Master's thesis

2022

Ravitej Bhagavathi

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Ravitej Bhagavathi

Predictive Digital Twin of an Unmanned Surface Vehicle

with Actuator Fault Diagnosis on Otter USV as a demonstrative feature

June 2022

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Norwegian University of
Science and Technology

Predictive Digital Twin of an Unmanned Surface Vehicle

with Actuator Fault Diagnosis on Otter USV as a demonstrative feature

Ravitej Bhagavathi

Master of Science in Simulation and Visualization
Submission date: June 2022
Supervisor: Agus Hasan
Co-supervisor: Giorgio Minde Kufoalor

Norwegian University of Science and Technology
Department of ICT and Natural Sciences

June 10, 2022

Abstract

Systems of the present day are getting increasingly complex with a higher degree of autonomy performing safety critical tasks.Digital twins, that are capable of representing these systems digitally throughout their life-cycle, are slowly replacing traditional simulation models.Technologies like IoT, ML, AI and Cloud Computing are making this possible faster that ever. Digital twins are already a reality in many industries showing their technical and commercial potential. On the other hand, Unmanned Surface Vehicles(USV) - a type of marine autonomous systems are getting popular in the robotics community for their ability to operate in complex and remote environments.This makes them suitable candidates for digital twin development. However, to the author's knowledge, the power of digital twins has not yet been fully realized in case of USVs. The current thesis addresses this gap with an aim to develop a digital twin for Otter - an Unmanned Surface Vehicle from Maritime Robotics, Norway. In doing so, two objectives are considered. Firstly, 'Probabilistic Graphical Models' are used as a mathematical framework for the digital twin due to their capability of handling complexity and uncertainty of systems. Also, these models provides a general framework that can accommodate different modelling techniques making them suitable for several digital twin applications. Secondly, the above framework is used to perform 'Actuator Fault Diagnosis' on Otter's actuators as a demonstrative application of the digital twin.Necessary theoretical treatment is given to both the above concepts.Faults were introduced into the actuator subsystem by means of a broken port side propeller and experiments were conducted on both the faulty and faultless Otter. Subsequently, the faults were identified using an Adaptive Extended Kalman Filter algorithm with promising results.The results are thoroughly discussed with regards to both their meaning and the value they add in the context of digital twins.Finally, recommendations for future work are made.
Keywords: Digital Twin, Unmanned Surface Vehicle, Otter, Probabilistic Graphical Model, Actuator Fault Diagnosis.

Preface

This monograph is submitted as a partial fulfillment of the requirements for the degree of *Master of Science in Simulation and Visualization* at the Norwegian University of Science and Technology(NTNU). This work is conducted at the Department of ICT and Natural Sciences under the supervision of Prof.Agus.S.Hasan. Giorgio D.K.M Kufuaolor from Maritime Robotics is the co-supervisor for the project

Acknowledgement

I express sincere gratitude to my supervisor, Prof. Agus Hasan, for being instrumental in this thesis work,right from conception to conclusion.His support in arranging my travel to Trondheim for Otter training, carrying out experiments and interpreting the results was crucial in delivering a quality work. I also extend my thanks to co-supervisor Giorgio Minde Kufoalor from Maritime Robotics for his guidance and critical feedback at all stages of the project. Both of them, through bi-weekly interactions, ensured the thesis was on track while also giving me the flexibility to pursue my interests.I also thank current graduate students Jincheng Liu and Ege Kandemir for helping me out with the field experiments.Overall, this thesis was a great learning experience to me and enabled me to understand the concepts and potential of Digital Twins.

Finally, I thank my mother for being a constant pillar of support in all my endeavours. I also thank my girlfriend for being there in times of doubt and helping me swim through the masters programme,sometimes against the tide.

Contents

Figures

Tables

Acronyms

AEKF Adaptive Extended Kalman Filter. 8, 22, 35, 36, 44, 47, 48, 51, 52, 54, 67

AKF Adaptive Kalman Filter. 22

GNSS Global Navigation Satellite Systems. 3, 8, 36

RMSE Root-mean square error. 68

USV Unmanned Surface Vehicle. iii, 3

Chapter 1

Introduction

This chapter introduces the broad concepts related to this thesis, identifies the problems it aims to solve and my contribution in addressing the same.

1.1 Background

As the title indicates, this thesis lies at the intersection of two emerging areas of interest in modern engineering - Digital Twins and Unmanned Vehicles, a class of marine autonomous systems.Therefore, we begin our discussion with a brief introduction to the core concepts of this thesis i.e., Digital Twins and Unmanned Surface Vehicles.

1.1.1 Digital Twins

From the times of antiquity, the use of models - both physical and abstract, has been central in various fields of science, engineering and technology. Whether it is the scaled architectural models that guide the construction of the built environment or mathematical models that help describe the motion of planetary objects and man-made satellites, models play a key role in understanding real-world phenomena and physical objects. Ever since computers came into existence, system modelling has also been digitized and the applications have grown multi-fold. For example, computer-aided design (CAD) programs make it easier to create and animate visual geometric models and computer-aided engineering (CAE) software run complex simulations based on numerical models. These digital tools enable their users to fully harness the power of modelling and simulation in the design and analysis of a wide range of physical systems and processes. Finite-element modelling and simulation helps in the design of safer bridges and buildings, computational fluid dynamic simulations which are vital in the aerodynamic and hydrodynamic analysis of fuel-efficient aircraft and ships respectively.

Though it is needless to stress the importance of modelling and simulation in the design stage, their usage does not progress into the operational stage. This is due to the inherent simplifications made in the model which can no longer be valid due to changing system parameters and complex interaction with the environment. Therefore, more advanced digital tools are needed to represent the system through its entire life-cycle. 'Digital twins' are a recent technological trend in engineering that can bridge the gap and provide a platform for continuous interface with a system throughout its life-cycle.Together with other emerging areas in technology like IoT, AI and ML, Digital Twins are expected to play a huge role in Industry 4.0

As mentioned above, unlike traditional models which are of limited use during service, 'Digital twins' incorporate dynamic system models which are effective especially during the system operation. A more formal definition of 'Digital twins' is given in Rasheed et al.[1] as a virtual representation of a physical asset enabled through data and simulators for real-time prediction, optimization, monitoring, controlling, and improved decision making. Thus, the continuous supply of real-time data makes 'Digital twins' more powerful than the traditional models. Rapid development in industrial and consumer hardware with integrated electronic sensors is making it easier than ever before to collect vast amounts of data. This availability of data coupled with other emerging technologies like IoT, Cloud Computing, Artificial Intelligence and Machine Learning is making data accessible and resourceful - therefore the term 'Big Data' has sprung in popularity. 'Digital twins' is an umbrella concept that encompasses all the above-mentioned technologies to create a seamless interface with an asset or a system at all times.

The concept of 'Digital twin' was conceived by Grieves.M [2] in 2002. Ever since many other terms referring to a similar concept like - computational mega model, device shadow, mirrored system, product avatar etc., have been in use in academia and industry [1]. It is important to note that, 'Digital twins' is not just an abstract concept. It is a framework that is already in place in a wide range of industries spanning manufacturing, health-care, smart cities and autonomous systems etc., For example, Kongitwin is a dynamic digital twin interface developed for oil-gas and energy sectors [3]. A digital twin is created for the whole city of Singapore [4] to facilitate better urban planning and improved decision making.

This thesis focuses on the application of 'Digital twins' in the case of autonomous systems, which are a special class of physical systems that operate in remote environments with minimum or no human supervision.

1.1.2 Unmanned Surface Vehicles

Unmanned Surface Vehicles or USVs are a special class of autonomous mobile systems that perform a wide variety of tasks in challenging environments without any human intervention [5]. Owing to their advanced instrumentation and autonomy, these vehicles have interesting research and military applications like bathymetry, harbour security, ocean environmental monitoring, search and rescue, reconnaissance etc.

Despite their growing popularity and usage, several USVs operate with limited autonomy due to challenges faced in reliable guidance, navigation and control systems (GNC), and sensor, actuator and communication failures [5]. These challenges need to be addressed for USVs to operate with full autonomy and minimize the reliance on human intervention.

This project explores the potential of 'Digital Twins' in overcoming some of the aforementioned challenges not just in USV research and development but also in their service. For example, the simulation and visualization capabilities of Digital Twins can be utilized for developing and testing advanced GNC(Guidance, Navigation and Control) algorithms for USVs. Similarly, the data-driven nature of Digital Twins enables condition monitoring, fault diagnosis and predictive maintenance providing a platform for effective asset management.

Otter USV

Otter is an Unmanned Surface Vehicle from Maritime Robotics, Norway. It is a robust vehicle mainly used in marine bathymetric surveys and sea-bed mapping. It runs on electric propulsion with a total installed power of 1830 Wh from two Torqeedo™ batteries and driven by two Torqeedo Ultralight 403 AC trolling motors(or thrusters) which provide a static thrust of 15 kg each approximately. Additionally, it is equipped with a GNSS sensor for navigation and a camera for capturing video feed. Otter is powered by a Raspberry-Pi as its main computer, also referred to as OBS (On-board System) and an additional Intel-based payload computer running Windows 10 OS for integrating optional payload sensors. There are four channels of communication with Otter namely - VHF Radio, 4G, WiFi and

Subsystem	Components
Power	Torqeedo 915Wh batteries × 2
Actuators	403 AC Trolling Motors × 2
Sensors	GNSS Video Camera
Computer	Raspberry-Pi (Main) Intel Mini PC(Secondary)
Communication	VHF-Radio 4G WiFi Ethernet

Table 1.1: Otter System Description

Ethernet.This gives a wide range of choice to the users to select a preferred communication channel based on application requirements and range limitations.In total, Otter weighs 55 Kg and has an overall size of $0.2m \times 0.1m \times 0.08m$ in length, breadth and height dimensions respectively. Table. 1.1 lists the components of Otter and maps them to the subsystems in Otter. This system description is important to understand the capabilities and limitations of the vehicle before we proceed to developing a Digital Twin of the vehicle.

1.2 Motivation

1.3 Literature Review

In order to better realise the potential of a digital twin platform for Otter, a literature review is carried out.The search is carried out in two directions. On one side, a search is conducted on Digital Twins and their application areas in other similar systems. On the other side, a parallel search is conducted on Unmanned Surface Vehicles to the identify the potential for a digital twin model. The search results are discussed below.

Digital Twins are an emerging trend in technology and find applications in many engineering domains like aviation, health care, manufacturing etc., [6]. Additionally, their use in different life-cycle phases of the asset such as design, development or manufacturing, service and disposal is identified [7]. Since, any asset is intended to spend the longest amount of time and generates most value in its operational phase, we restrict our scope of digital twin usage to this phase.

Figure 1.1: Otter USV

In the service or operational phase, digital twins can be utilised for the following purposes [7]:

- Predictive maintenance
- Condition Monitoring and Fault diagnosis
- State monitoring
- Virtual testing

Since, the focus of the current thesis is mobile systems similar to USVs, the state-of-the-art of digital twins in these areas is studied. Major.P.Y et al.(2021) created a real-time digital twin of research vessel RV Gunnerus and used the twin for remote monitoring of ship and on-board crane system [8] within a digital twin framework. Alexander Danielsen-Haces, in his master's thesis [9], documents the development of digital twin platform for an autonomous ship model. A fault detection feature for ship thruster faults is implemented using machine learning along with the necessary digital twin infrastructure. Grigoropoulos.N and Lalis.S (2020) present a simulation-cum-digital twin environment for managing multiple quadcopter drones using PAAS cloud infrastructure [**droneDT**].Moghadam, F.K, Rebouças G.F.S and Nejad.A.R [**turbineDT**] developed a digital-twin model for estimating the remaining lifetime of wind turbine drive trains.Most of the literature on digital twins is focused Predictive Maintenance and Condition Monitoring applications. Predictive Maintenance is more relevant in high-value assets in continuous operation where down-time leads to significant costs. In the context of USV's, given the complex nature of environment in which they operate with minimum human supervision, Condition Monitoring is deemed to be more useful.Zhang.Q (2018) propose an Adaptive Kalman Filter for actuator fault diagnosis in Linear Time Variant/Linear Parameter Varying(LTV/LPV) systems [10]. Extending this work to non-linear systems, Skiver.M, Helck.J and Hasan.A (2019) used an Adaptive Extended Kalman Filter for actuator fault diagnosis and tested the method in case of autonomous car and gantry crane [11] and commented on the stability of the algorithm.Alessandri.A, Caccia.M and Verrugio.G (1999)applied a fault diagnostic system based on a bank of EKF fault estimators in case of an Unmanned Underwater Vehicle [12].Ko.N.Y et al.(2021) used a two stage EKF filter to estimate sensor and actuator faults in which real experimental data is used for IMU sensor faults while simulated data is used for thruster faults due to difficulties in introducing thruster faults in experiments [13]. They achieve satisfactory results for sensor faults estimation however notice a time delay in thruster fault estimation. Zhou.Z, Zhong.M and Wang.Y(2019) used a fault and state observer in case of Unmanned Surface Vehicles in network environments. They also implement a fault-tolerant control system based on the observer estimates [14].In contrast to the above mentioned, model-based fault diagnosis techniques, Abed.W, Sharma.S and Sutton.R (2015)implemented a Neural Network for fault diagnosis of USV trolling motors trained on the data from the stator current and motor vibrations[15]. Another interesting application of digital twins in Unmanned Surface Vehicles is the System Identification of hydrodynamic parameters using Machine Learning [16] and Model-based methods[17]. This application reduces,if not completely, eliminates the need for CFD simulation which are time and resource consuming.

Interestingly, Kapteyn.M.G, Pretorius J.V.R and Willcox.K.E (2021) propose a mathematical framework for developing digital twins at scale [18]. The framework is based on 'Probabilistic Graphical Models' which inherently support representation and inference making them best suitable for digital twin applications. It is one of the few works, which attempts to define a mathematical model for digital twins.

Based on the above literature search, Fault Diagnosis of Otter's actuators is selected as a demonstrative feature of the current digital twin application. The availability of existing literature in this area and the practical usefulness of this feature compared to others are the main reasons behind this choice.

1.4 Problem formulation

With a backdrop of a massive rise of Digital Twins with several industrial applications, this thesis aims to develop a digital twin for an Unmanned Surface Vehicle, Otter. In view of this, two problems are identified

1. Given the wide scope of Digital Twins and an ambiguity in their definition, implementing digital twins in real world applications can be fraught with challenges unique to each system with no uniform mathematical model. This thesis explores the possibility of a generalised mathematical framework than can be adjusted to a wide gamut of digital twin applications.
2. Though various condition monitoring techniques like fault diagnosis exist for Unmanned Surface Vehicles and other autonomous systems, not many are developed for integration within a digital twin framework.Therefore,this thesis deploys the digital twin to perform 'Actuator fault diagnosis' on Otter's thrusters and demonstrate the power of digital twins in addressing real world problems in cyber-physical systems.

1.4.1 Objectives

Based on the problem formulated above, the following concrete research objectives are generated which will be systematically addressed in this thesis

- Constructing a unifying mathematical model for digital twins development at scale.
- Utilizing the fore-mentioned mathematical framework and performing actuator fault diagnosis on Otter's actuators.
- Making meaningful predictions on the Otter's behaviour based on the updated digital twin that is cognizant of actuator faults.

1.5 Contributions

In the process of achieving the objectives laid out in the previous section, I relied on existing work mentioned in the Literature review section(1.3) which also is a source of inspiration for me. Even though, the thesis is an extension or re-adaption of already existing work, there are some unique contributions due to the way some of the methods are implemented to suit real-time needs of digital twins. These contributions are summarised below.

- Firstly, the thesis uses Probabilistic Graphical Models as a mathematical framework for the digital twin. The same is proposed by Kapteyn.M.G et al [18] in the case of structural health monitoring for drones. In this thesis, I extend the idea to Fault Diagnosis of Unmanned Surface Vehicles, validating the general nature of the framework and lending more credibility to it. The framework can be further used in creating other digital twin features in the context of Unmanned Surface Vehicles.
- Secondly, and most importantly,the thesis validates the use of Adaptive Kalman Filter for actuator fault diagnosis experimentally. The method, though proven to be robust, has only been applied in simulated and fictive problems so far. By implementing the method on experimental data, the thesis brings forth practical issues with respect to the application of AEKF to real data and solves them. In doing so, a low-cost, easy to deploy solution is contrived which solely uses GNSS sensor data in comparison to the more expensive data-driven methods which warrant additional sensors and more training data. This can be considered as the novel contribution of this thesis as it has not been found in the collected literature to my knowledge.

1.6 Thesis structure

This section describes the organisation of this thesis into different chapters and highlights the key points discussed in each chapter.

Chapter 1 gives a brief introduction to the broad concepts used in this thesis - Digital Twins and Unmanned Surface Vehicles(USVs), and gives a system description of Otter, the USV used in this thesis. It also lays out the state-of-the-art of digital twins applications in autonomous systems similar to Otter and identifies potential gaps for further research. Based on the above literature search, we formulate the research objectives for the current work and present the proposed methodology. Finally, the contribution of this thesis is discussed putting it in context with the existing body of work.

Chapter 2 delves into the theoretical background of the methods used in the thesis. It details two major works on which the proposed methodology is based, namely - *Probabilistic Graphical Models* as a general framework for the predictive digital twins [18] and *Adaptive Extended Kalman Filter* based actuator fault diagnosis as a demonstrative application of the digital twin [10] [11].

Chapter 3 is related to the implementation aspects of the proposed methods and considers both the simulated and physical models. It briefly discusses the practical issues in data acquisition, processing, filtering and obtaining meaningful results for fault diagnosis of Otter's actuators.

Chapter 4 presents the results obtained from the simulated and real experiments using the methods discussed in **Chapter 3**. After each experiment, the results are discussed to aid in their interpretation. The final results of actuator fault diagnosis is presented in the end of the chapter. The results are promising in the way they validate the used methods and indicate that a further research in this direction is reasonable.

Finally, **Chapter 5** concludes the thesis by providing a summary of the overall results of this work and their usefulness in the context of digital twins for Unmanned Surface Vehicles and other autonomous systems.It also provides a set of recommendations for future work to improve the existing method but also to use the results generated from the current work as bench-mark for other state-of-the-art methods.

Chapter 2

Theoretical Background

This section introduces the reader to the theoretical and mathematical foundation behind the digital twin framework and also to the demonstrative application 'Actuator Fault Diagnosis' using this framework. Much of the theory presented in this section forms the skeleton on which a Digital Twin application for Otter is built but is also flexible enough to be re-adapted to digital twin applications in other cyber-physical systems.

2.1 Systems and Models:

In systems engineering, a system is defined as an arrangement of parts or elements that together exhibit behaviour that individual constituents do not [19]. Systems can be physical or conceptual with varying layers of constituent subsystems and complexity that arises from the time-dependent interaction between these subsystems. For example, a spring-mass-damper can be viewed as a simple system while a satellite or an industrial plant is a relatively complex dynamic system. However, even simple systems can become complex when considered in a finer level of detail. We need a simplified abstraction of any system in order to make meaningful use of it. Models are simplified representations of a system and its components at some particular point in time or space intended to promote understanding of the real system. As abstractions of a system, they offers insights about one or more of the system's aspects, such as its function, structure, properties, performance, behavior, or cost. Models can be classified into different types based on features of the system they represent. For example, *Physical Models* represent the physical aspects of the system and are used in experimental testing of the system. *Descriptive Models* describe the logical relationships between the sys-

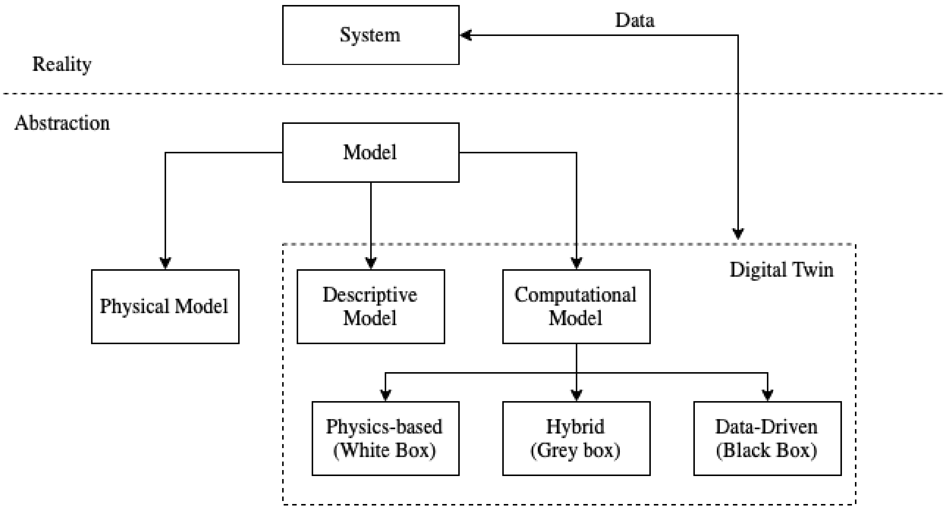


Figure 2.1: System Models and Digital Twins

tem constituents, their hierarchical structure, functions of the components and the three dimensional geometric representation of the system. *Analytical or Computational Models* describe mathematical relationships, such as differential equations, that support quantifiable analysis about system parameters. They help us understand, design, test and predict performance of complex systems. There are a wide range of Computational Modelling techniques depending on the nature of the system and modelling assumptions like Non-deterministic, Deterministic, Static, Dynamic, Discrete, Continuous, Stochastic (also called probabilistic or statistical), Agent-based and Data-driven models [20]. The different types of modelling methods can be grouped into three major categories i.e., (i) Physics-based Models (ii) Data-Driven Models and (iii) Hybrid Models [1][21]. The above concepts are illustrated in Figure.2.1

A Digital Twin is defined as a set of virtual information constructs that mimics the structure, context and behaviour of an individual/unique physical asset, or a group of physical assets, dynamically updated with data from its physical twin throughout its life cycle. Though Digital Twins appear similar to models of a system, there are three key differences [22] [23]

- The system which the digital twin is intended for should be a physical entity generating real data while model can represent abstract entities also
- Digital Twins encompass the model/models of a system
- Connected knowledge transfer to dynamically update the used models which may not be present in system models.

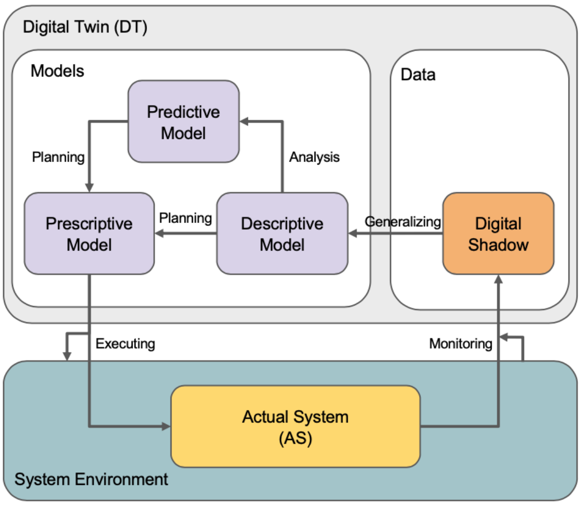


Figure 2.2: Digital Twin Concept [24]

Therefore, the computational models in digital twins evolve over time to persistently represent the structure, behaviour and context of a unique physical asset [18]. The descriptive models are also updated to represent the present state of the system. In case of physical entities with geometrical features, 3D geometric models can be viewed as descriptive models.

From the several types of computational models discussed above, which ones are most suitable for Digital Twin applications? For answering this question, three kinds of modelling approaches mentioned above are discussed briefly [1]:

- **Physics based modelling (White-box):** In this approach, the system is described using mathematical equations derived from underlying physical laws of the system. These equations are further developed and validated using experiments and high-fidelity numerical simulations. While they offer an accurate system description in the short-term, they suffer from long-term data assimilation. They are also sensitive to numerical instability. Moreover, the models may not capture the whole physics resulting in under representation. However, the major advantage of physics based models is their lower susceptibility to bias, easier interpretability and better generalization capability to new problems within similar physical domains. While physics based models can be used in the earlier stages of system design and analysis, the lack of long-term model fidelity is a major drawback and hence render them unsuitable for Digital Twin applications which are required to support the physical asset throughout its life-cycle.

- **Data-driven modelling (Black-box:** As opposed to physics based models described above, data-driven models rely solely on data acquired from the system fed into black-box models based on Artificial Intelligence and Machine Learning methods. Unlike physics based models, data-driven models do not suffer from long-term data fit and numerical stability issues. However, the most pressing issue is their black-box nature which makes them almost impossible to interpret. Also, they cannot be generalized easily into unseen problems and the bias in data gets transferred to the bias in prediction.
- **Hybrid modelling (Grey-box:** This is a recent modelling method which is intended to remove the shortfalls of both the above approaches. In this approach physics-based models or data-driven models or a combination of both are used in connection with the data acquired from the system. When physics-based models are used alongside data, it is referred to as physics-based surrogate models or data assimilation techniques. When data-driven models are used in combination with physics-based models, it is referred to as physics-informed data-driven modelling. Therefore, hybrid models are a general approach utilizing the strengths of both the above discussed modelling methods making them suitable digital twin models for a wider class of systems.However, one challenge with grey-box models is the sophisticated mathematics involved in merging real-time date with physics and the associated computational time [25].

Fig.2.3 compares the three main modelling approaches

To summarize the above discussion, models encode the knowledge of a system. Digital Twins are a digital representation of system models dynamically updated with data acquired from the system throughout its life. Hybrid modelling techniques which combine physics-based and data-driven methods while utilizing the data from the real system are better suitable in a wide range of digital twin applications. Probabilistic Graphical Models, particularly Dynamic Bayesian Networks, are one such class of hybrid modelling methods which are found to adequately address the needs of digital twin models at scale[18]. Hence, the same will be discussed below at a greater detail and their implementation in the present work will be discussed in subsequent chapters.

Features	Whitebox	Greybox	Blackbox
Suitable for extremely complicated modeling tasks	✗	✗	✓
Online retuning possible using measurement data	✗	✓	✓
Model retuning is computationally inexpensive	N/A	✓/✗*	✗✗
Model retuning is possible with a small dataset	N/A	✓	✗
Zero risk of overfitting	✓	✓	✗
Zero risk of unexpected model behavior	✓	✗	✗✗
Model equations explicitly defined by user	✓	✓	✗

Figure 2.3: Feature comparison of different modelling methods [21]

2.2 Probabilistic Graphical Models:

Continuing from earlier discussion, it is reasonable to state that systems that warrant Digital Twin representations usually also contain inherent *complexity* and *uncertainty*.If the system is simple and deterministic, a dynamic model of the system will be able to represent it at all times. However, that is not the case with most real-world systems. At least, the ones that are the subject of this work.

System Complexity: Complexity in systems arises from multiple factors. For engineering systems, it can arise from *Structural Complexity* due to a large number of interacting subsystems, part and components and their relationships. Complexity can also arise from system dynamics due to the time varying nature of system states and parameters. This is called *Dynamic Complexity* [19].

System Uncertainty: Uncertainty, like complexity, is inherent in most real-world systems models both because of abstractions and assumptions made while modelling the system and also because of the noisy observations that are used to update these models [26].

Owing to the dual challenge posed by *uncertain complex* systems, a robust hybrid modelling framework is required which is able to incorporate physics-based and/or data-driven methods along with the knowledge transferred from the real system via data while handling the system uncertainty and complexity. Probabilistic Graphical Models use a graph-based representation as a basis for compactly encoding a complex distribution over a high-dimensional space. They combine graph theory with probability theory and are shown to facilitate *Representation*, *Inference* and *Learning* [26] which are the corner stones of complex system modelling. *Representation* is the encoding of knowledge about the system in an machine-readable format, *Inference* is the ability to use the existing representation to perform meaningful analysis of the system and *Learning* is the ability to use past experience and data to update the existing representation.Many of the classical multivariate probabilistic systems like mixture models, factor analysis, hidden Markov models, Kalman filters and Ising models, are special cases of the general graphical model formalism. The graphical model framework provides a way to view all of these systems as instances of a common underlying formalism. [27]

Probabilistic graphical models are graphs in which nodes represent random variables, and the edges represent conditional independence assumptions. Hence they provide a compact representation of joint probability distributions. [27].

There are two main kinds of graphical models: undirected and directed. Undirected graphical models are also known as Markov networks or Markov random fields (MRFs) and directed graphical models are also known as Bayesian networks (BNs), belief networks, generative models, causal models, etc. [27].

2.2.1 Dynamic Bayesian Networks:

As mentioned above, Bayesian Networks are a class of Probabilistic Graphical Models where the graph network is directed. The nodes in the graph represent random variables and the edges represent their conditional dependence. The directed nature of the graph edges indicate the causal relationship between the nodes. Kalman filters can be viewed as dynamic Bayesian Networks with continuous variables where probability dependencies are linear Gaussian.

2.2.2 Probabilistic Graphical Models and Digital Twins:

This graphical model represents the structure in an asset–twin system by encoding the interaction and evolution of the digital twin and asset variables. In particular, the model encodes the end-to-end digital twin data-to-decisions flow, from sensing through inference and assimilation to action. The graphical model formalism provides a firm foundation on which models from different domains like control theory,artifical intelligence, decision theory can be integrated to solve complex tasks such as data assimilation, state estimation, prediction, planning and learning, all of which are crucial to realizing the potential of digital twins.[18]. Importantly, the proposed graphical model framework does not restrict the nature of the models comprising the digital twin. These models could be physics-based, data-driven and rule-based.This versatile nature of the graphical models is one of their most important properties especially in the context of digital twin applications.Fig.2.4 is an example graphical model used in [18] for the purpose of structural health monitoring and optimal control of a fleet of drones. At any time t, nodes lettered S_t represent the state of the physical asset, nodes O_t are the observations from the asset flowing into the digital space, nodes D_t are the digital states updated by means of system models and assimilated data. Q_t are the quantities of interest which can be either the hidden system states or other parameters. R_t is the computed reward or cost to enable optimisation techniques. Thus it can be seen that the graph is dynamically updated using a combination of measuring data and system models satisfying the essential requirement for digital twins. In the subsequent sections and chapters, we will see how this framework can be adapted in case of Otter for the purpose of Actuator Fault Diagnosis.

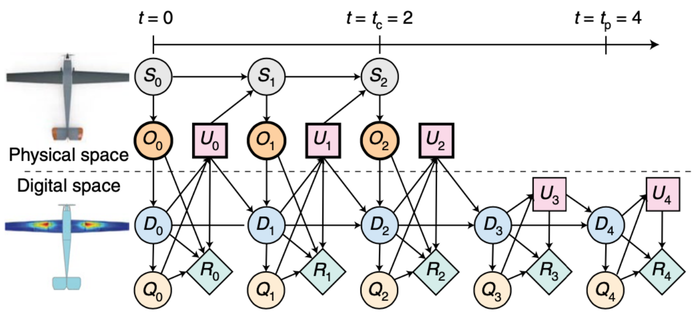


Figure 2.4: Probabilistic Graphical Model example for Digital Twins[18]

2.3 Fault Diagnosis

Since we decided to use Actuator Fault Diagnosis as a demonstrative application within the graphical model framework,the concepts, definition and techniques underlying the subject of 'Fault Diagnosis' are detailed in this section.

2.3.1 Faults

Before we delve deeper into the subject, it is important to cover the commonly used terminology for the sake of clarity. The below definitions are from SAFE-PROCESS Technical Committee [28]

Fault: A fault is defined as the deviation in at least one property or feature of the system from the usual condition

Failure: A failure is the permanent interruption in the system's ability to perform a required function. Failure usually results from faults.

Malfunction: Intermittent irregularities in meeting the system's desired objectives are called Malfunctions. Malfunctions also result from system fault.

In this work, we mainly focus on 'Faults'. Faults in process equipment or instrumentation, or within the process itself, can result in off-specification production, increased operating costs, shut-down, and the possibility of detrimental environmental impact. Furthermore, prompt detection and diagnosis of process malfunctions are strategically important due to the economic and environmental demands required for companies to remain competitive in world markets.[29]. Given the sophisticated nature of today's systems, manual supervision of faults in all the system components is not a feasible solution. Hence, there is an increasing move towards automatic supervision of system abnormalities.

The automatic supervision in the past was mostly realized by limit checking (or threshold checking) of some important process variables, like, e.g. force, speed, pressure, liquid level, temperatures. Usually alarms are raised if limit values are exceeded and operators have to act or protection systems act automatically. This is in many cases sufficient to prevent larger failures or damages. However, faults are detected rather lately and a detailed fault diagnosis is mostly not possible with this simple method. Methods of modern systems theory show the systematic use of mathematical process and signal models, identification and estimation methods and methods of computational intelligence can accelerate the fault diagnosis process and make it more reliable[30]. Below, the varying levels of fault handling methods are defined [28] and the same are illustrated in Fig. 2.5 for easier interpretation.

Fault Detection: Determination of faults present in a system and the time of detection.

Fault Isolation: Determination of the kind, location and time of detection of

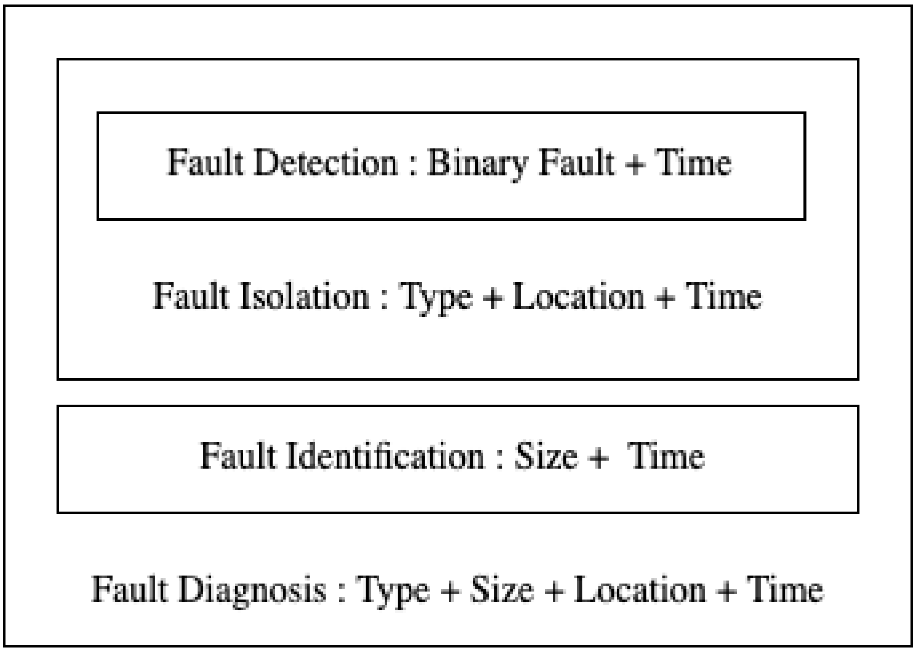


Figure 2.5: Hierarchy of fault handling techniques

fault of a fault. It usually follows fault detection.

Fault Identification: Determination of the size and time-varying behaviour of a fault following fault isolation.

Fault Diagnosis: Determination of the kind, size, location, and time of detection of a fault. It follows fault detection and includes both fault detection, isolation and identification.

2.3.2 Fault detection methods

Once a fault occurs in a system, it changes the behaviour of the system. The first step in reacting to this fault is to simply detect its presence. Different methods are used for fault detect depending on the nature of the fault and the type of system. These methods are can be grouped into three main categories as discussed below and illustrated in Fig.2.6

Model-based: Model-based methods of fault detection use the relations between several measured variables to extract information on possible changes caused by faults. The relation between the input signals **U** and the output signal **Y** are represented by a mathematical model of the system.Fault detection methods then extract special features, like parameters θ, state variables $\mathbf{x}$ or residuals $\mathbf{R}$ [30]

Signal-based: Signal based methods utilize measured signals rather than ex-

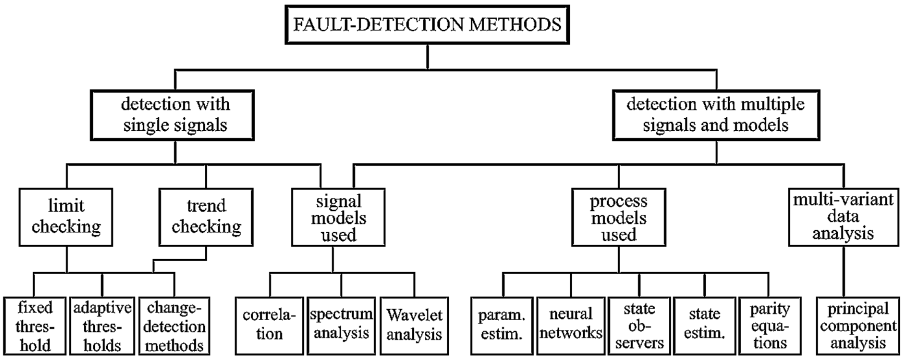


Figure 2.6: Fault Detection Methods [30]

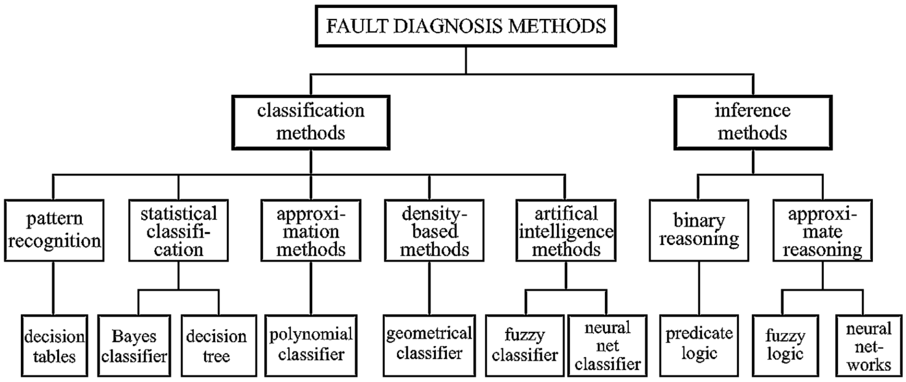


Figure 2.7: Fault Diagnosis Methods [30]

plicit input-output models. The faults in the process are reflected in the measured signal, whose features are extracted. The feature signals can be either in time domain (mean,trends, standard deviation, slope and magnitudes such as peak and RMS) or frequence domain (eg., spectrum).

2.3.3 Actuator Fault Diagnosis using Adaptive Kalman Filter:

Adaptive Kalman Filters have been used for actuator fault diagnosis for LTV/LPV systems [10] in a stochastic framework with rigorously proved stability and minimum variance properties.

The discrete time LTV system subject to actuator faults can be written in the form:

$$x(t) = A(t)x(t-1) + B(t)u(t) + \Psi(t)\theta + w(t) \tag{2.1}$$

$$y(t) = C(t)x(t) + v(t) \tag{2.2}$$

where t = 0,1,2,... is the discrete time instant index, $x(t) \in \mathbb{R}^{\ltimes}$ is the state, $u(t) \in \mathbb{R}^{\lessdot}$ the input, $y(t) \in \mathbb{R}^{\gtrdot}$ the output, $A(t), B(t), C(t)$ are time-varying matrices of appropriate sizes characterizing the nominal state-space model, $w(t) \in \mathbb{R}^{\ltimes}, v(t) \in \mathbb{R}^{\gtrdot}$ are mutually independent centered white Gaussian noises of covariance matrices $Q(t) \in \mathbb{R}^{\ltimes \times \ltimes}$ and $R(t) \in \mathbb{R}^{\gtrdot \times \gtrdot}$, and the term $\Psi(t)\theta$ represents actuator faults with a known matrix sequence $\Psi(t) \in \mathbb{R}^{\ltimes \times \shortmid}$ and a constant vector (or piece wise constant with rare jumps) parameter vector $\theta \in \mathbb{R}^{\shortmid}$.

A typical example of actuator faults represented by the term $\Psi(t)\theta$ is the actuator gain loses. When affected by such faults, the nominal control term $B(t)u(t)$ becomes

$$B(t)(I_l - diag(\theta))u(t) = B(t)u(t) - B(t)diag(u(t))\theta$$

where I_l is the $l \times l$ identity matrix, the diagonal matrix $diag(\theta)$ contains gain loss coefficients within the interval $[0, 1]$, and $\Psi(t) \in \mathbb{R}^{\ltimes \times \lessdot}(p = l)$ is, in this particular case,

$$\Psi(t) = -B(t)diag(u(t)) \tag{2.3}$$

The problem of actuator fault diagnosis is to characterize actuator parameter changes from the input output data sequences $u(t), y(t)$,and the matrices $A(t)$, $B(t), C(t), Q(t), R(t), \Psi(t)$. The characterisation of actuator parameter changes is based on joint estimation of states and parameters. The difference between the nominal value of the parameter vector θ and its recursively computed estimate can be viewed as a residual vector, and its evaluation can be simply based on thresholds or more sophisticated decision mechanisms. Residual generation and residual evaluation jointly form the basis of classic fault diagnosis model-based procedures [30].

Rewriting the augmented system from 2.1 and 2.2 in matrix form:

$$\begin{bmatrix} x(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} A(t) & \Psi(t) \\ 0 & I_p \end{bmatrix} \begin{bmatrix} x(t-1) \\ \theta(t-1) \end{bmatrix} + \begin{bmatrix} B(t) \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} w(t) \\ 0 \end{bmatrix}$$

$$y(t) = \begin{bmatrix} C(t) & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \theta(t) \end{bmatrix} + v(t)$$

The above system is assumed to be completely observable and completely controllable. These assumptions along with the persistent excitation condition ensure the stability of the Kalman filter used for fault diagnosis [10]

For non-linear systems, we cannot directly use the AKF algorithm which is meant for linear systems.First, We need to linearise the system around the state estimates using the Jacobian of the system matrix. This version of AKF for non-linear system is termed as Adaptive Extended Kalman Filter(AEKF) and was successfully applied in case of actuator fault diagnosis like the linear version [11] with the only change being the replacement of $A(k)$ with the Jacobian $F(k)$ in equation 2.8 to 2.19 as given below:

The system from equation can written in discrete form as

$$x(k+1) = f(x(k)) + B(k)u(k) + \Psi(k)\theta + w(k) \tag{2.4}$$

$$y(k) = C(k)x(k) + v(k) \tag{2.5}$$

The Adaptive Extended Kalman Filter

In the adaptive extended Kalman filter, the state estimate $\hat{x}(t|t) \in \mathbb{R}^{\ltimes}$ and the parameter estimate $\hat{\theta}(t) \in \mathbb{R}^{\shortmid}$ are recursively updated at every estimate instant t. This algorithm involves other recursively updated auxiliary variables: $P(t|t) \in \mathbb{R}^{\ltimes\times\ltimes}, \Upsilon \in \ltimes\times\shortmid, S(t) \in \shortmid\times\shortmid$ and a forgetting factor $\lambda \in (0,1)$. The initial conditions can be assumed to be $x(0) \sim \mathcal{N}(x_0, P_0), \theta_0 \in \mathbb{R}^{\shortmid}, \lambda \in (0,1)$ and ω be a chosen positive value for initializing $S(t)$.The adaptive Kalman filter consists of the initialization and recursion steps as described below:

Initialization

$$P(0|0) = P_0 \quad \Upsilon(0) = 0 \quad S(0) = \omega I_p \tag{2.6}$$

$$\theta(\hat{0}) = \theta_0 \quad \theta(0|\hat{0}) = x_0 \tag{2.7}$$

Recursions for t=1,2,3..

$$P(t|t-1) = F(x(\hat{t}))P(t-1|t-1)F(x(\hat{t}))^T(t) + Q(t) \tag{2.8}$$

where the Jacobian of $f(x(k)$ matrix is given by $F(x(k)) = \partial f(x(k))/\partial x(k)|$

$$\Sigma(t) = C(t)P(t|t-1)C^T(t) + R(t) \tag{2.9}$$

$$K(t) = P(t|t-1)C^T(t)\Sigma^{-1}(t) \tag{2.10}$$

$$P(t|t) = [I_n - K(t)C(t)]P(t|t-1) \tag{2.11}$$

$$\Upsilon(t) = [I_n - K(t)C(t)]F(x(\hat{t}))\Upsilon(t-1) + [I_n - K(t)C(t)]\Psi(t) \tag{2.12}$$

$$\Omega(t) = C(t)F(x(\hat{t}))\Upsilon(t-1) + C(t)\Psi(t) \tag{2.13}$$

$$\Lambda(t) = [\lambda\Sigma(t) + \Omega(t)S(t-1)\Omega^T(t)]^{-1} \tag{2.14}$$

$$\Gamma(t) = S(t-1)\Omega^T\Lambda(t) \tag{2.15}$$

$$S(t) = \frac{1}{\lambda}S(t-1) - \frac{1}{\lambda}S(t-1)\Omega^T(t)\Lambda(t)\Omega(t)S(t-1) \tag{2.16}$$

$$\tilde{y}(t) = y(t) - C(t)[f(\hat{x}(t-1|t-1)) + B(t)u(t)\Psi(t)\hat{\theta}(t-1)] \tag{2.17}$$

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \Gamma(t)\tilde{y}(t) \tag{2.18}$$

$$\begin{aligned} \hat{x}(t|t) &= f(\hat{x}(t-1|t-1)) + B(t)u(t) \\ &+\Psi(t)th\hat{e}ta(t-1) + K(t)\tilde{y}(t) \\ &+\Upsilon(t)[\hat{\theta}(t) - \hat{\theta}(t-1)] \end{aligned} \tag{2.19}$$

Recursions 2.6 to 2.9 compute the covariance matrix $P(t|t) \in \mathbb{R}^{\kappa\times\kappa}$ of the state estimate, the innovation covariance matrix $\Sigma(t) \in \gtrdot \times \gtrdot$ and the state estimation gain matrix $K(t) \in \ltimes \times \gtrdot$. These equations are identical to those of the classical Kalman Filter. Equations 2.10 to 2.14 compute the parameter estimation gain matrix $\Gamma(t) \in \mathbb{R}^{\imath\times\gtrdot}$ through the auxiliary variables $\Upsilon(t) \in \mathbb{R}^{\ltimes\times\imath}, \Omega(t) \in \mathbb{R}^{\gtrdot\times\imath}, S(t) \in \mathbb{R}^{\imath\times\imath}$. Equation 2.15 computes the innovation $\tilde{y} \in \mathbb{R}^{\gtrdot}$. Finally, recursions 2.16 and 2.17 compute the state estimate and the fault parameter estimate.

Part of the equation 2.17,

$$\hat{x}(t|t) = \hat{x}(t|t) = f(\hat{x}(t-1|t-1)) + B(t)u(t) + K(t)\tilde{y}(t)$$

represents the classical extended Kalman filter with the traditional *prediction* and *update* combined into a single step. The term $\Psi(t)\hat{\theta}(t-1)$ corresponds to the actuator fault term $\Psi(t)\theta$ in 2.1 with θ replaced with $\hat{\theta}(t-1)$. The term $\Upsilon[\hat{\theta}(t)-\hat{\theta}(t-1)$ is for the purpose of compensating the error caused by $\hat{\theta}(t-1) \neq \theta$.

The stability of the filter and the bounded nature of the recursively computed matrices $P(t), \Upsilon(t), S(t), K(t)$ and $\Gamma(t)$are rigorously proven in [10] and [11]

2.4 USV State-Space model:

As mentioned earlier, hybrid modelling is best suitable for the digital twin which updates the current state of the system using a physics-based model and state measurements from the real asset. In order to propagate the digital state of the system in time, a dynamic model of Otter is required. Generally, if the USV is treated as rigid body, the dynamic model can be divided into two parts: Kinematics, which treats the geometrical aspects of motion and Kinetics, which is the analysis of the forces causing the motion [31].

2.4.1 Kinematics:

Ocean vehicles undergo a 6-DOF motion in space relative to a chosen coordinate system which is usually earth-fixed in case of low-speed vehicles. These degrees of freedom are termed as surve, sway and heave for translational motion pitch, roll and yaw for rotational motion. In case of surface vehicles,especially when dealing with actuator effects, the effects of roll, pitch, and heave can be ignored and the vehicle model can be simplified to a planar motion with only 3 degrees of freedom as shown in Fig. 2.8

Therefore, a typical USV model in planar motion can be expressed as:

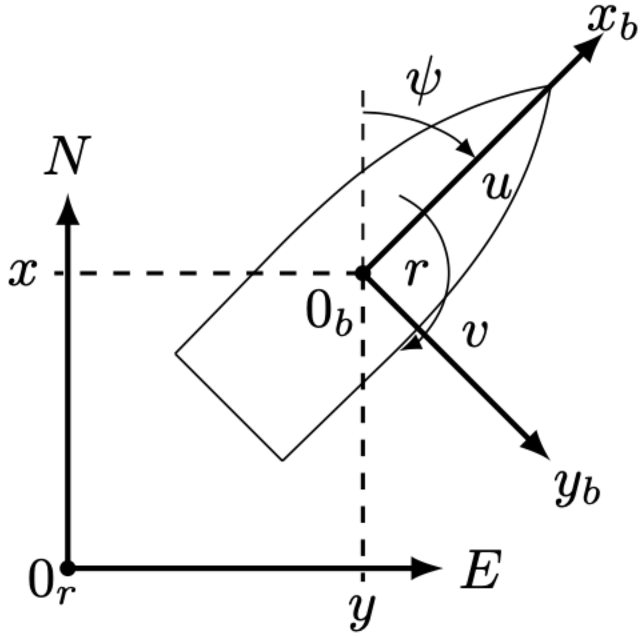


Figure 2.8: Schematic diagram of a 3-DOF surface vehicle

$$\dot{\eta} = J(\eta)\nu \tag{2.20}$$

where $\eta = [x, y, \psi]^T$ is the position (x, y) and yaw angle ψ of the USV in the earth-fixed frame, $\dot{\eta} = [\dot{x}, \dot{y}, \dot{\psi}]^T$ describes the vehicle North, East and Z-axis angular velocities in the earth-fixed frame in that order, $\nu = [u, v, r]^T$ is the vehicle surge velocity(u), sway velocity(v), and yaw rate(r) in the body-fixed frame and the transformation matrix $J(\eta)$, as a function of yaw, is given by:

$$J(\eta) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.21}$$

From equations 2.20 and 2.21, we can write:

$$\dot{x} = u\cos\psi - v\sin\psi \tag{2.22}$$

$$\dot{y} = u\sin\psi + v\cos\psi \tag{2.23}$$

$$\dot{\psi} = r \tag{2.24}$$

As the motion is only in the surge direction, the sway velocity can be ignored. Additionally, the time-derivative of surge velocity is considered, resulting in below equations:

$$\dot{x} = U\cos\psi \tag{2.25}$$

$$\dot{y} = U\sin\psi \tag{2.26}$$

$$\dot{U} = a \tag{2.27}$$

$$\dot{\psi} = r \tag{2.28}$$

where U, a denotes the surge velocity and surge acceleration respectively.

Equations 2.25 to 2.28 represent the kinematics of an Unmanned Surface Vehicle moving in calm waters and are referred to as the Kinematic model of the USV system.

In state-space representation, the above equations [2.25 to 2.28] can be written as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{U} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} U\cos\psi \\ U\sin\psi \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ r \end{bmatrix} \tag{2.29}$$

Equation 2.29 can be written in a compact state-space notation as:

$$\mathbf{x} = f(x)\mathbf{x} + B\mathbf{u} \tag{2.30}$$

where $\mathbf{x} = [x, y, U, \psi]^T$ is the state vector, f(x) is a non-linear function of the state vector, B is the control matrix and $\mathbf{u} = [a, r]^T$ is the control input.

2.4.2 Kinetics:

While the kinematic equations describe the motion of the system, they do not provide any information on the forces causing the motion and their relation to the state variables.

From Newton's laws, the acceleration of a rigid body under the influence of external forces is given by:

$$M_{RB}\,\dot{\nu} + C_{RB}\,\nu = \tau_{RB} \tag{2.31}$$

where $\nu = [u, v, w, p, q, r]$ is the body-fixed linear and angular velocity vector, M_{RB} is the rigid-body inertia matrix,$C_{RB}(\nu)$ is the rigid-body Coriolis and centripetal matrix, and τ_{RB} is the sum total of external forces and moments on the rigid-body. Considering the motion of the vehicle in a marine environment, τ_{RB} can be further decomposed into radiation-induced, environmental and propulsion forces.

$$\tau_{RB} = \tau_H + \tau_E + \tau \tag{2.32}$$

Furthermore, τ_H denotes the net effect of radiation-induced forces which includes added inertia, hydrodynamic damping and restoring forces which are nonlinear in nature and highly depend on the vehicle geometry. Added inertia is due to the inertia of the surrounding fluid. Hydrodynamic damping $D(\nu)$ is combination of fluid damping forces like potential damping $D_P(\nu)$, skin friction $D_S(\nu)$, wave drift damping $D_W(\nu)$, and damping due to vortex shedding $D_M(\nu)$. Restoring forces are the effect of gravity and buoyancy $g(\eta)$. All the above components are accounted in equation 2.33 and

$$D(\nu) = D_P(\nu) + D_S(\nu) + D_W(\nu) + D_M(\nu) \tag{2.33}$$

$$\tau_H = -M_A\,\dot{\nu} - C_A\,\nu - D(\nu)\,\nu - g(\eta) \tag{2.34}$$

From equations 2.31 and 2.32, the final dynamics equation of a rigid body in a fluid medium is derived as:

$$M\,\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau_E + \tau \tag{2.35}$$

where $M = M_{RB} + M_A$ and $C = C_{RB}(\nu) + C_A(\nu)$

For a surface vehicle, the terms of the above equation are given using the below relations:

$$M = \begin{bmatrix} m - X_{\dot{u}} & 0 & -my_g \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ -my_g & mx_g - N_{\dot{v}} & I_z - N_{\dot{r}} \end{bmatrix} \tag{2.36}$$

$$C(\nu) = \begin{bmatrix} 0 & 0 & -m(x_g r + v) + Y_{\dot{v}} v + \frac{Y_{\dot{r}} + N_{\dot{v}}}{2} r \\ 0 & 0 & (m - X_{\dot{u}})u \\ m(x_g r + v) - Y_{\dot{v}} v - \frac{Y_{\dot{r}} + N_{\dot{v}}}{2} r & -(m - X_{\dot{u}})u & 0 \end{bmatrix} \tag{2.37}$$

$$\begin{aligned} D(\nu) &= D + D_n(\nu) \\ &= -\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix} - \begin{bmatrix} X_{u|u|}|u| & 0 & 0 \\ 0 & Y_{v|v|}|v| + Y_{v|r|}|r| & Y_{r|v|}|v| + Y_{r|r|}|r| \\ 0 & N_{v|v|}|v| + N_{v|r|}|r| & N_{r|v|}|v| + N_{r|r|}|r| \end{bmatrix} \end{aligned} \tag{2.38}$$

where x_g and y_g are the coordinates of the USV center of gravity in the body-fixed frame and I_z denotes the moment of inertia about the Z-axis. If we designate the body-fixed origin at the vehicle center of gravity, $x_g, y_g = 0$. The above expression uses SNAME [32] notation for hydrodynamic derivatives. For example, the hydrodynamic added mass force Y along the y-axis due to acceleration $\dot{u}$ in the x-direction is written as:

$$Y = -Y_{\dot{u}}\dot{u}; Y_{\dot{u}} := \frac{\partial Y}{\partial \dot{u}}$$

For a rudderless USV with double thrusters, the propulsion force component τ can be given as:

$$\tau = [\tau_u, 0, \tau_r]^T \tag{2.39}$$

τ_u represents the surge force is given by $\tau_u = X_{P1} + X_{P2}$ and the yaw moment is given by $\tau_r = (X_{P1} - X_{P2}).d_p$ where X_{P1}, X_{P2} denote the thrust values from port and starboard thrusters while d_p is the lever arm from each thruster to the vehicle's central plane of symmetry.

In order to simply the model, we can make the following assumptions and reduce the number of parameters required to sufficiently represent the vehicle dynamics.

Inertia	Damping	Vehicle
$m11 = 60.28$	$X_{\dot{u}} = 5.28$	$L = 2.0m$
$m22 = 137.5$	$Y_{\dot{v}} = 82.5$	$B = 1.08m$
$m33 = 45.26$	$N_{\dot{r}} = 27.1$	$m = 55.0Kg$
$m23 = 11.0$	$X_u = -77.55$	$d_p = 0.395m$
$m32 = 11.0$	$Y_v = 0$	-
	$N_r = -45.26$	-

Table 2.1: Otter system parameters [33]

1. At slow speeds, the non-linear drag terms can be ignored i.e., $D_n(\nu) = 0$
2. The effect of environmental forces can also be neglected by assuming that the USV operates in a calm environment. Therefore, $\tau_E = 0$.
3. A combination of approximate fore-aft symmetry and light draft suggests that the sway force arising from yaw rotation and the yaw moment induced by the acceleration in the sway direction are much smaller than the inertial and added mass terms. Therefore, in the Coriolis and centripetal matrix $C(\nu)$, $N_{\dot{v}} = 0$, $Y_{\dot{r}} = 0$

From the above equations and simplification,equation 2.35 can be re-framed as:

$$\dot{\nu} = -M^{-1}(C(\nu) + D(\nu))\nu + M^{-1}\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}\tau \tag{2.40}$$

Above equation can be expressed in compact state-space notation as:

$$\dot{\boldsymbol{\nu}} = C\boldsymbol{\nu} + D\mathbf{u}' \tag{2.41}$$

where $\nu = [u, v, r]$ is the state vector, C is the system matrix, D is the control matrix, and $\tau = [\tau_u, \tau_r]^T$ is the control input.

In case of Otter, inertia parameters, hydrodynamic derivatives and other vehicle particulars are obtained from MSS github repository by Fossen.T.I et al [33] and the same are given in the table below:

To summarize the section, the dynamic model of the USV is derived through kinematic and kinetic representations.

Kinematic model:

$$\mathbf{x} = A\mathbf{x} + B\mathbf{u} \tag{2.42}$$

where $\mathbf{x} = [x, y, U, \psi]^T$ is the state vector, A is the system matrix, B is the control matrix and $\mathbf{u} = [a, r]^T$ is the control input.

Kinetic model:

$$\dot{\boldsymbol{\nu}} = C\boldsymbol{\nu} + D\mathbf{u}' \tag{2.43}$$

where $\nu = [u, v, r]$ is the state vector, C is the system matrix, D is the control matrix, and $\tau = [\tau_u, \tau_r]^T$ is the control input.

Chapter 3

Implementation - Predictive Digital Twin

As discussed in the previous chapter, 'Probabilistic Graphical Model' is a general framework on top which several digital twin applications can be built. The current work uses 'Actuator Fault Diagnosis' as a demonstrative application for the current digital twin. In this context, the digital twin model is served in two stages.

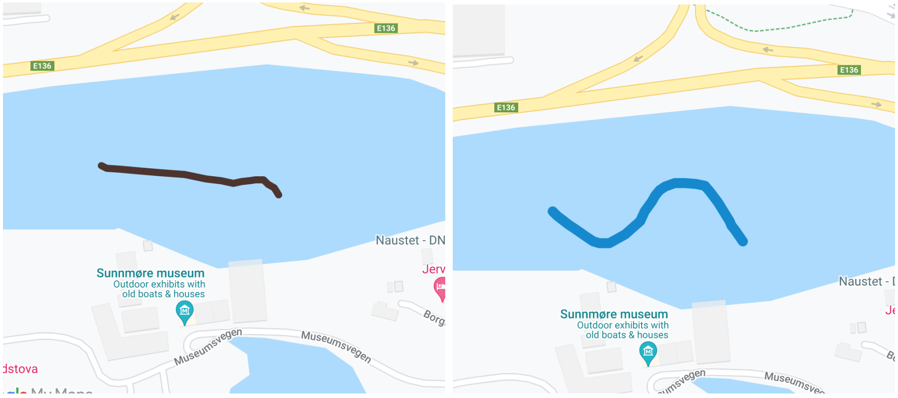
1. Experimental calibration and validation of digital twin
2. Prediction using the validated digital twin

In the first stage, the model is calibrated and validated with experiments using data from the physical asset i.e., Otter. In the second stage, this validated set-up can be further used in predicting the future states of Otter given the actuator faults.

The 'Probabilistic Graphical Model' mentioned in the previous chapter makes it easier to break-down the components of the digital twin and describe them independently.

3.1 Experimental set-up for physical asset (Otter):

This section deals with the physical aspects of the asset-twin system.It describes the experimental set-up, data acquisition and data-processing as well as the control inputs to Otter.



(a) Path 1: Straight Line Maneuver **(b)** Path 1: S-Maneuver

3.1.1 Experiments:

In order to demonstrate 'Actuator Fault Diagnosis' as a digital twin application, real data from Otter with faulty actuators is needed. The faults can then be estimated using the proposed algorithms. For this purpose, Otter is taken to sea-trials with different propeller configurations, some with known faults, for example broken propeller fins.

The sea-trials were carried out at Sunnmøre Museum Dock in Ålesund, Norway. Otter is made to track two paths in way-point mode using VCS software and control station provided by Maritime Robotics, the manufacturer of Otter.

Sea-trial Paths:

Path 1 is a straight-line maneuver and Path 2 is an S-maneuver.The paths are shown in Fig.3.1a and Fig.3.1b. The reason for choosing these different paths is to test the performance of the fault detection algorithm in different scenarios.

Propeller configurations:

In order to develop a robust fault detection algorithm for Otter's actuators, a suitable fault should first be introduced to test and validate the algorithm. Several actuator faults are possible in case of Unmanned Surface Vehicles as mentioned in the literature [34]. One such faults is the breaking of propeller blades which can adversely affect the performance of the USV [15]. Propeller fault is also relatively easy and inexpensive to introduce.

Otter has two Torqeedo Ultralight 403 AC motors fitted with a v10/p350 propeller each as can be seen in Fig.

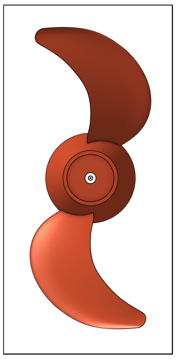
Figure 3.2: Otter's original propeller

A 3D model of the original propeller is created using CAESES®, a propeller design software from From Friendship Systems™ and further modified in On-Shape™, a cloud-based CAD software. The 3D model is made with the help of visual reference from the original propeller. Therefore, it is not precisely similar to the original propeller. However, this difference will either not matter due to the robust closed-loop control system of Otter or it will be picked up as a fault by the fault diagnosis algorithm. The 3D model is further modified by creating a broken blade to simulate a faulty propeller. Both the 3D models, intact and broken, are then 3D printed on Prusa MK3S ® 3D printers. The 3D model and 3D print outputs are shown in Fig.

3.2 Digital-twin set-up:

Once the experiment is set-up as described above and Otter is ready to transmit data,we shift our focus to setting up the digital twin that can ultimately help us estimate the actuator faults.

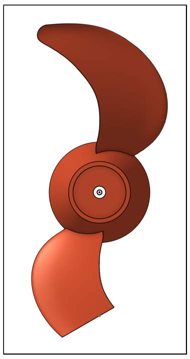
Once again, the graphical model framework comes in handy in understanding the various components of the digital twin and their purpose. The graph and its propagation in time (one time step for brevity) is shown in Fig. 3.5



(a) 3D CAD model of the original otter propeller

(b) 3D printed version of original otter propeller

Figure 3.3: Replicated propeller



(a) 3D CAD model of the broken propeller

(b) 3D printed version of the broken propeller

Figure 3.4: Broken propeller

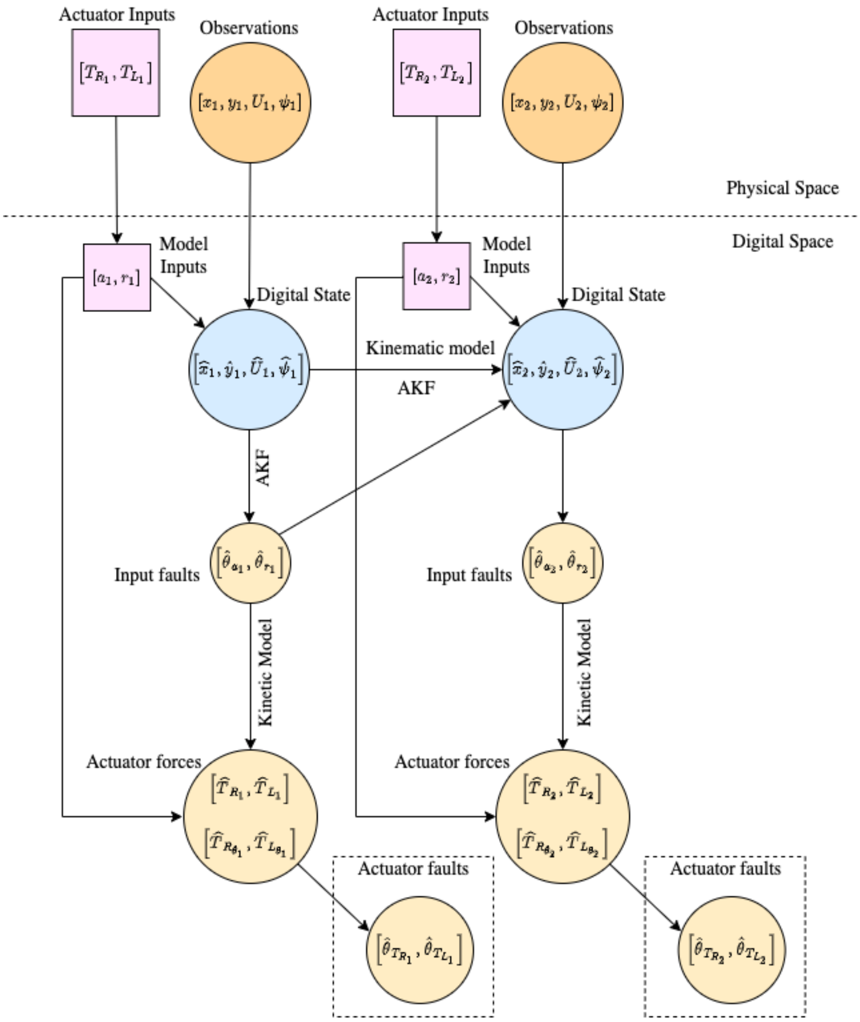


Figure 3.5: Graphical framework of the Digital Twin equipped with state and fault parameter estimation to accomplish actuator fault diagnosis

We begin our discussion with a discussion on some of the methods used to handle the practical issues and adapt the -algorithm to real data from the experiments. Subsequently, we discuss the implementation of the USV kinematic model presented in 2.4.1 as the system model in the AEKF-based state and fault estimation algorithm.

3.2.1 Actuator Fault Diagnosis of Otter's thrusters:

In this section, a model-based method for detecting faults in Otter's thrusters is discussed. This method is based on the Adaptive Kalman Filter algorithm presented in the previous chapter. In the first step, we estimate the input faults in 'acceleration' and 'yaw-rate' as fault parameters using Otter's 3-DOF kinematic model. In the second step, we identify the faults in the actuators using Otter's kinetic model 2.4.2 and the input fault parameters from the first step. The complete schematic for Actuator Fault Diagnosis is shown in Fig. **??** and Fig. **??**

3.2.2 Step 1:Data acquisition from Otter:

For implementing the Actuator Fault Diagnosis on the above system, it should satisfy the completely-observable, completely controllable condition [10]. For this purpose, we need state-observations $[x, y, U, \chi]$ from Otter. Otter is equipped with a GNSS sensor which logs the location, orientation and speed as Latitude, Longitude, SOG (speed-over-ground) and COG(course-over-ground) in world frame which can be converted to $[x, y, U, \chi]$ in North-East frame.

3.2.3 Data Processing

Before we proceed to the fault estimation using AEKF, the state-measurements from the GNSS sensor should be converted to appropriate units.

North-east positions from Latitude and Longitude:

Assume that the flat Earth coordinate origin is located at longitude and latitude (l_0, μ_0). With the origin as reference, we have [35]:

$$\Delta l := l - l_0 \tag{3.1}$$

$$\Delta \mu := \mu - \mu_0 \tag{3.2}$$

The Earth radius of curvature in the prime vertical R_N and the radius of curvature in the meridian R_M are given by equations 3.3 and 3.4 [36]:

$$R_N = \frac{a}{\sqrt{1 - e^2 \sin^2(\mu_0)}} \tag{3.3}$$

$$R_M = R_N - \frac{1 - e^2}{\sqrt{1 - e^2 \sin^2(\mu_0)}} \tag{3.4}$$

where $a = 6378137m$ is the semi-minor axis (equatorial radius) and $e = 0.0818$ is the Earth eccentricity. Small changes in the North and East positions (x, y) are given by equations 3.5 and 3.6

$$x = \frac{\Delta l}{\text{atan2}(1, R_N \cos(\mu_0))} \tag{3.5}$$

$$y = \frac{\Delta \mu}{\text{atan2}(1, R_M)} \tag{3.6}$$

where atan2(y, x) is the inverse tangent confining the result to $[-\pi, \pi]$

Course angle:

The course angle measurement(COG) provided by GNSS is converted to smallest signed angle to fit in the range of $[-\pi, \pi]$.

$$\theta = \begin{cases} COG & if\, 0 < COG < \pi \\ COG - 2\pi & if\, COG > 2\pi \end{cases} \tag{3.7}$$

3.2.4 Step 2: Input Data and Filtering

In order to perform fault diagnosis of Otter's thrusters, actuator inputs in the form of input voltages or forces are required for a direct fault estimation. But at the time of writing this thesis, such data is not accessible to the user directly. Moreover, the kinematic model used in the system representation takes the acceleration and yaw-rate as the input vector. Since, Otter in its current configuration, is not equipped with an Inertial Measurement Unit or accelerometers, direct measurements of the required acceleration and yaw-rate is not possible. In order to overcome this problem, we use the method proposed in [35]. By a combination of backward difference and low-pass filtering, we acquire the required inputs to the kinematic model from the state measurements. As a minor modification to the cited method, use a forward-difference operator and a butter-worth low-pass filter to achieve better results.

Forward difference operation:

Let U_{t+1}, U_t and ψ_{t+1},ψ_t be the surge velocity and course angle measurement at time $t+1$ and t respectively and dt be the sample time at that instant. Then, the input acceleration a_t and yaw-rate r_t at time t are given by:

$$a_t = \frac{U_{t+1} - U_t}{dt} \tag{3.8}$$

$$r_t = \frac{\psi_{t+1} - \psi_t}{dt} \tag{3.9}$$

Data filtering:

To eliminate noise from the above equations, we use a low-pass butter worth filter available as `scipy.signal.butter` library function for Python users. Given the popularity of the filter, implementation in other languages intended for scientific applications may also be found.

We chose a cut-off frequency of 1Hz for acceleration and yaw-rate respectively as this gives us satisfactory filtered results.

3.2.5 Step 3: Actuator Fault Detection using Adaptive Extended Kalman Filter

This step is based on the Adaptive Kalman Filter(AEKF) algorithm presented in 2.3.3. For the system model, we use the kinematic model discussed above along with the process and measurement noises terms. The input is derived from the COG and SOG measurements as discussed in Step 2.

The system from equation can written in discrete form as

$$x(k+1) = f(x(k)) + B(k)u(k) + \Psi(k)\theta + w(k) \tag{3.10}$$

$$y(k) = C(k)x(k) + v(k) \tag{3.11}$$

where

$$f(x(k)) = \begin{bmatrix} U\sin\psi \\ U\cos\psi \\ U \\ \psi \end{bmatrix} \tag{3.12}$$

The Jacobian of the above matrix is given by $F(x(k)) = \partial f(x(k))/\partial x(k)|$

$$F(x(k)) = \begin{bmatrix} 0 & 0 & \sin\psi & U\cos\psi \\ 0 & 0 & \cos\psi & -U\sin\psi \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

The subsequent steps are given by equations 2.8 to 2.19 and are omitted here to avoid repetition.

A signal flow diagram for Steps 1 to 3 is shown in Fig.**??**

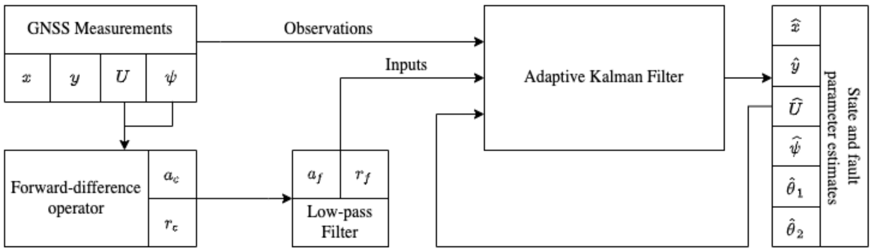


Figure 3.6: Step 1-3: Fault Parameter Estimation

3.2.6 Step 4: Actuator Fault Diagnosis

From the above, we obtain the state estimates and fault parameter estimates for every time step of the experiment. However, the fault parameters are related to model inputs $[a, r]$. While, these estimates are useful in detecting an input fault in the system, it is difficult to correlate them to the actuators which is the main aim of this thesis. Therefore, we use the kinetic model of the system to obtain the actuator forces from surge acceleration and yaw angular acceleration.

Since, we only consider surge motion in this experiment, we can safely assume $\dot{v} = 0$ in the sway direction.Hence, we can ignore the second row and second column terms and rewrite equation 2.41 as below

$$\begin{bmatrix} T_R + T_L \\ (T_R - T_L).d_p \end{bmatrix} = M\begin{bmatrix} \dot{u} \\ \dot{r} \end{bmatrix} - (C(\nu) + D(\nu))\begin{bmatrix} u \\ r \end{bmatrix} \tag{3.14}$$

Simplifying the above equation gives the direction solution to $T_R and T_L$

$$\begin{bmatrix} 1 & 1 \\ d_p & -d_p \end{bmatrix}\begin{bmatrix} T_R \\ T_L \end{bmatrix} = M\begin{bmatrix} \dot{u} \\ \dot{r} \end{bmatrix} - (C(\nu) + D(\nu))\begin{bmatrix} u \\ r \end{bmatrix} \tag{3.15}$$

$$\begin{bmatrix} T_R \\ T_L \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ d_p & -d_p \end{bmatrix}^{-1} \left[M \begin{bmatrix} \dot{u} \\ \dot{r} \end{bmatrix} - (C(\nu) + D(\nu)) \begin{bmatrix} u \\ r \end{bmatrix} \right] \tag{3.16}$$

Similar equations can be used to derive the actuator forces due to faulty thrusters by multiplying the acceleration vector $\dot{\nu}$ with fault parameter loss.

$$\begin{bmatrix} T_{R_f} \\ T_{L_f} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ d_p & -d_p \end{bmatrix}^{-1} \left[M \begin{bmatrix} (1-\theta_1)\dot{u} \\ (1-\theta_2)\dot{r} \end{bmatrix} - (C(\nu) + D(\nu)) \begin{bmatrix} u \\ r \end{bmatrix} \right] \tag{3.17}$$

The yaw acceleration can be acquired from input yaw-rate and sample time similar by using the same methods discussion in Step 2 3.2.4.

$$\dot{r_t} = \frac{r_{t+1} - r_t}{dt} \tag{3.18}$$

For filtering, once again we use `scipy.signal.butter` with a cutoff frequency of 0.9Hz.

Finally, the actuator fault values can be computed, using

$$\boxed{\theta_R = 1 - \frac{T_{R_f}}{T_R}} \boxed{\theta_L = 1 - \frac{T_{L_f}}{T_L}} \tag{3.19}$$

A signal flow diagram for Step 4 is shown in Fig.3.2.6

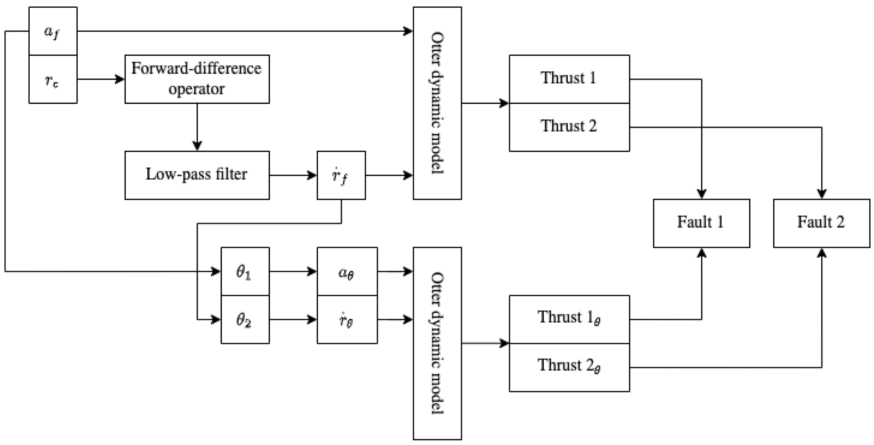


Figure 3.7: Step 4: Final Actuator Fault Estimation

3.2.7 Tools and Techniques:

The above steps are implemented in Python using standard libraries like `pandas` and `numpy`. Additionally, parts of MSS library[33] originally written in Matlab were re-written in Python to compute the hydrodynamic forces in the kinetic model.

Chapter 4

Results and Discussion

In this section, the results from applying AEKF-based FDI algorithm on the data collected from field experiments with Otter are discussed.Before, we analyse the algorithm's performance on experimental data, we apply it on simulated data from Otter's kinematic model. This provides important insights on some key features of the method that are useful in interpreting the actual results.

4.1 Fault Parameter Estimation from simulation data:

Case 1:

As a simple demonstration of the FDI method, a kinematic USV model with known actuator faults is used in simulation to generate the state data. AEKF algorithm is then used on this model to estimate states and fault parameters. The results of the simulation and estimation are shown in Fig.4.1. The simulation parameters are shown in Table.4.1

Simulation run time	20 s
Sampling time	0.001 s
Inputs	$[1,1]^T$ for $t \in [0,10)$ $[2,0.5]^T$ for $t \in [10,15)$ $[-0.5,0.4]^T$ for $t \in [15,20)$
Faults	$[0.5,0.3]^T$ for $t \in [5,20]$

Table 4.1: Simulation parameters for Case 1

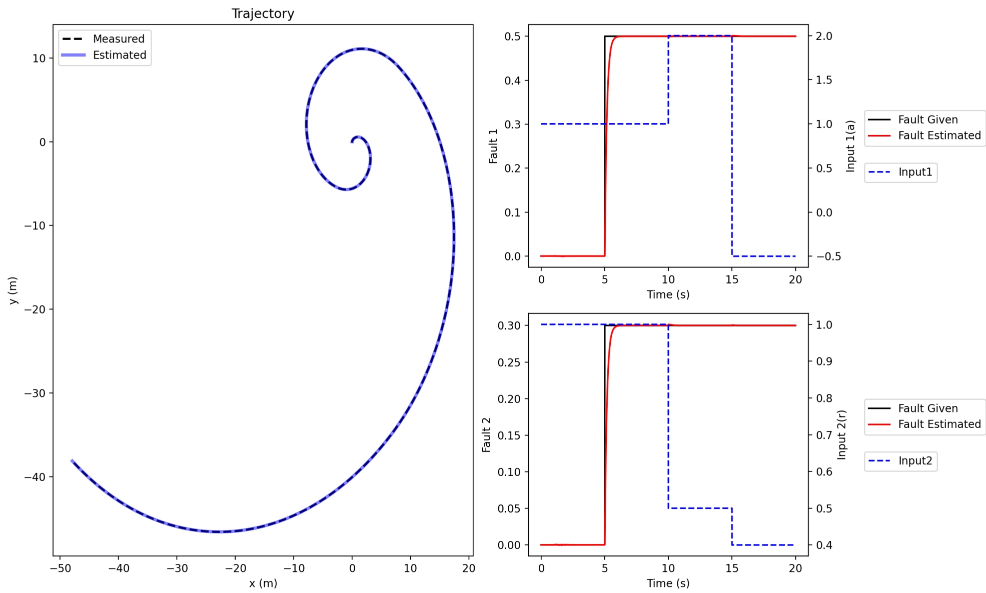
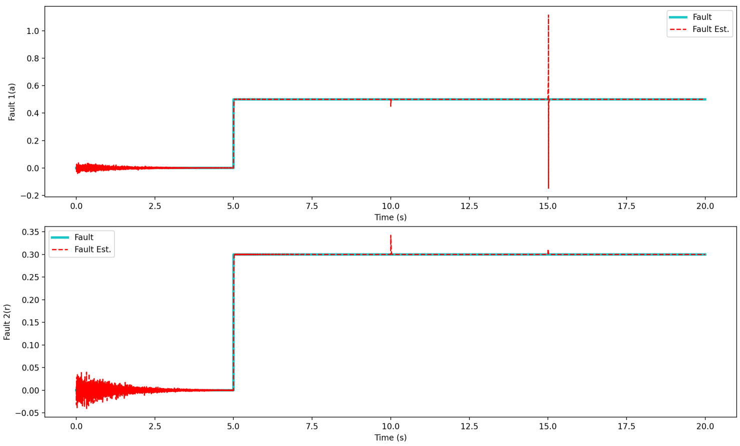


Figure 4.1: State and fault parameter estimation

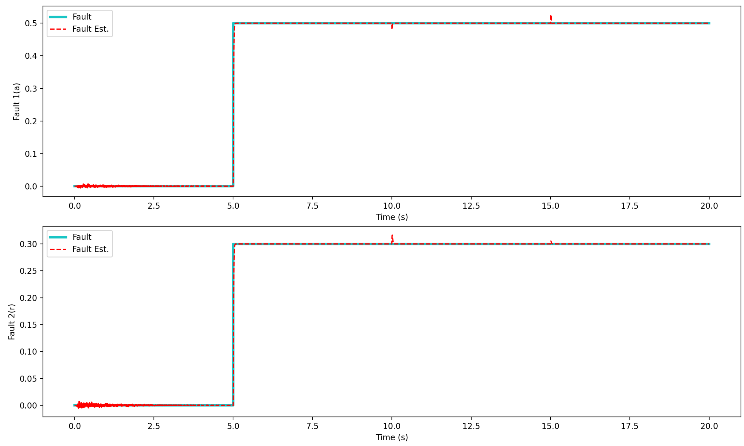
Discussion:

The trajectory plot on the left of Fig.4.1 shows the state estimation with respect to the measured values[Latitude, Longitude, Speed-over-ground, Course-over-ground] from Otter's GNSS sensor. As expected, the Kalman-filter performs well in state estimation eliminating the process and measurement noise. The inputs given to excite the system are surge-acceleration 'a' and yaw-rate 'r'.The fault parameters $[\theta_1, \theta_2]$ on these inputs are estimated as 'Fault 1' and 'Fault 2' in the top-right and bottom-right plots.In this case, step faults of magnitude $[0.5, 0.3]$ are given at $t = 5s$ for both inputs. The faults are estimated accurately by the AEKF algorithm as it can be seen from plot. This is a simple demonstration of the fault parameter estimation algorithm in an ideal case with step inputs and step faults. An important parameter that effects the convergence of estimation is the forgetting factor λ. This acts as a tuning parameter with higher values suppressing the noise and resulting in smoother estimates but also slowing down the convergence.

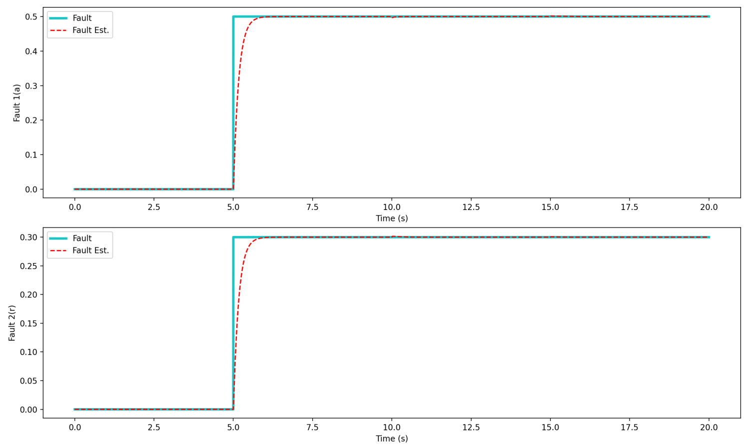
It can be seen from the plots in Fig.4.2, that a low value of λ results in faster but noisy estimation. On the other hand, a larger value results in slower yet smoother estimates. Smoothness is more important in reality as it helps us to infer the fault value easily compared to noisy estimates. Therefore, a forgetting factor of $\lambda = 0.995$ is chosen for further simulations as well as in experiments.The problem of slower convergence is handled by selecting appropriate number of iterations in



(a) $\lambda = 0.1$



(b) $\lambda = 0.9$



(c) $\lambda = 0.995$

Figure 4.2: Effect of forgetting factor on convergence

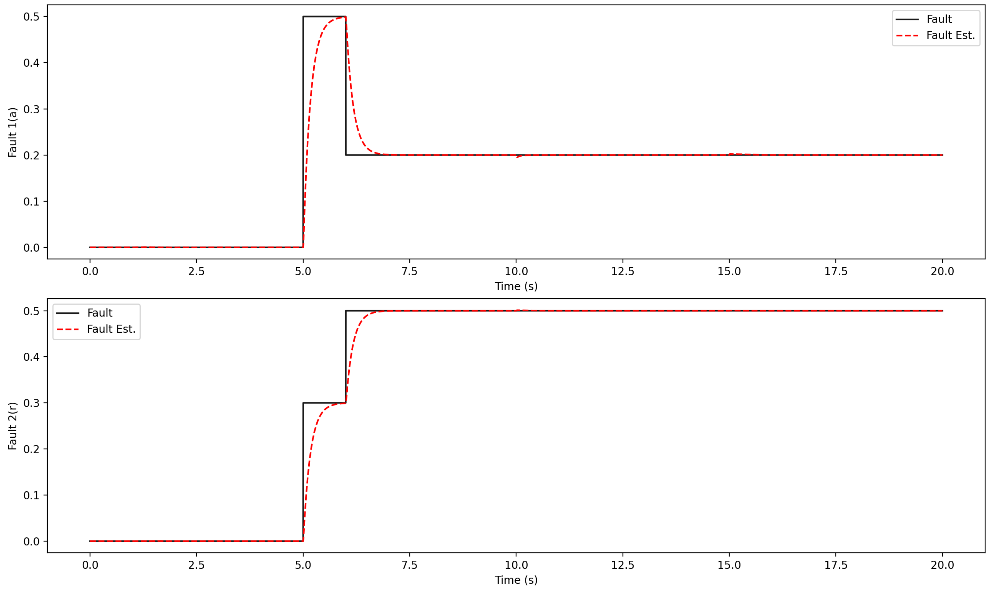


Figure 4.3: Fault estimation in 1000 iterations from t=5s to t=6s

the fault phase.

Case 2:

In this case, the effect of number of iterations available per piece-wise continuous fault parameter on the convergence of the estimation algorithm is studied. For this purpose, a fault is introduced from t=5s to 6s keeping other parameters same as the previous case. This gives 1000 iterations ($\frac{\text{time}}{\text{sample time}} = 1/0.001 = 1000$) for the algorithm to estimate the fault. The results are shown are in Fig. 4.3. It can be seen that the true fault values could be estimated in 1000 iterations before the faults changed. The same result can be achieved in 500 iterations with a small loss in convergence as can be seen in Fig.4.4.

Discussion:

Therefore, from the above analysis, it can be inferred that a forgetting factor $\lambda = 0.995$ and an iteration number of 500 is ideal to achieve satisfactory convergence on fault estimation results.

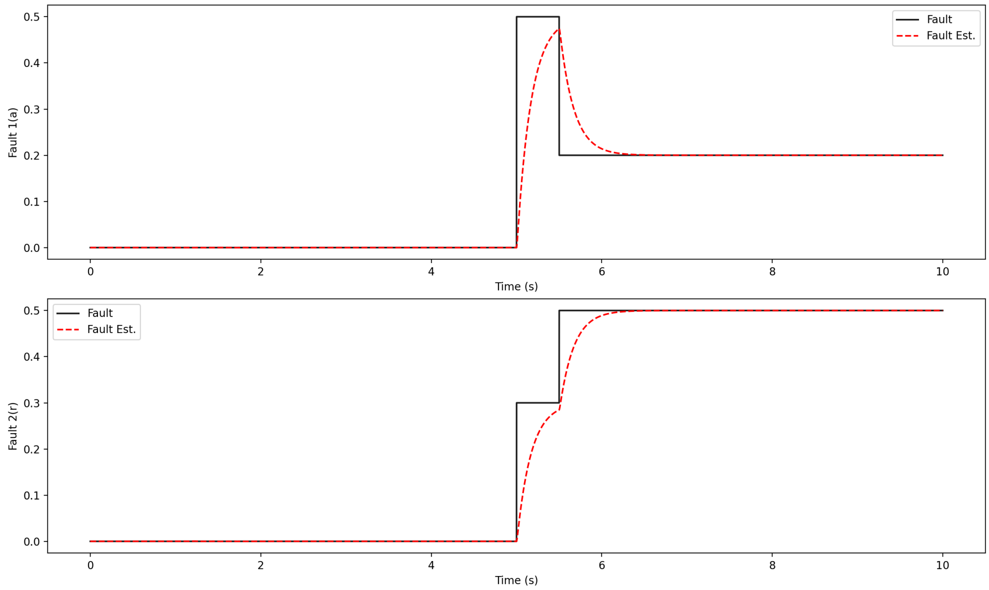


Figure 4.4: Fault estimation in 500 iterations from t=5s to t=5.5s

4.1.1 Case 3:

In this case, we look at the effect of input on estimation convergence. Once again, we use the same simulated system from the previous case, only changing the input and fault pattern to understand the relationship between the two.We change the input while fault estimation is converging to the actual fault value.The results can be seen in Fig.4.5.

Discussion:

The plot in Fig.4.5 shows that when the input is removed during the estimation i.e., made to be zero, the algorithm settles on a value reached thus far hence slowing down the convergence. Once the input is fed back to the system, the algorithm eventually converges to the actual value.This is a noteworthy result, as it demonstrates one of the conditions of the AEKF algorithm i.e., *persistent excitation*. It implies that the estimation will only work during the time the input is non-zero. The same can be seen in the next section where real input from the experiments is used on the simulation data to establish the same result.

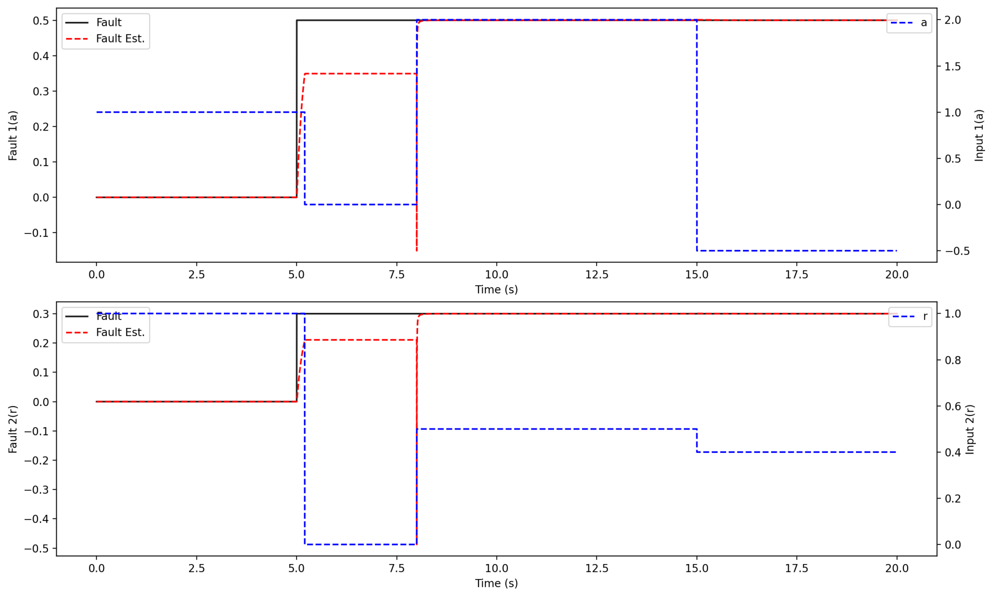


Figure 4.5: Effect of input on convergence

4.1.2 Case 4:

In this section, we continue our analysis of AEKF-based fault estimation algorithm with respect to the nature of input.In the previous section, we used a step input which reduces to zero during the estimation effectively prolonging the convergence. Now, we use the input data acquired during the experiments to examine the performance of the estimation algorithm on real data. The input data is obtained from the forward speed and course angle measurements using the forward-difference operation and low-pass filter methods discussed in the previous chapter in 3.2.4.The results of applying this input on the simulated system is shown in Fig.4.6 and the simulation parameters are given in Table 4.2. It is important to note that the real data is collected for 120s. However, the data is replicated to fill in an additional 120s to achieve convergence of estimation.

Simulation run time	240 s
Sampling time	0.2 s
Input	From real data
Faults	$[0.1, 0.5]$ $t \in [25, 240]$

Table 4.2: Simulation parameters for Case 4

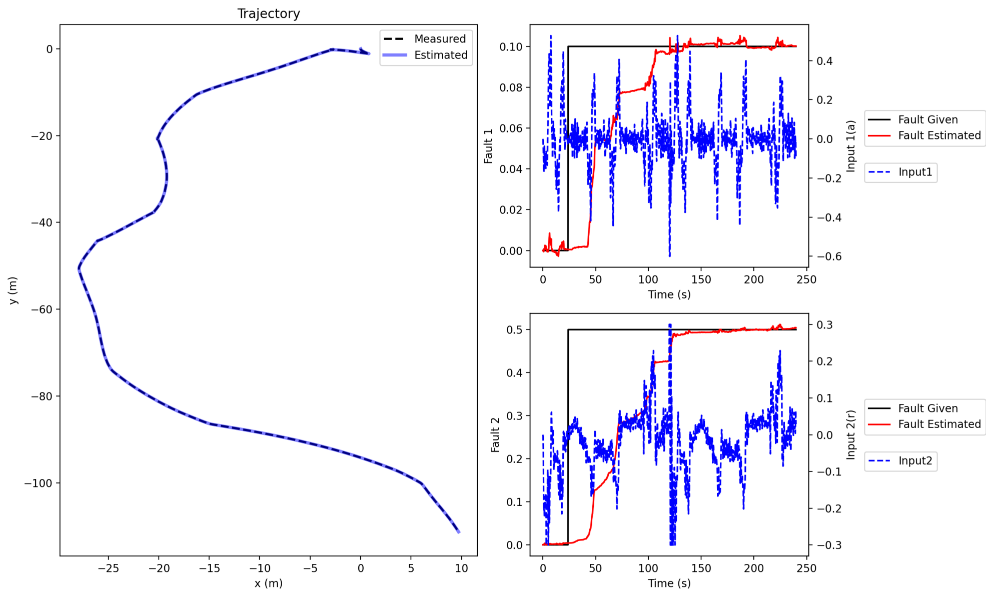


Figure 4.6: State and fault estimation using real inputs on simulated system

Discussion:

As we can see from the fault estimation plot on the right of Fig. 4.6, the convergence of the estimation algorithm happens in steps, moving closer to the actual value in the presence of input and staying flat when the input is zero. Since, the input is from Otter performing an S-Maneuver, it is expected to have zero surge-acceleration and yaw-rate during the path. These points result in zero input where the fault estimation algorithm stops to function due to the nature of the algorithm.The result is in line with our earlier discussion on convergence in case of changing input, especially when input is zero. The fault occurs at t=25s and the algorithm converges at approx. t=150s. Therefore, the convergence took 125s which is equivalent to 625 ($\frac{125}{0.2} = 625$) iterations similar to previously mentioned 500 iterations.

As a test to confirm the accuracy of the estimation, a zero given fault with the same input as above leads to a non-converging estimation with fault estimate fluctuating over zero.The same can be seen in Fig. 4.7 validating the obvious of premise of zero given fault results in zero estimated fault.

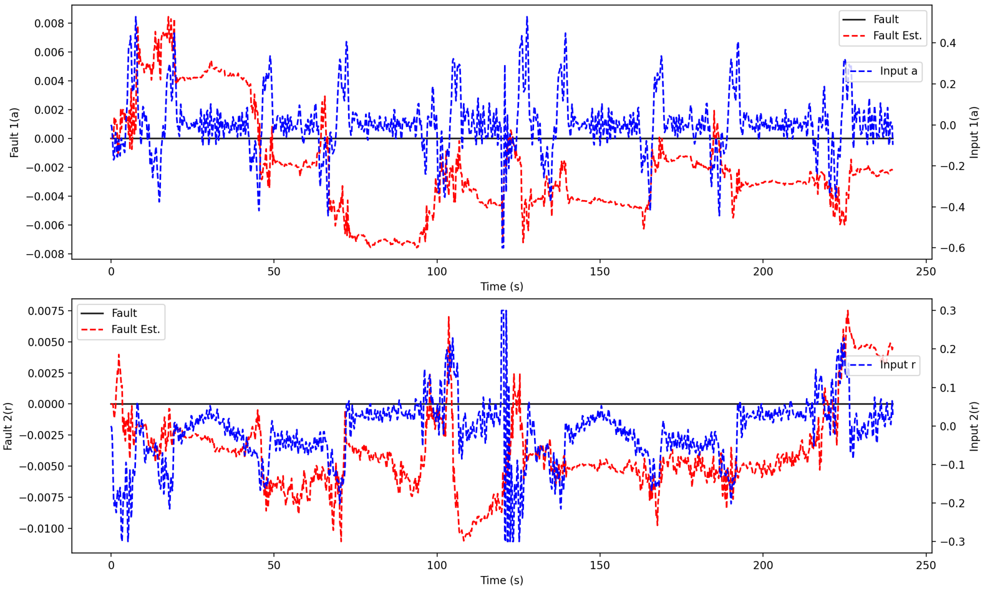


Figure 4.7: Fault estimation with zero given fault and real input data

4.1.3 Summary:

From the above discussion, we can summarise the below key points:

1. Forgetting factor λ influences the speed and smoothness of convergence. For our application, we desire smooth estimates over faster ones to make meaningful inference of fault parameters. $\lambda = 0.995$ is found to be a suitable value.
2. With the above forgetting factor, the algorithm is observed to take around 500 iterations to reach satisfactory convergence.The sampling period and data size should be selected accordingly to contain approximately 500 iterations. Recall that Number of iterations $= \frac{\text{Total time}}{\text{Sampling period}}$
3. The fault estimation functions only with non-zero input.Hence, absence of excitation, leads to a slower convergence with the estimation falling flat during zero input phase. The fault estimate takes a step shape due to this as can be seen in Fig.4.5 and Fig.4.6

The points summarised above will be useful in interpreting the results and inferring the final fault estimate value.

4.2 Fault parameter estimation with experimental data from physical asset (Otter USV):

In this section, we apply the AEKF algorithm using the data obtained through field experiments on Otter both with faulty and faultless[1] propellers. Tests were carried out in four different propeller configurations with two test paths for each test giving out a total of eight test cases. We categorise them into four cases for each propeller configuration and divide each into two sub-cases one for each path.

4.2.1 Test 1

In this test, we operate Otter in its standard propeller configuration i.e, manufacturer supplied propellers through two different maneuvers. This can be treated as a faultless case and can be used as a benchmark for results from other propeller configurations. The test details for both the paths are given in Table.4.3a and 4.3b

Test run time	60 s
Sampling time	0.2 s
Iterations	300
Path	Straight line
Propellers	Stb - Original Port - Original

(a) Test parameters of straight line maneuver

Test run time	120 s
Sampling time	0.2 s
Iterations	600
Path	S-Maneuver
Propellers	Stb - Original Port - Original

(b) Test parameters for S-Maneuver

Table 4.3: Test 1 with original propellers

Discussion

Fig.4.8a and 4.8b show the position and fault parameter estimates for both the maneuvers. From the plots, two observations can be made:

- The fault estimation is different for both cases though the propeller conditions are same.Particularly, the estimated value for S-maneuver is higher compared to the straight line maneuver.
- The fault estimation is non-zero, even though there are no known faults in either of the propellers.

[1]Here, faultless mean original manufacturer supplied propellers which are considered as benchmark

At first, this results may seems bit strange. However, if we recall our discussion from 4.1.1 and 4.1.2, we know that the AEKF algorithm performs fault estimation only with non-zero input. In case of a straight line maneuver in 4.8a, the inputs i.e., acceleration and yaw-rate are close to zero as Otter is operating in a way-point mode where it tracks the input trajectory with constant speed. This is the reason for a zero-fault estimation seen in Fig.4.8a. Coming to the second observation of non-zero faults in inputs in Fig.4.8b,it should be remembered that this indicates the faults in input acceleration and yaw-rate but not the actual actuator faults. We can use these faults to derive the final actuator faults as described in 3.2.6. Once we do that, we get the final actuator fault coefficients as shown in Fig.4.9a and 4.9b which represent the faults in starboard and port thrusters respectively. The actuator fault magnitudes are small with maximum values of 0.02 and 0.01 respectively. However, one interesting result is the appearance of the peaks which coincide with the curvature of the trajectory. In other words, the fault in starboard side thruster has a maximum value when the vehicle is turning left. Similarly, the port side thruster has a maximum value when the vehicle is turning right. This logically follows from the above discussion that the fault parameter magnitude depends on the input magnitude. Since, a right turning thruster will have higher input to its port side thruster, this fault magnitude is also higher during that maneuver.

4.2.2 Test 2

In this test, we change the starboard side propeller to a 3D version of the original propeller. The new propeller is visually similar to the original but has minor differences in geometry and finishing due to modelling and 3D printing errors. In this configuration, we perform both the straight line and S-maneuvers and estimate faults using the recorded data.The test parameters are presented in Tables.4.4a and 4.4b and the results are shown in Fig.4.14 and Fig.4.11

Discussion

Again, the results for straight line maneuver in Fig.4.10a and Fig.4.11a are similar to the ones observed for the respective case in Test 1 for same reasons of zero-inputs.Moreover, the results for S-maneuver shown in Fig.4.10b and Fig.4.11b are also not very different for the corresponding case in Test 1. We can also observe the fault magnitude peaks occur at different points for both the thrusters in accordance with the discussion for Test 1.

Test run time	60 s
Sampling time	0.2 s
Iterations	300
Path	Straight line
Propellers	Stb[a] - New Port - Original

(a) Test parameters for straight line maneuver

[a] Starboard side of Otter

Test run time	120 s
Sampling time	0.2 s
Iterations	600
Path	S-Maneuver
Propellers	Stb - New Port - Original

(b) Test parameters for S-Maneuver

Table 4.4: Test 2 with original and new propellers

4.2.3 Test 3

In this test, we replace both the propellers on Otter with the 3D printed replicas. We do not expect the results to vary largely from the earlier case but still present the results for the sake of completion.

Test run time	60 s
Sampling time	0.2 s
Iterations	300
Path	Straight line
Propellers	Stb - New Port - New

(a) Test parameters for straight line maneuver

Test run time	120 s
Sampling time	0.2 s
Iterations	600
Path	S-Maneuver
Propellers	Stb - New Port - New

(b) Test parameters for S-Maneuver

Table 4.5: Test 3 with new propellers

4.2.4 Test 4

This is the final test aimed at the validating the method by introducing a known fault in the actuators by means of a broken propeller on the port side. All other test parameters, given in Tables.4.6a and 4.6b, are kept the same as the earlier test except the propeller configuration to get comparable results. The results of input fault estimates and final actuator faults are shown in Fig.**??** and Fig.4.15

Discussion

As we introduced a fault in the actuators by means of a broken propeller in the port thruster, we expect to see a actuator fault in Fig.4.15b for port thruster alone

Test run time	60 s
Sampling time	0.2 s
Iterations	300
Path	Straight line
Propellers	Stb - New Port - Broken

(a) Test parameters for straight line maneuver

Test run time	120 s
Sampling time	0.2 s
Iterations	600
Path	S-Maneuver
Propellers	Stb - New Port - Broken

(b) Test parameters for S-Maneuver

Table 4.6: Test 4 with new and broken propellers

while starboard thruster's fault value should be similar to the earlier tests. In line with our expectation, we see a peak in the fault curve of the port thruster with a magnitude of 0.18 corresponding to a fault percentage of 18%. This value is higher than any of the previous cases indicating a fault and thus validating our AEKF based fault detection algorithm and overall fault diagnosis technique. It is important to note that the input fault estimates alone that are produced by AEKF are not sufficient to diagnose a fault as the estimated values are similar for all the test cases. It is by means of the actuator fault values generated by the kinetic model given in 3.2.6 we can conclude the presence of the fault, its location and magnitude.

4.2.5 Compiled results

The compiled results from all the test cases discussed above are presented in Table.4.7 and Table.4.8. As mentioned in 4.1.1 and 4.1.2, we only consider the peak value of the estimation as these are closer to the convergent solution. As we can see from Table.4.7,the input fault parameter value look similar across all the test cases and are insufficient in concluding whether the fault occurred in starboard or port thrusters.This is the because the thrust value in each thruster is a function of both the fault parameter estimates and it is difficult to estimate the thrust fault without information on the system's kinetic model.

Using the system's kinetic model we calculate the fault thrust and actual thrust values from the input data and fault parameter estimates as discussed in 3.2.6. The final peak thrust values computed by this method are compiled in Table.4.8. It can be easily seen that the maximum fault value occurs with the broken propeller case during an S-maneuver. Therefore we can conclude that the actuator fault for the thruster with the given broken propeller is 18%. This completes our fault

		Straight Line		S-Maneuver	
Right prop	**Left prop**	**Fault in 'a'**	**Fault in 'r'**	**Fault in 'a'**	**Fault in 'r'**
Original	Original	0.04	0.02	0.05	0.04
New	Original	0.03	0.02	0.06	0.04
New	New	0.05	0.02	0.06	0.03
New	Broken	0.12	0.01	0.07	0.03

Table 4.7: Compiled results for input fault parameter estimates with peak values

detection and identification attempt. However, we can also spot an anomaly for where a fault of 7% is recorded for the left propeller during the S-maneuver. This value, though lower than the broken propeller fault, is still relatively high compared to other test cases.The reason for this is not truly understood. It is either an outlier or indicates an defect in the method. One possible cause is,the bias caused by using state measurements for calculating input data. Recording independent acceleration values using IMU might mitigate this problem and might lead to more accurate results. Either way, more research is required to identify the root cause for this anomaly.

		Straight Line		S-Maneuver	
Right prop	**Left prop**	**RT fault**	**LT fault**	**RT fault**	**LT fault**
Original	Original	0.0	0.0	0.02	0.01
New	Original	0.0	0.0	0.02	0.01
New	New	0.0	0.0	0.01	0.07
New	Broken	0.0	0.0	0.01	0.18

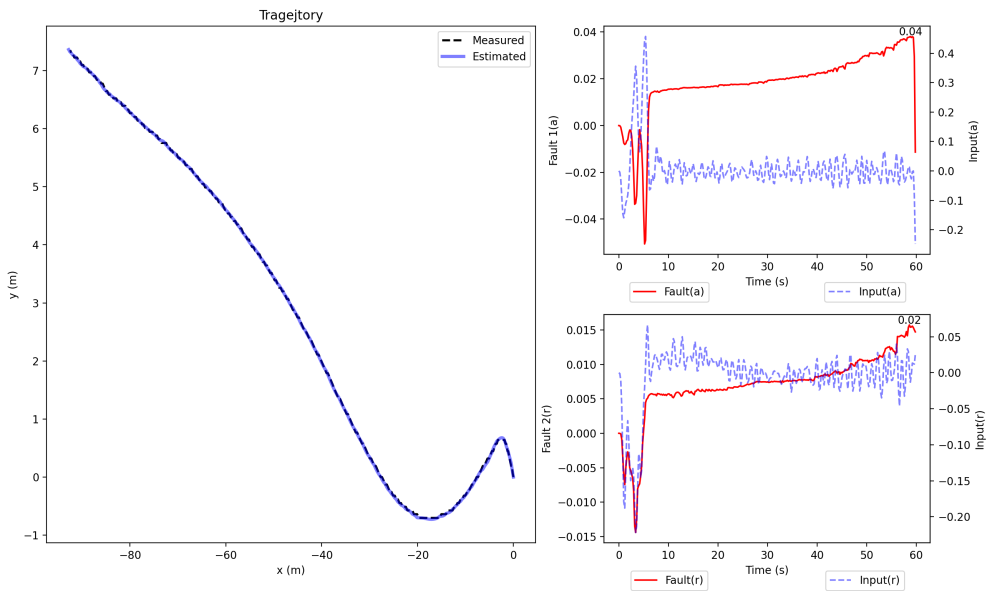
Table 4.8: Compiled results of peak values of actuator faults on each thruster for all test cases

4.3 Calibration and Prediction of the Digital Twin

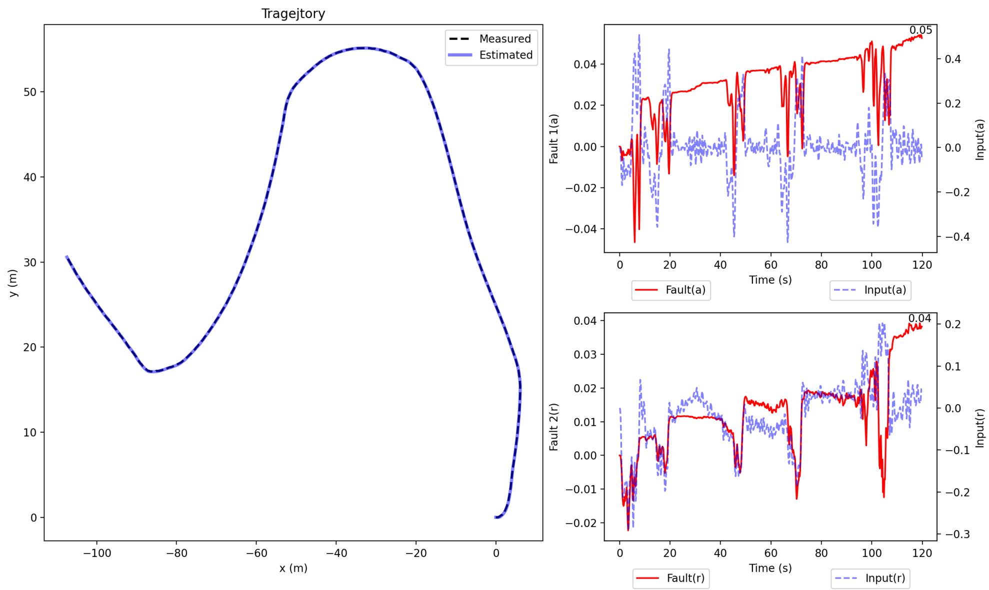
The actuator faults acquired from the previous step can be used to calibrate the Digital Twin. This calibrated Digital Twin is then ready to make predictions of states considering the actuator fault.In the prediction phase, no state observations are available to the Digital Twin and it solely operates based on the system model and inputs. The results are shown in Fig.4.16. The prediction errors for all states are measured using RMSE error percentage and the same is given in Table.4.9. The errors in the Table.4.9 and the Fig.4.16 show that the Digital Twin is able to predict the future states with reasonable accuracy with the prediction error increasing with time.

	x	y	U	ψ
RMSE %	0.35%	9.01%	3.5%	2.61%

Table 4.9: RMS Error for calibrated Digital Twin state predictions in the prediction phase.

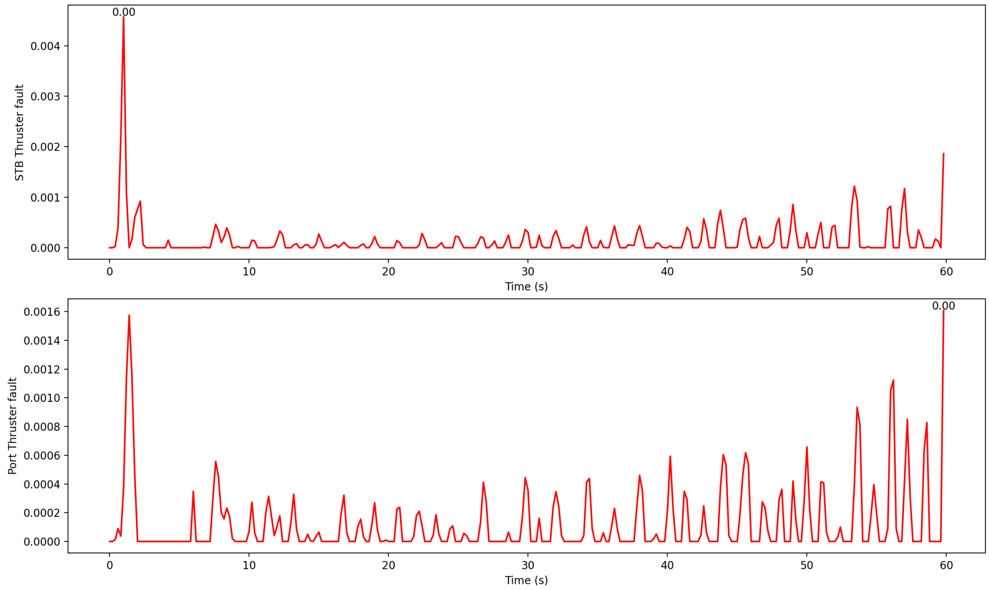


(a) Position(left) and fault(right) estimates during straight line maneuver

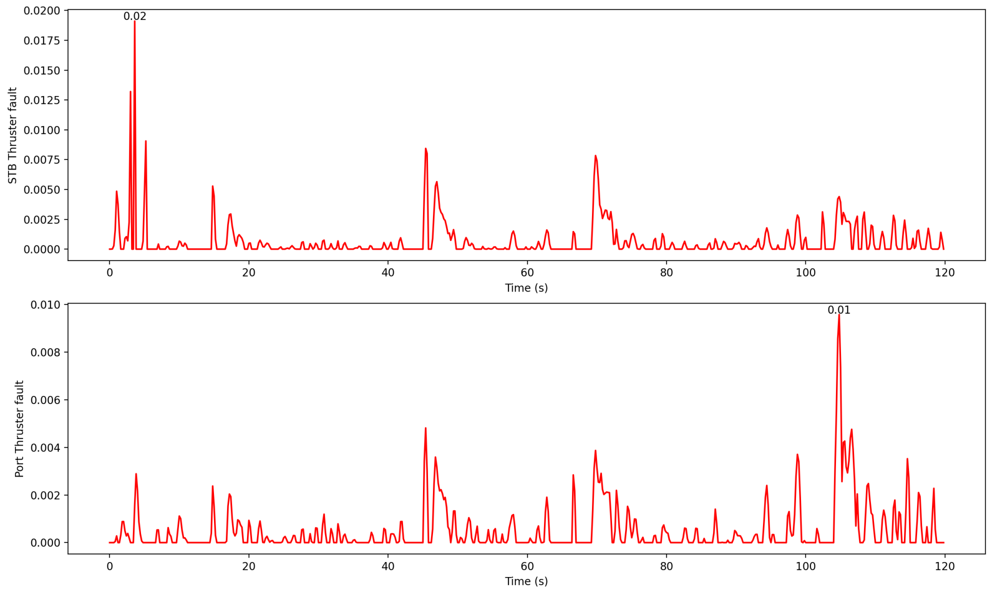


(b) Position(left) and fault(right) estimates during S-maneuver

Figure 4.8: Test 1 with original propellers

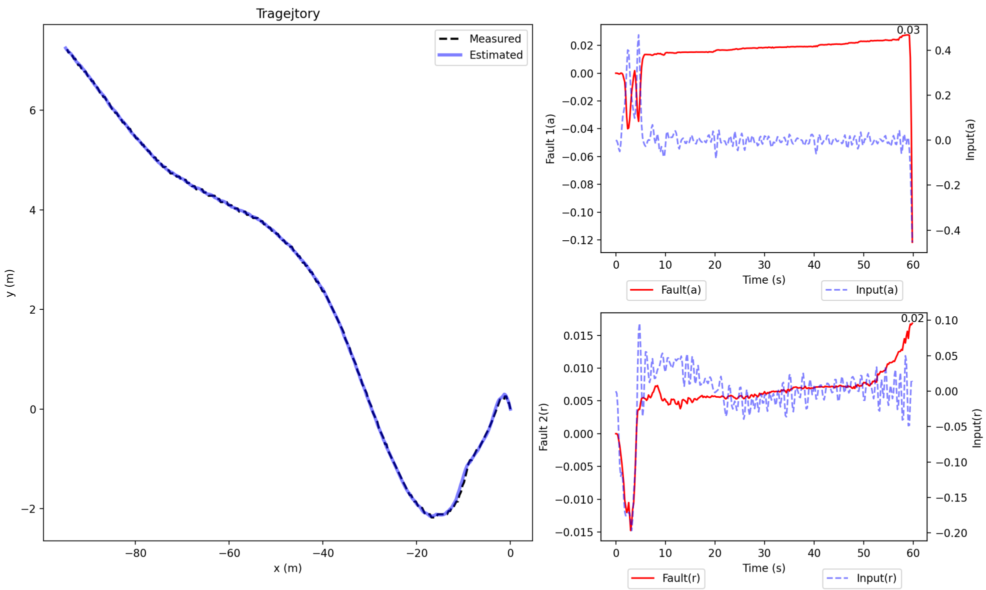


(a) Actuator faults during straight line maneuver

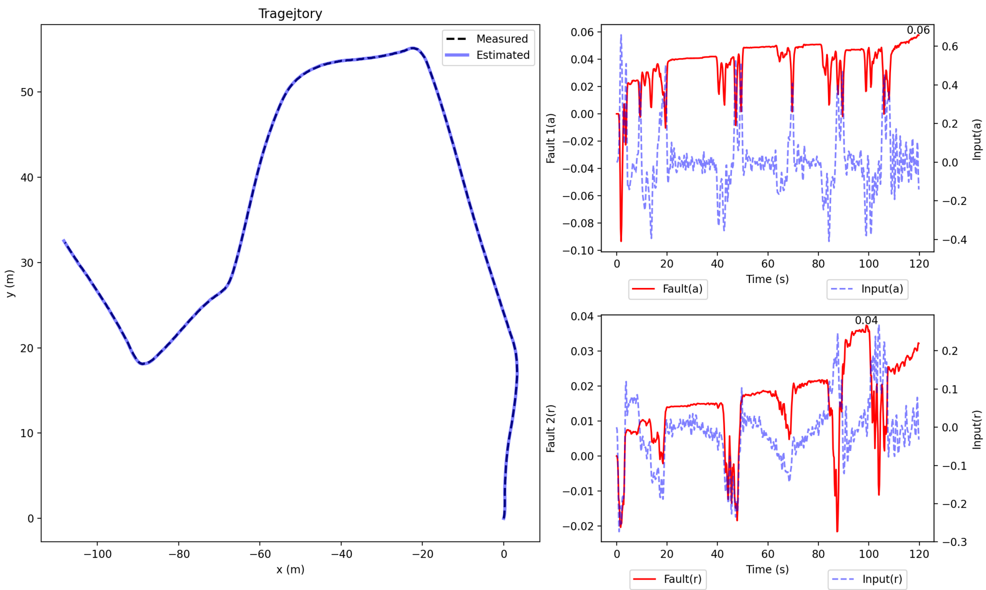


(b) Actuator faults during S-maneuver

Figure 4.9: Actuator faults for original propellers

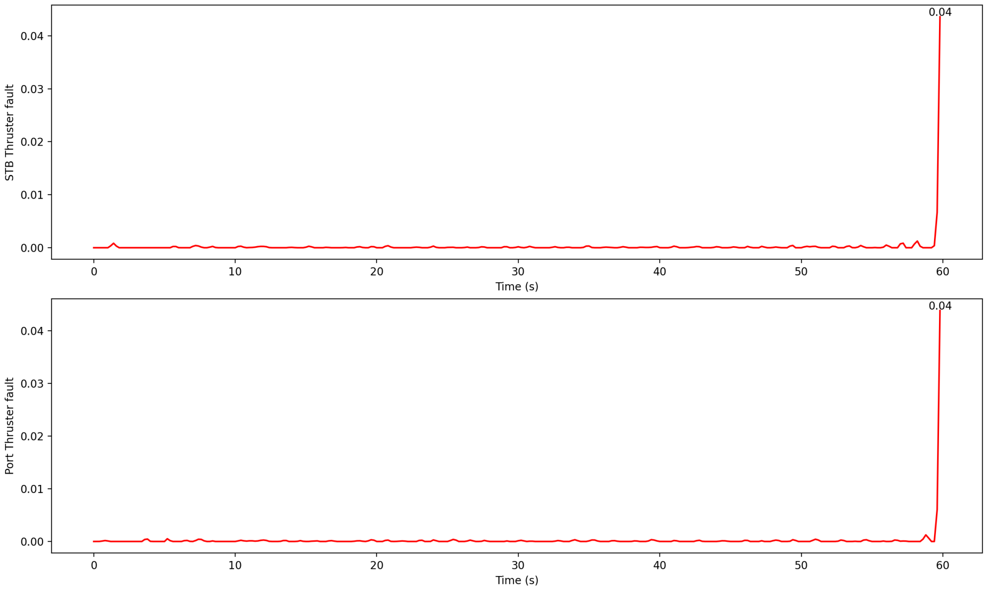


(a) Position(left) and fault parameter(right) estimates during straight line maneuver

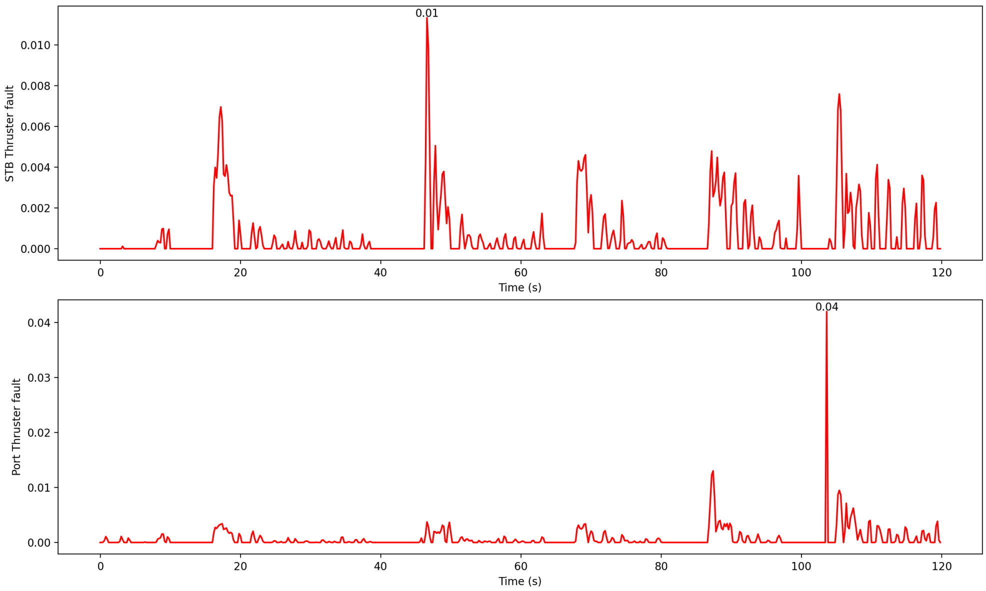


(b) Position and fault parameter estimates during S-maneuver

Figure 4.10: Test 2 with original and new propellers

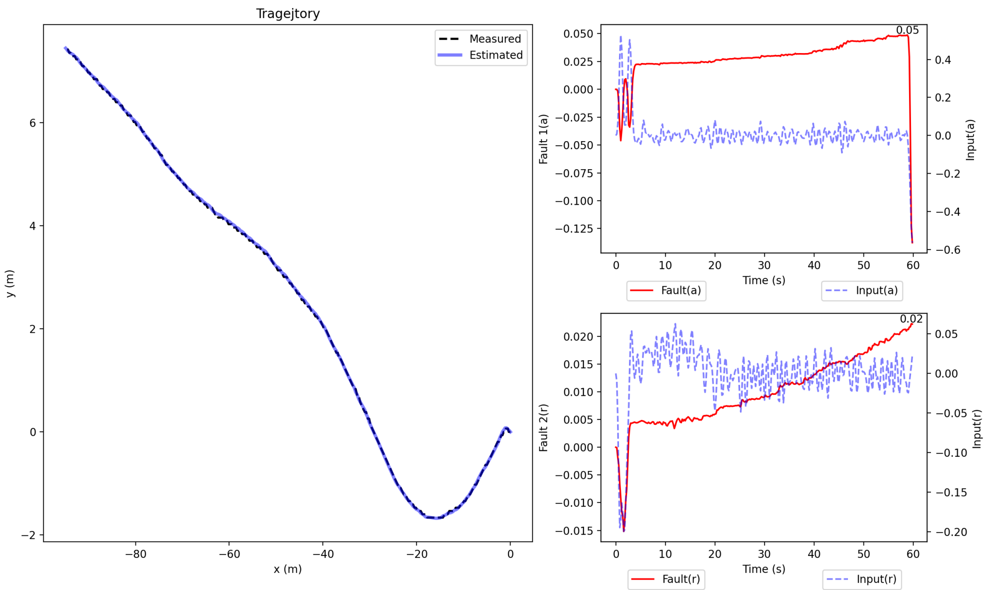


(a) Actuator faults during straight line maneuver

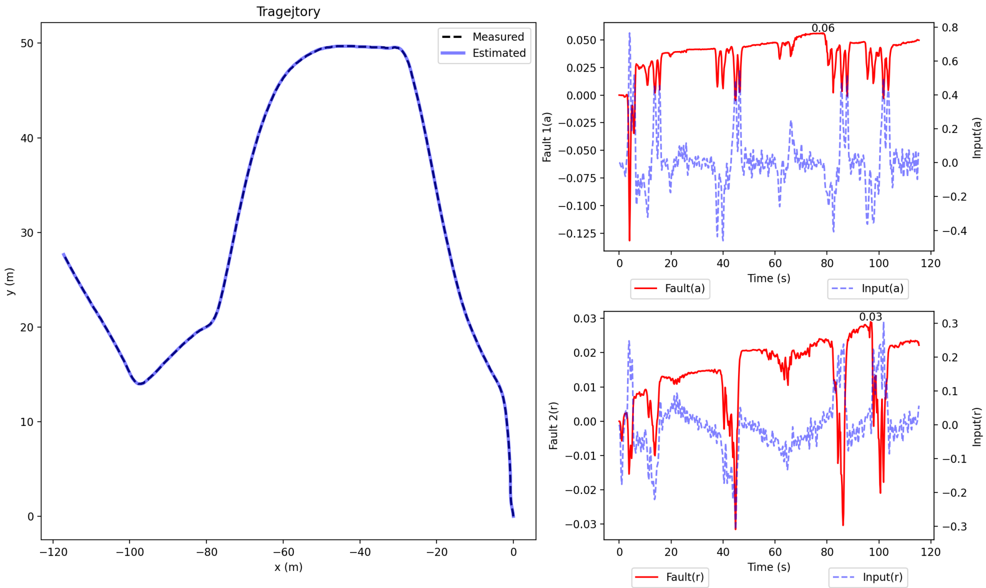


(b) Actuator faults during S-maneuver

Figure 4.11: Actuator faults for different propellers on each side

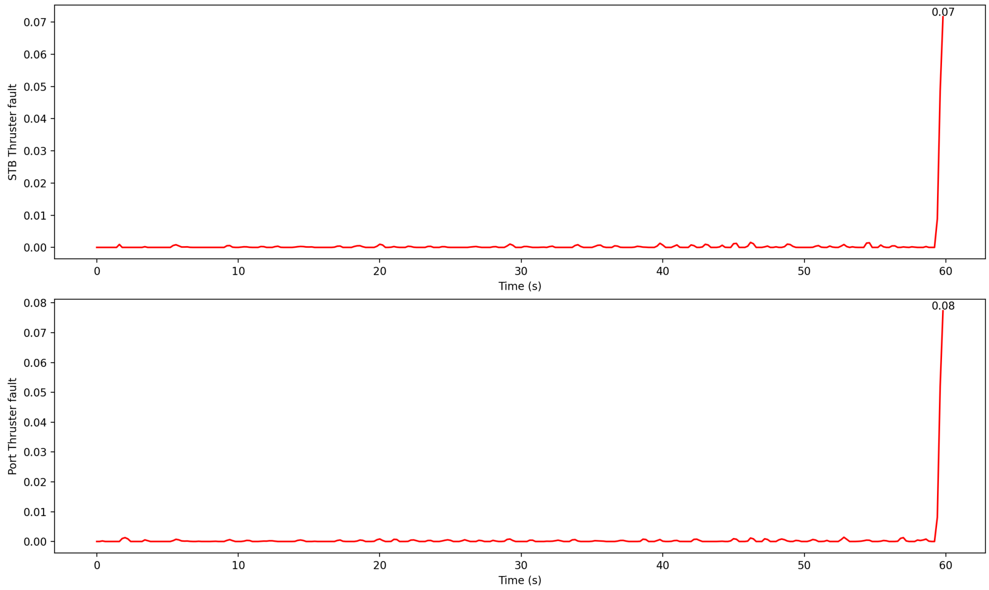


(a) Position and fault parameter estimates during straight line maneuver

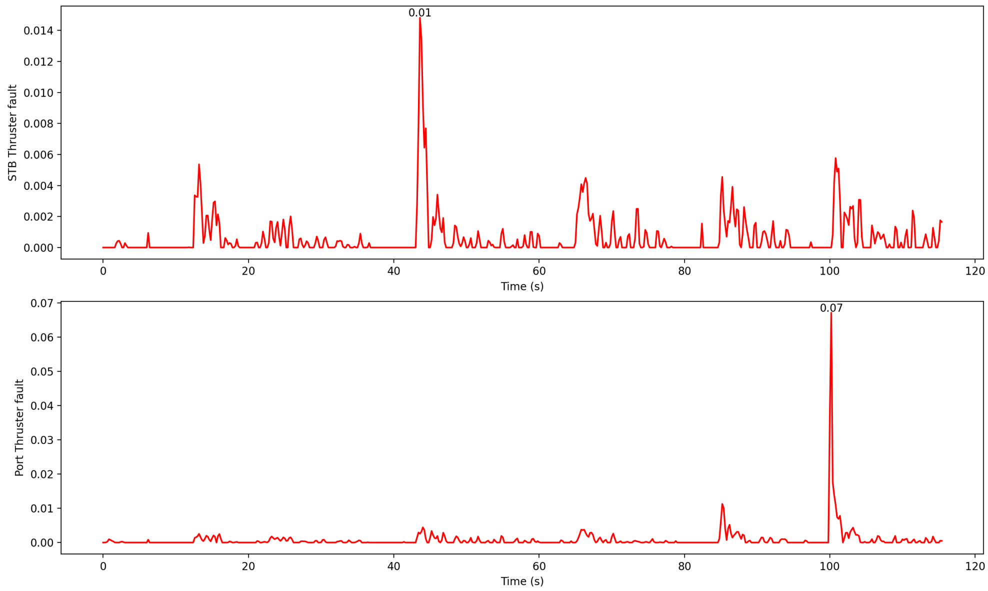


(b) Position and fault parameter estimates during S-maneuver

Figure 4.12: Test 3 with new propellers

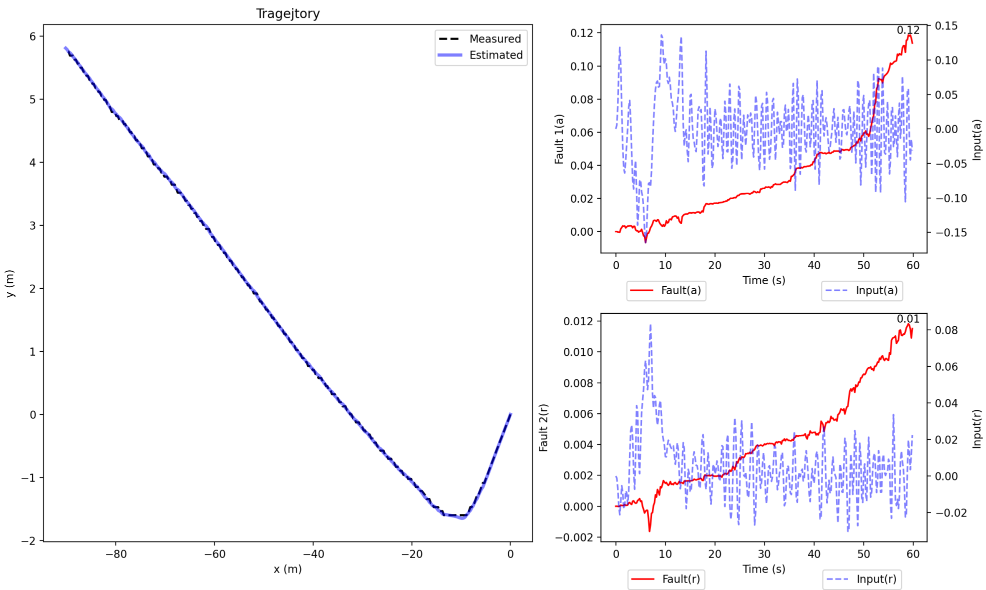


(a) Actuator faults during straight line maneuver

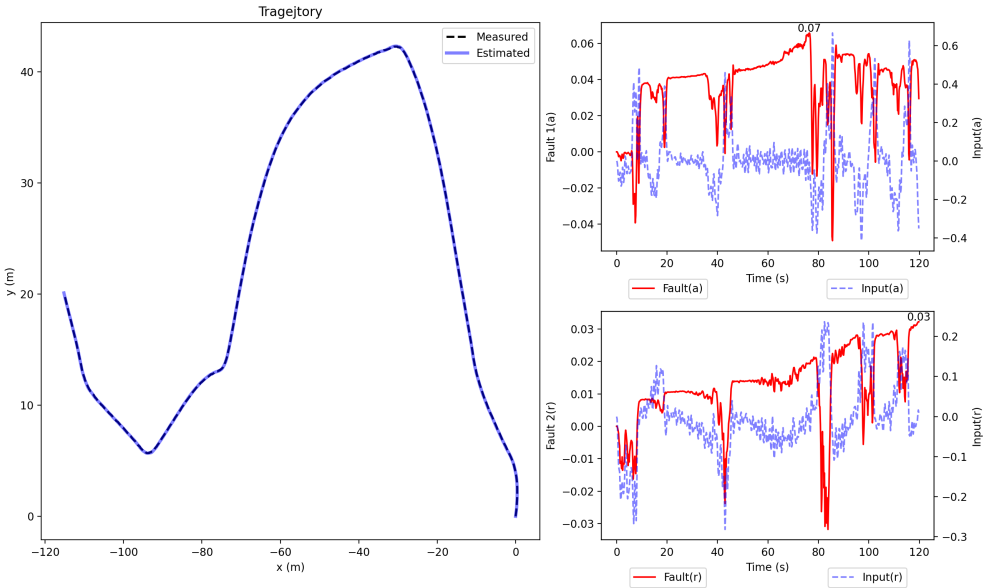


(b) Actuator faults during S-maneuver

Figure 4.13: Actuator faults for Otter with new propeller on both sides

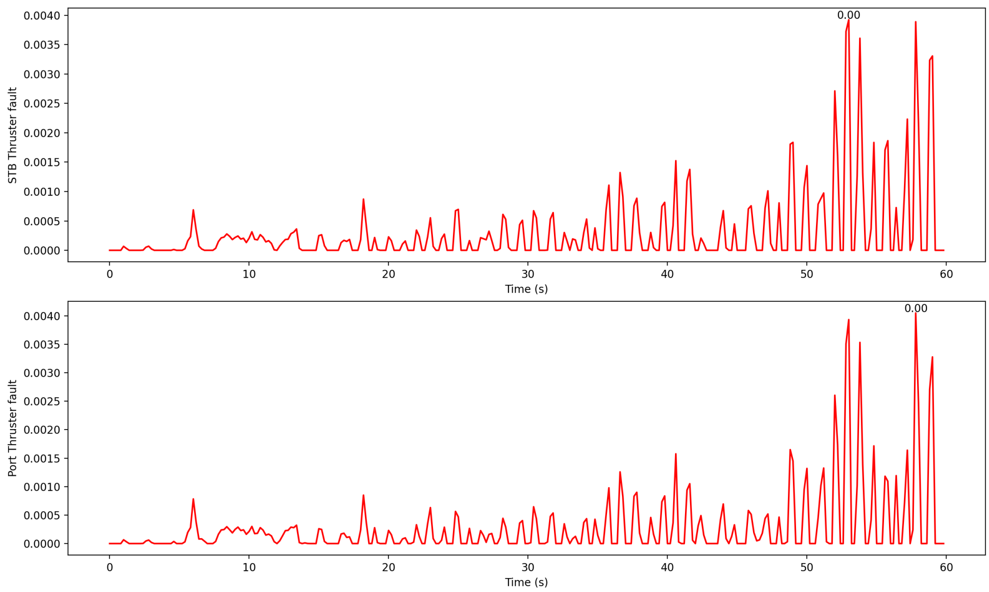


(a) Position(left) and fault parameter(right) estimates during straight line maneuver

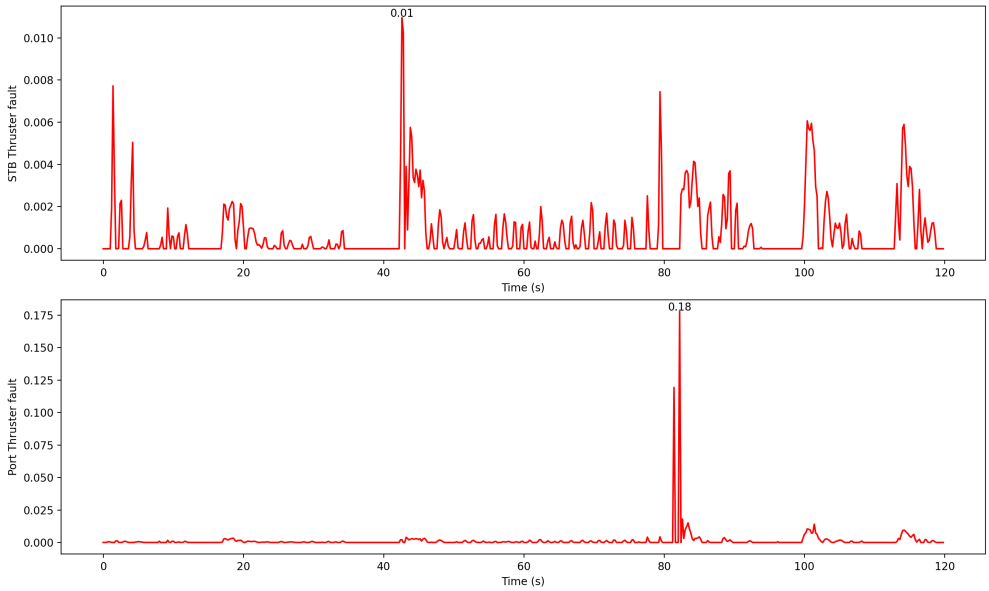


(b) Position(left) and fault(right) parameter estimates during S-maneuver

Figure 4.14: Test 4 with new and broken propellers



(a) Actuator faults during straight line maneuver



(b) Actuator faults during S-maneuver

Figure 4.15: Actuator faults for Otter with new propeller on both sides

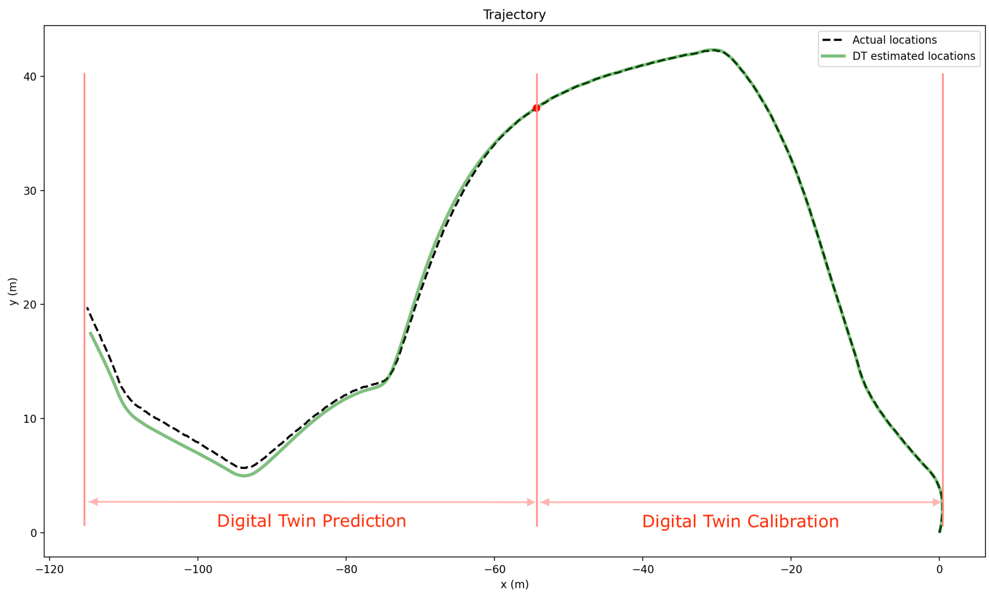


Figure 4.16: Calibration and prediction phases of the Digital Twin

Chapter 5

Conclusion

Given the emerging interest in digital twins and autonomous systems, this thesis aims to develop a digital twin for an Unmanned Surface Vehicle. We explore the use of 'Probabilistic Graphical Models' as a general mathematical framework for digital twin applications and demonstrate the capabilities of the digital twin by performing 'Actuator Fault Diagnosis' and identifying a broken propeller fault in Otter's port thruster. The research objectives(1.4.1) selected in Chapter 1 and their fulfillment is listed again below

Objective 1: Constructing a unifying mathematical model for digital twins development at scale.

Proposed solution: The thesis explores the use of Dynamic Bayesian Networks, a version of 'Probabilistic Graphical Models' as a generalized mathematical framework for developing digital twin applications at scale. This framework offers are unique perspective on digital twin implementation while being flexible enough to modified for customisation.

Objective 2: Utilizing the fore-mentioned mathematical framework and performing actuator fault diagnosis on Otter's actuators.

Proposed solution: As a concrete example, 'Actuator Fault Diagnosis' is found to be an interesting application feature to demonstrate the capabilities of the digital twin. Adaptive Extended Kalman Filter based actuator fault diagnosis method is implemented using the kinematic model of the vehicle and state measurements (Location,SOG, COG) from the GNSS sensor on the vehicle. Since, Kalman filters can be viewed as dynamic Bayesian Networks, the method fits well into the Probabilistic Graphical Model framework discussed previously.With AEKF algorithm at the core along with the vehicle's 3-DOF kinetic model, we identify the fault in port thruster with a broken propeller as 18%. This proves to be an inexpensive and easy to implement method compared to the existing data-driven methods found in literature. However, one limitation with the method is that it does not indicate the type of fault (for example, broken propeller, stuck propeller, stator faults etc.,)

Objective 3: Making meaningful predictions on the Otter's behaviour based on the updated digital twin that is cognizant of actuator faults.
Proposed solution: In order to make use of the updated digital twin that is aware of the actuator fault, we run state estimation using with pre-computed fault parameter values from the calibration step to predict the future states of Otter over a prediction interval. We observe that from the RMSE error that Digital Twin prediction are reasonable given the length of the prediction horizon

5.1 Recommendations

- One immediate improvement of the existing work is the real-time implementation of the methods discussed here. The framework of graphical models is known to be especially suitable real-time digital twin update.
- Also, the twin can be made more interactive by creating a 3D visualization and a game-like user-interface using Unity 3D. Both the above points were part of my initial scope but couldn't be completed as conducting experiments with Otter and working with real data was itself time-consuming.
- Using input acceleration and yaw-rate calculated from the state-measurements eliminates the need to collect more data, but may also result in a bias in the results. This can be handled by independent measurements of acceleration, yaw-rate and yaw acceleration using an Inertial Measurement Unit (IMU).
- Other improvement in the test conditions are smaller sample time than the current 0.2s and a longer run time. Both of them will be improve the convergence of the fault estimation by giving it more iterations.
- Though we estimate the faults in inputs and actuators, the uncertainty associated with our estimation is left unaccounted due to time constraints.This is a important factor that can be considered by taking advantage of the probabilistic nature of the graphical model framework.

Bibliography

[1] A. Rasheed, O. San and T. Kvamsdaal, 'Digital twin: Values,challenges and enablers from a modelling perspective,' *IEEE Access*, vol. 8, pp. 21 980–22 012, 2020.

[2] M.Grieves, *Digital Twin: Manufacturing Excellence Through Virtual Factory Replication*.

[3] 'Kognitwin.' (), [Online]. Available: `https://www.kongsberg.com/digital/solutions/kognitwin-energy/`.

[4] 'Virtual singapore.' (), [Online]. Available: `https://www.nrf.gov.sg/programmes/virtual-singapore`.

[5] Z. Liu, Y. Zhang, X. Yu and C. Yuan, 'Unmanned surface vehicles: An overview of developments and challenges,' *Annual Reviews in Control*, vol. 41, pp. 71–93, 2016.

[6] B. R. Barricelli, E. Casiraghi and D. Fogli, 'A survey on digital twin: Definitions, characteristics, applications, and design implications,' *IEEE access*, vol. 7, pp. 167 653–167 671, 2019.

[7] M. Liu, S. Fang, H. Dong and C. Xu, 'Review of digital twin about concepts, technologies, and industrial applications,' *Journal of Manufacturing Systems*, vol. 58, pp. 346–361, 2021.

[8] P. Y. Major, G. Li, H. Zhang and H. P. Hildre, 'Real-time digital twin of research vessel for remote monitoring,' *Proceedings of 35th European Council for Modelling and Simulation*, 2021.

[9] A. Danielsen-Haces, 'Digital twin development, condition monitoring and simulation comparison for the revolt autonomous model ship,' Master's thesis, NTNU, 2018.

[10] Q. Zhang, 'Adaptive kalman filter for actuator fault diagnosis,' *Automatica*, vol. 93, pp. 333–342, 2018.

[11] M. Skriver, J. Helck and A. Hasan, 'Adaptive extended kalman filter for actuator fault diagnosis,' in *2019 4th International Conference on System Reliability and Safety (ICSRS)*, IEEE, 2019, pp. 339–344.

[12] A. Alessandri, M. Caccia and G. Veruggio, 'Fault detection of actuator faults in unmanned underwater vehicles,' *Control Engineering Practice*, vol. 7, no. 3, pp. 357–368, 1999.

[13] N. Y. Ko, G. Song, H. T. Choi and J. Sur, 'Fault detection and diagnosis of sensors and actuators for unmanned surface vehicles,' in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, IEEE, 2021, pp. 1451–1453.

[14] Z. Zhou, M. Zhong and Y. Wang, 'Fault diagnosis observer and fault-tolerant control design for unmanned surface vehicles in network environments,' *IEEE Access*, vol. 7, pp. 173 694–173 702, 2019.

[15] W. Abed, S. Sharma and R. Sutton, 'Neural network fault diagnosis of a trolling motor based on feature reduction techniques for an unmanned surface vehicle,' *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 229, no. 8, pp. 738–750, 2015.

[16] J. Luo, Y. Shi and L. Xie, 'An unsupervised learning based simplification on ship motion model and its verification,' in *2019 34rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, IEEE, 2019, pp. 52–57.

[17] H. A. Uehara Sasaki, A. S. S. Ianagui, P. Cardozo de Mello and E. A. Tannuri, 'Digital twin of a maneuvering ship: Real-time estimation of derivatives and resistance coefficient based on motion sensor,' in *International Conference on Offshore Mechanics and Arctic Engineering*, American Society of Mechanical Engineers, vol. 85161, 2021, V006T06A021.

[18] M. G. Kapteyn, J. V. Pretorius and K. E. Willcox, 'A probabilistic graphical model foundation for enabling predictive digital twins at scale,' *Nature Computational Science*, vol. 1, no. 5, pp. 337–347, 2021.

[19] 'About systems engineering.' (), [Online]. Available: `https://www.incose.org/about-systems-engineering`. (accessed: 17.05.2022).

[20] C. M. et al, 'Computational modelling for decision-making: Where, why, what, who and how,' *R. Soc. open sci*, vol. 5, no. 172096, pp. 39–41, 2018.

[21] A. Wunderlich, K. Booth and E. Santi, 'Hybrid analytical and data-driven modeling techniques for digital twin applications,' in *2021 IEEE Electric Ship Technologies Symposium (ESTS)*, IEEE, 2021, pp. 1–7.

[22] A. D. E. I. Committee, 'Digital twin: Definition & value,' *(American Institure of Aeronautics and Astronautics (AIAA) and Aerospace Industries Association (AIA), 2020)*,

[23] L. Wright and S. Davidson, 'How to tell the difference between a model and a digital twin,' *Advanced Modeling and Simulation in Engineering Sciences*, vol. 7, no. 1, pp. 1–13, 2020.

[24] R. Eramo, F. Bordeleau, B. Combemale, M. van Den Brand, M. Wimmer and A. Wortmann, 'Conceptualizing digital twins,' *IEEE Software*, 2021.

[25] Y.-W. Lin, T. L. E. Tang and C. J. Spanos, 'Hybrid approach for digital twins in the built environment,' in *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, 2021, pp. 450–457.

[26] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[27] K. Murphy, 'An introduction to graphical models,' *Rap. tech*, vol. 96, pp. 1–19, 2001.

[28] S. Simani, C. Fantuzzi and R. J. Patton, 'Model-based fault diagnosis techniques,' in *Model-based Fault Diagnosis in Dynamic Systems Using Identification Techniques*, Springer, 2003, pp. 19–60.

[29] R. J. Patton, P. M. Frank and R. N. Clark, *Issues of fault diagnosis for dynamic systems*. Springer Science & Business Media, 2013.

[30] R. Isermann, *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media, 2005.

[31] T. I. Fossen, *Guidance and control of ocean vehicles*. John Wiley & Sons, 2002.

[32] S. of Naval Architects, M. E. (Technical and R. C. H. Subcommittee, *Nomenclature for Treating the Motion of a Submerged Body Through a Fluid: Report of the American Towing Tank Conference*, ser. Technical and research bulletin. Society of Naval Architects and Marine Engineers, 1950. [Online]. Available: `https://books.google.no/books?id=VqNFGwAACAAJ`.

[33] T. Fossen and T. Perez, Marine Systems Simulator (MSS) URL: 2004. [Online]. Available: `https://github.com/cybergalactic/MSS`.

[34] G. Antonelli, ‘A survey of fault detection/tolerance strategies for auvs and rovs,’ in *Fault diagnosis and fault tolerance for mechatronic systems: Recent advances*, Springer, 2003, pp. 109–127.

[35] S. Fossen and T. I. Fossen, ‘Exogenous kalman filter (xkf) for visualization and motion prediction of ships using live automatic identification systems (ais) data,’ 2018.

[36] J. Farrell, *Aided navigation: GPS with high rate sensors*. McGraw-Hill, Inc., 2008.

Appendix A

Additional Material

Otter's 3D Model

Below is a 3D model (Fig.??) of Otter that can be used in an interactive Digital Twin application. This adds a visualization dimension to the digital twin where the vehicle can be perceived better.

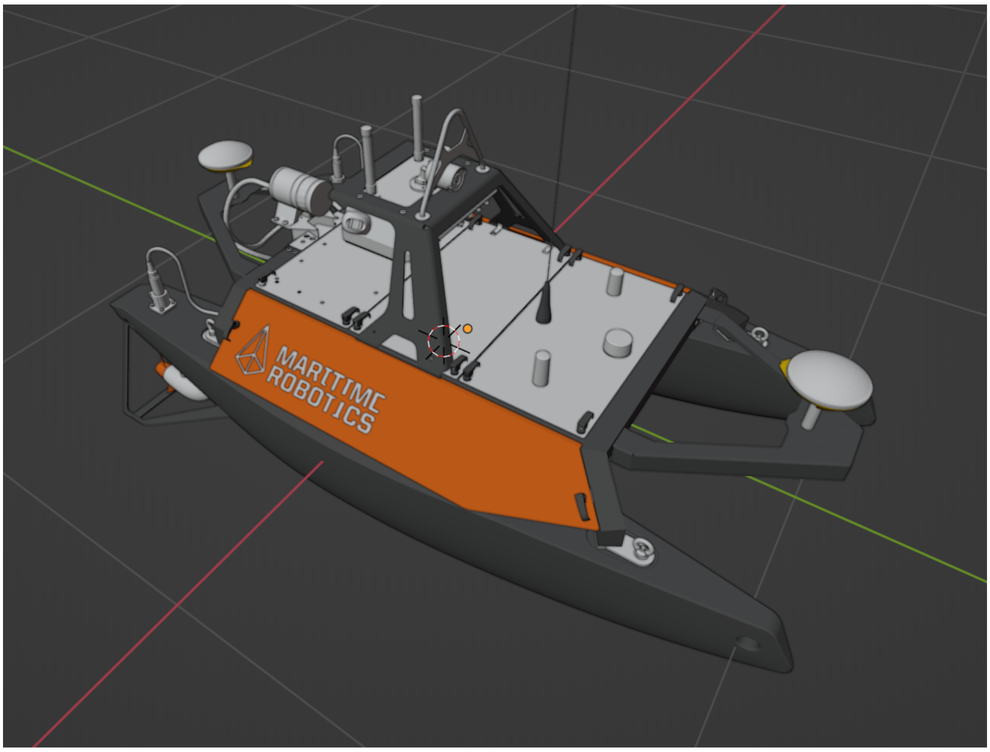


Figure A.1: 3D model of Otter

Digital Twin UI

A sample application for Otter's digital twin is in making in Unity 3D. Below is the snapshot (see Fig.A.2) of the user-interface along with the 3D map of the area surrounding Otter. This 3D map is imported to Unity using Mapbox API.

Figure A.2: Unity Application for Otter's Digital Twin