is a design trade-off between maximizing detection performance and isolation performance. Since the focus in this work is on the use of ensemble models for reducing the risk of false alarms caused by OOD data, the MSO sets are selected to maximize detection performance. However, the ensemble-based methods in this work can be applied to systematic residual selection for fault detection and isolation, see, e.g., [4].

3 Data-Driven Residual Design for the LiU-ICE Benchmark

The LiU-ICE benchmark is used as a case study, which considers the air path through an internal-combustion engine, see Figure 2. A model structure is available derived from component models described in, e.g., [1]. The set of available measurements reflects a standard configuration in a production vehicle, comprising eight sensor signals: intercooler pressure y_{pic} , intake manifold pressure y_{pim} , ambient pressure y_{pamb} , intercooler temperature y_{Tic} , ambient temperature y_{Tamb} , air mass flow after air filter y_{Waf} , engine speed y_ω , throttle position y_{xpos} , as well as two actuator signals: wastegate position u_{wg} , and injected fuel mass into the cylinders u_{mf} .

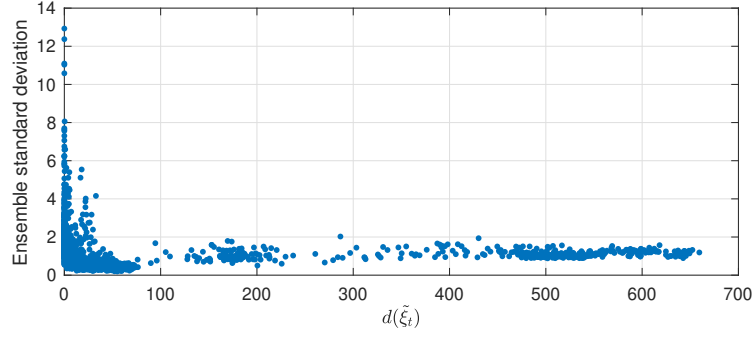
The selected NODE model structures are derived from three selected MSO sets derived based on the structural model provided in the LiU-ICE benchmark. Each MSO set defines a specific residual generator structure with its corresponding input signals and output target. The general NODE model structure is in the form:

$$\begin{aligned} r_i &= y_i - h_i(x_i) \\ \dot{x}_i &= g_i(x_i, u_i) \end{aligned} \quad (1)$$

where r_i is the residual, y_i is the target signal, x_i is an internal state vector and u_i are the model inputs based on the selected MSO set. For example residual r_1 is derived based on MSO 591_85 as:

$$\begin{aligned} r_1 &= y_{pic} - h_1(x_1) \\ \dot{x}_1 &= g_1(x_1, u_1) \\ u_1 &= [y_{Tamb}, y_{Waf}, y_{xpos}, y_\omega, y_{pamb}, y_{pim}, u_{wg}, u_{mf}] \end{aligned} \quad (2)$$

In addition, two more residuals, r_2 and r_3 , are implemented with target signals y_{pim} and y_{Waf} , respectively.



■ **Figure 3** Example of the standard deviation of the ensemble prediction model as a function of $d(\tilde{\xi}_t)$. The standard deviation is almost constant for larger $d(\tilde{\xi}_t)$ while it varies significantly when $d(\tilde{\xi}_t)$ is close to zero.

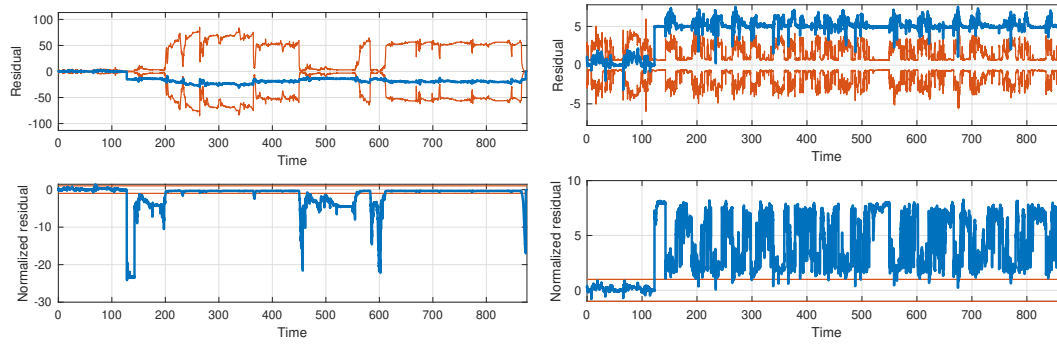
The models were implemented in PyTorch [8] and trained on fault-free data collected from different driving cycles to minimize the mean squared error (MSE) between the model prediction and the target signal. The layer parameters in the NODE models were initialized using stability-informed initialization [9]. The Runge-Kutta (RK4) ODE solver was set to a fixed time step of 0.05 seconds, which is consistent with the sampling rate of the input signals.

4 Evaluation

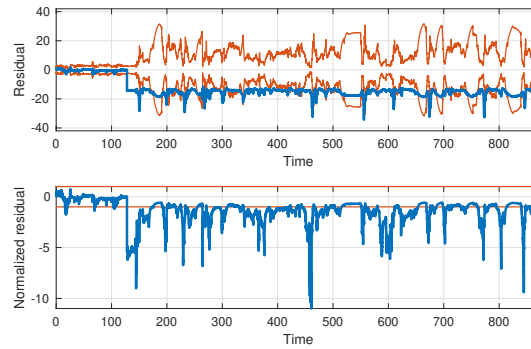
The principle used to identify when anomalies are likely caused by faults and not OOD data, is when the residual output exceeds a level that can be explained by model inaccuracy alone. Since the model parameters are optimized by minimizing the MSE cost function, training will prioritize fitting to operating conditions that are well represented in training data. Thus, the idea is that model accuracy of the ensemble output is evaluated as a function of density of training data, here denoted $d(\xi_t)$ where $\tilde{\xi}_t = (\tilde{u}_t, \tilde{x}_t)$ represents the model inputs and internal states, for different operating conditions. The generalizability of the model is assumed to degrade as soon as the model inputs ξ_t become OOD. To quantify OOD, the local density of the sample $\tilde{\xi}_t$, is computed by counting the number of training samples within a predefined neighborhood.

The ensemble standard deviation $\hat{\sigma}$ can be used as a measure of prediction uncertainty. Figure 3 shows the results of one trained residual that $\hat{\sigma}$ is almost constant when $d(\tilde{\xi}_t)$ is sufficiently large while it varies more as $d(\tilde{\xi}_t)$ goes to zero. Similar results are observed for other residuals and datasets, indicating that it is not sufficient to analyze the range of data when training a model but also the density in different operating conditions. Experiments also show that fault detection performance drops significantly when $d(\tilde{\xi}_t)$ is small and faults become easier to detect in operating conditions where $d(\tilde{\xi}_t)$ is large.

The results show that it can be easier to detect a small fault than a large fault due to increasing model inaccuracies due to OOD data when a larger fault occurs. The left plot in Figure 4 shows the residual output during a fault in sensor y_{pic} with a fault magnitude of -15% and the adaptive threshold computed based on the ensemble output variance. The fault occurs at 120 s, but the adaptive threshold exceeds the residual output resulting in missed detections. The right plot in Figure 4 shows the corresponding residual output of a fault in the same sensor, but the magnitude is +5%. Even though the threshold is varying during the scenario, the residual exceeds the threshold. Thus, the detection accuracy is higher for the small fault size.



■ **Figure 4** Evaluation of r_1 on the fault scenario $f_{ypic} = -15\%$ (left plot) and the fault scenario $f_{ypic} = +5\%$ (right plot) during the ftp75 city2 cycle. The OOD input data for the larger fault result in a higher threshold which increases the missed detection rate with respect to the smaller fault.



■ **Figure 5** Evaluation of the ensemble model r_1 on the same fault scenario as in the left plot of Figure 4 but with an augmented training dataset. The adaptive threshold varies less, resulting in better detection performance.

To illustrate the impact of training data, the same NODE model for r_1 is trained with an augmented training dataset with a higher density of data from various operating conditions from additional drive cycles. With respect to the residual output in the left plot in Figure 4, Figure 5 shows the residual output from the same fault scenario, but the residual model trained on the augmented dataset. It is visible that the threshold does not increase as much during the fault scenario, because the augmented training data covers a larger variety of operating conditions. This leads to that the residual exceeds the threshold more consistently.

The results from the experiments have been validated on other residual models derived from the other MSO sets.

5 Conclusions

This study highlights the importance of handling OOD data in data-driven fault diagnosis to improve the ability to reason about previously unseen faults. Experiments show that ensemble methods can effectively quantify prediction uncertainty and thereby reduce false alarms due to OOD inputs. This advantage, however, is accompanied by a higher risk of missed detections for large faults. The results underscore the need for systematic experimental design when constructing compact yet informative training datasets. Moreover, model selection, especially the choice of input and target signals, is crucial, as OOD inputs can severely impair detection performance. At the same time, variability in ensemble predictions offers a valuable complement to residuals for identifying abnormal behavior.

References

- 1 L. Eriksson and L. Nielsen. *Modeling and control of engines and drivelines*. John Wiley & Sons, 2014.
- 2 E. Frisk, A. Bregon, J. Åslund, M. Krysander, B. Pulido, and G. Biswas. Diagnosability analysis considering causal interpretations for differential constraints. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 42(5):1216–1229, 2012. doi:10.1109/TSMCA.2012.2189877.
- 3 E. Frisk, M. Krysander, and D. Jung. A toolbox for analysis and design of model based diagnosis systems for large scale models. *IFAC-PapersOnLine*, 50(1):3287–3293, 2017.
- 4 D. Jung and E. Frisk. Residual selection for fault detection and isolation using convex optimization. *Automatica*, 97:143–149, 2018. doi:10.1016/J.AUTOMATICA.2018.08.006.
- 5 D. Jung, E. Frisk, and M. Krysander. The liu-ice benchmark—an industrial fault diagnosis case study. *arXiv preprint*, 2024. arXiv:2408.13269.
- 6 M. Krysander, J. Åslund, and M. Nyberg. An efficient algorithm for finding minimal overconstrained subsystems for model-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(1):197–206, 2007. doi:10.1109/TSMCA.2007.909555.
- 7 A. Mohammadi, M. Krysander, and D. Jung. Consistency-based diagnosis using data-driven residuals and limited training data. *Control Engineering Practice*, 159:106283, 2025.
- 8 Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems (NeurIPS)*, 2019.
- 9 Theodor Westny, Arman Mohammadi, Daniel Jung, and Erik Frisk. Stability-informed initialization of neural ordinary differential equations. In *International Conference on Machine Learning (ICML)*, 2024.