A Hybrid Nonlinear Model Predictive Control and Recurrent Neural Networks for Fault-Tolerant Control of an Autonomous Underwater Vehicle

Mahsa Khoshab

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

December 2017

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Mahsa Khoshab

Entitled: A Hybrid Nonlinear Model Predictive Control and Recurrent Neural Networks for Fault-Tolerant Control of an Autonomous Underwater Vehicle

and submitted in partial fulfilment of the requirements for the degree of

Master of Applied Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Dr. Rabindranath Raut, Chair

_____ Dr. Mehdi Hojjati, External Examiner

_____ Dr. Shahin Hashtrudi Zad, Examiner

_____ Dr. Khashayar Khorasani, Supervisor

Approved by: _____

Dr. W. E. Lynch, Chair
Department of Electrical and Computer Engineering

_____ 20 _____

Dr. A. Asif
Dean, Faculty of Engineering and Computer Science

ABSTRACT

A Hybrid Nonlinear Model Predictive Control and Recurrent Neural Networks for Fault-Tolerant Control of an Autonomous Underwater Vehicle

Mahsa Khoshab

The operation of Autonomous Unmanned Vehicles (AUVs) that is used for environment protection, risk evaluation and plan determination for emergency, are among the most important and challenging problems. An area that has received much attention for use of AUVs is in underwater applications where much work remains to be done to equip AUVs with systems to steer them accurately and reliably in harsh marine environments. Design of control strategies for AUVs is very challenging as compared to other systems due to their operational environment (ocean). Particularly when hydrodynamic parameters uncertainties are to be integrated into both the controller design as well as AUVs nonlinear dynamics. On the other hand, AUVs like all other mechanical systems are prone to faults. Dealing effectively with faulty situations for mechanical systems is an important consideration since faults can result in abnormal operation or even a failure. Hence, fault-tolerant and fault-accommodating methods in the controller design are among active research topics for maintaining the reliability of complex AUV control systems.

The objective of this thesis is to develop a nonlinear Model Predictive Control (MPC) that requires solving an online Quadratic Programming (QP) problem by using a Recurrent Neural Network (RNN). Also, an Extended Kalman Filter (EKF) is integrated with the developed scheme to provide the MPC algorithm with the system states estimates as well as a nonlinear prediction. This hybrid control approach utilizes both the mathematical model of the system as well as the adaptive nature of the intelligent technique through neural networks. The reason behind the selection of MPC is to benefit from its main capability in optimization within the current time slots while taking future time slots into consideration. The proposed control method is integrated with EKF which is an appropriate

method for state estimation and data reconciliation of nonlinear systems. In order to address the high performance runtime cost of solving the MPC problem (formulated as a quadratic programming problem), an RNN is developed that has a low model complexity as well as good performance in real-time implementation. The proposed method is first developed to control an AUV following a desired trajectory. Since the problem of trajectory tracking and path following of AUVs exhibit nonlinear behavior, the effectiveness of the developed MPC-RNN algorithm is studied in comparison with two other control system methods, namely the linear MPC using Kalman Filter (KF) and the conventional nonlinear MPC using the EKF.

In order to guarantee the fault-tolerant features of our proposed control method when faced with severe actuator faults, the developed MPC-RNN scheme is integrated with a dual Extended Kalman Filter that is used for a combined estimation of AUV states and parameters. The actuator faults are defined as the system parameters that are to be estimated online by the dual-EKF. Therefore, the developed Active Fault-Tolerant Control (AFTC) strategy is then applied to an AUV faced with loss of effectiveness (LOE) actuator fault scenarios while following a trajectory. Analysis and discussions regarding the comparison of the proposed AFTC method with Fault-Tolerant Nonlinear Model Predictive Control (FT-NMPC) algorithm are presented in this work. The proposed approach to AFTC exploits the advantages of the MPC-RNN algorithm properties as well as accounting explicitly for severe control actuator faults in the nonlinear AUV model with uncertainties that are formulated by the MPC.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Dr. Khorasani for his patient guidance, support and his immense knowledge. This thesis could not be accomplished without his guidance and insightful comments.

I would also like to thank my colleagues at Concordia University: Esmaeil AliZadeh, Hossein Khodadadi, Dr. Amir Baniamerian, Maria Enayat, Yanyan Shen, Maryam Abdollahi, Amin Salar, and Dr. Bahar Pourbabaee for all the stimulating discussions and exchanges, as well as for all the fun and nice time we have had throughout these years. Moreover, I would also like to thank my beloved uncle, Mohsen Khoshab, and his family for their support throughout recent years that I moved to Montreal.

Last but not the least, I would like to thank my beloved ones in particular my parents, my brothers, Dr. Masih Khoshab and Dr. Mohammad Khoshab for their unconditional love and inspiration throughout my life. None of this would have been possible without their encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Oceans are occupying approximately 71 percent of surface of the earth and the collection of ocean data by survey for studies and development is indispensable. Therefore, the research and contribution in underwater vehicles have become an important field for researchers since almost six decades [14].

There are basically three types of underwater vehicles: Remotely Operated Vehicle (ROV), Unmannd Underwater Vehicles (UUV) and Autonomous Underwater Vehicle (AUV) [13]. In 1957, Stan Murphy, Bob Francois and later on Terry Ewart, developed the first AUV, Self-Propelled Underwater Research Vehicle (SPURV), at Applied Physics Laboratory in the University of Washington, which was used to study diffusion, acoustic transmission, and submarine wakes [15]. Most of early AUVs that had been developed were large, inefficient, and expensive.

While the ROVs were becoming more mature in early 1980s, the AUV technology was essentially in its infancy. In fact, ROVs have attributes of a brain (the human operator) attached via a long nervous system (the umbilical cable) and brawn (hydraulic power),

which is provided by heavy-duty electro-hydraulic power systems to thrusters, tools and manipulators. AUVs are required to carry their brain and brawn with them, a requirement that in early 1980s left them waiting for advances in computer technology and energy storage. After more than twenty years of continuous development, AUVs can operate without the need of constant monitoring and supervision from a human operator [16]. Therefore, among all types of underwater vehicles, AUVs are used across a wide range of mission scenarios and from an increasingly diverse set of operators [17].

Since then, with the development of activities in ocean, AUVs have been widely used in many fields such as military reconnaissance and mine countermeasures, region surveillance, search and rescue, profiling the water column for scientific measurements of conductivity, temperature, density, sound speed and other acoustic measures [13]. AUVs present an ever expanding range of applications that enhance human capabilities and mitigate human risks.

Meanwhile, oil and gas industries highly benefit from AUVs technology for the inspection of the pipelines and other marine related tasks [15]. Therefore, underwater vehicles have been developed that range from Robo-Lobster and Robo-Tuna to the giant Defense Advanced Research Projects Agency's (DARPA) UUVs. The offshore oil industry looks at AUVs, such as the Hugin used by Norways Statoil, to lower the cost of operations in many areas. Japan is developing an AUV to reach the depths of Mariana Trench, while Jet Propulsion Laboratory (JPL) is developing AUVs to bore through the ice and investigate the seas of other planets and moons [16]. The Urashima [18], for which JAMSTEC started actual ocean experiments in Fiscal 2000, was chosen for the demonstration of simulator application. Nowadays, many countries either operate AUVs or they are at the stage of developing one [19]. For example, the WHOIs ABE AUV [20] has been operational for the accurate terrain mapping and sea floor hydrothermal vents investigation for more than ten years. It was then replaced with a $4500m$ leveled SENTRY AUV which has better maneuverability [21].

In [13], a detailed classification of survey AUVs are discussed. In this thesis, a small survey AUV (Remus [9]) is used which has been developed for specific applications including coastal monitoring, turbulence studies, environmental samplings in support of coastal modeling programs, hydrography, and support of special warfare, so far. REMUS is known as an accomplishment in this industry since this type of AUV has been used for many researches and experiments; i.e. REMUS 6000 is the maximum depth AUV of REMUS series developed by MBARI [22], which has the capability of sea floor terrain mapping .

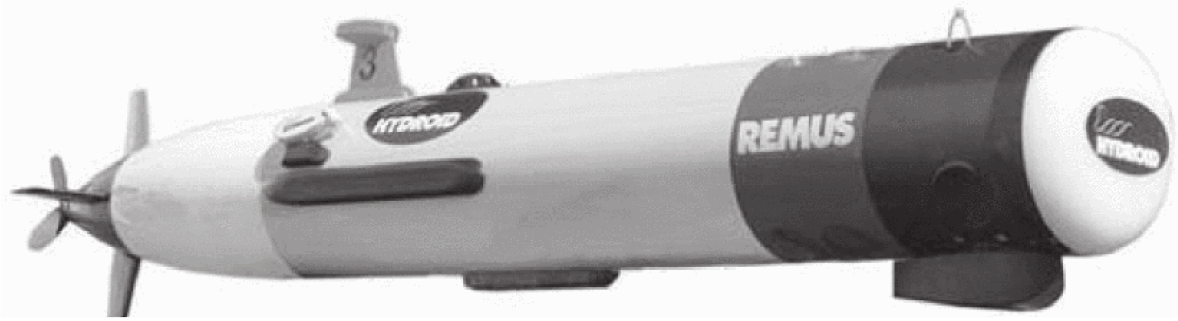


Figure 1.1: A Remus AUV [3].

A tremendous progress in development of marine technologies was seen in the last decades providing scientists with advanced ocean exploration methods. Recent advances in marine robotics, sensors, computers, communications, and information systems are used to develop sophisticated technologies leading to safer, faster and more efficient ways of exploring the ocean frontier. As a part of this trend, there has been much interest worldwide in the development of AUVs that are capable of roaming the oceans freely and collecting data at the surface of ocean and underwater on an unprecedented scale. Various mission involving challenging tasks without close supervision of human operators envisioned a call for the control of AUVs. Thus, it is important to push the development of methods for reliable vehicle and mission control of AUVs.

1.2 Literature Review

This section aims to briefly review the control of AUVs and to bring some background about the importance of fault tolerant control systems. In addition, some preliminary topics about computational intelligence in control systems are presented.

1.2.1 Control of AUVs

Generally, what makes it difficult to control AUVs are: the highly nonlinear time-varying dynamic behavior of robots, uncertainties in hydrodynamic coefficients and disturbances by ocean currents. Also, when the manipulators are attached to the AUV the higher order and redundant structure as well as the changes in the centers of the gravity and buoyancy (due to the manipulator motion) disturb the main body of AUVs. Therefore, it is complicated to fine-tune control gains during an operation in water. It is highly desirable to have a control system that has a self-tuning ability when the control performance degrades during the operation due to changes in dynamics of the AUV and its environment.

Yoerger and Slotine [23] have proposed a sliding mode controller for an underwater vehicle to control its trajectory. They have investigated the effects of uncertainty of the hydrodynamic coefficients and negligence of cross-coupling terms. Healey and Lienard [24] have used sliding mode methods for controlling underwater vehicles. The system in [24] is decomposed into non-interacting (or lightly interacting) subsystems, and certain key motion equations are combined according to separate functions of steering, diving, and speed control.

In [25], Nakamura and Savant have proposed a nonlinear tracking control of a 4 degree of freedom (surge, roll, pitch and yaw) AUV considering its kinematic motion . The work [25] uses the nonholonomic nature of the system without considering system dynamics. Goheen *et al.* [26] have proposed multi-variable self-tuning controllers acting as an autopilot

for underwater vehicles to overcome model uncertainties while performing auto-positioning and station-keeping. Later, Choi and Yuh [27] have developed a novel Multiple Input Multiple Output (MIMO) adaptive controller using bound estimation for underwater robotic systems. They have implemented the developed control system on an AUV, namely the Omni Directional Intelligent Navigator (ODIN).

A hybrid adaptive control (suggesting that the procedure is a mixture of continuous and discrete operations) of an AUV was investigated by Tabaii *et al.* [28]. The system in [28] was simulated in the continuous time domain while control and identification sections were discrete-time. Yuh [14] has proposed a neural network control system using a recursive adaptation algorithm with a critic function (a reinforced learning approach). The special feature of this controller is that the central system adjusts itself directly and in an online manner without an explicit model of vehicle dynamics.

Ishii *et al.* [29] have proposed a neural network based control method called Self-Organizing Neural-net-Controller System (SONCS) for AUVs and also examined its effectiveness through another AUV called Twin-Burger. In [29], a quick controller adaptation method called Imaginary Training is used to improve the time-consuming adaptation process of SONCS. Tsukamoto *et al.* [30] practically implemented four model-free control systems for the position and velocity control of a single thruster system that are: online neural net controller, off-line neural net controller, fuzzy control, and the non-regressors based adaptive control . In [30], the off-line neural controller utilizes Intel i80170 Electrically Trainable Artificial Neural Network (ETANN) chips.

Since there are many challenging engineering problems for vehicles with manipulators, unlike many underwater remotely operated vehicles (ROVs), most AUVs are not equipped with mechanical manipulators. In fact, for a large robot, effects of arm motion on the main body may be negligible, and even the main body and arm can be considered as two separate systems with different bandwidths. Unlike large robots, coupling effects of the main body

and arm are significant in case of a small robot and must be taken into account in the overall control system design. With the arm attached to the vehicle, the overall system becomes a multi-rigid body. The vehicle's main body continuously moves in water, and consequently high arm control performance, in terms of speed and accuracy, requires highly accurate information about the vehicle's position and velocity. Therefore, most commercial sensors for vehicle's position and velocity do not meet the accuracy requirements of the arm control. Hence there are few papers about the coordinated motion of the vehicle and manipulator [31–35].

Mahesh *et al.* in [31] have developed a coordinated control scheme, using a discrete-time approximation of the dynamic model of underwater robotic systems, which controls the vehicle and manipulator simultaneously and also compensates for end-effector errors resulting from motion of the vehicle. McLain *et al.* [32] have conducted experiments at the Monterey Bay Aquarium Research Institute (MBARI) using the OTTER vehicle and have shown that dynamic interactions between robot arm and vehicle can be very significant. McLain *et al.* [32] pointed out that coordinated motion control strategy along with an accurate model of the arm/vehicle hydrodynamic interaction forces would impose the station-keeping capability and end-effector accuracy.

Taren *et al.* [33] investigated a nonlinear model based control scheme that simultaneously controls the position and orientation of the vehicle and manipulator. Canudas-de-Wit *et al.* in [35] have designed a robust nonlinear control for a vehicle/arm system to compensate for the coupling effects due to an onboard robot arm . Different bandwidth characteristics of the composite vehicle-manipulator dynamics are used [35] as a basis for the controller design via singular perturbation theory. Moreover, both the robust controller as well as the partial linearized controller proposed in [35] achieve similar performance even in the presence of saturation fault. Antonelli and Chiaverini [34] proposed a task-priority based redundancy resolution scheme for kinematic control of an underwater vehicle-manipulator

system by suitably using the null space vector.

Motion Control of AUVs

There is a considerable interest in the development of advanced methods for motion control of marine vehicles (including surface and underwater vehicles) in the presence of unknown ocean currents, wave action, and vehicle model uncertainty. The most relevant AUV problems are as follows:

- vertical and horizontal plane control,
- pose (position and attitude) control,
- trajectory tracking and path following control.

Vertical and Horizontal Plane Control

In a vast number of mission scenarios, underwater vehicles are required to maneuver in vertical and horizontal planes while tracking a desired speed profile bounded away from zero. Examples include heading control in the horizontal plane and depth or altitude control (above the seabed) in the vertical plane [10, 36, 37]. More challenging applications require depth control close to the sea surface while in the presence of strong wave action [38]. Examples of this type of control that is required for both streamlined and bluff bodies are the Infante AUV and the Sirene AUV, respectively [39]. The first class of bodies has a preferred direction of motion and the control objective is usually accomplished by resorting to simplified dynamic models of motion that is obtained by linearizing their nonlinear dynamics about trimming conditions. The second class of bodies does not have a preferred direction of motion leading to a more difficult control strategy that requires more complex nonlinear dynamic models of motion. The problem of control in the horizontal plane is also relevant in the case of autonomous surface craft such as the Delfim or Caravela vessels [39].

Pose Control

A completely different class of problems arises when an underwater vehicle must be steered to a final target point with a desired orientation. This situation calls for the development of controllers to manoeuvre the vehicle at speeds around zero. The problem is especially challenging when the number of vehicle actuators is fewer than its degrees of freedom, as in case of the Sirene AUV [40]. In this situation, theoretical limitations arising from the fact that the vehicles are non-holonomic [41], and hence discontinuous, hybrid, or even time-varying feedback control laws should be used.

Trajectory Tracking and Path Following

Trajectory tracking refers to the problem of tracking a time-parameterized reference curve in two or three-dimensional space, usually by a marine vehicle. Simply, one requests that the vehicle to be at assigned spatial coordinates at particular time instants. This requires that the position of the vehicle to be controlled with respect to an inertial frame. In the case of an AUV faced with strong currents, trajectory tracking may lead to a situation where the vehicle surfaces stall and also the control authority is drastically reduced.

Furthermore, the trajectory tracking control often leads to unpredictable vehicle motions in an attempt to meet stringent spatial requirements as well as requiring considerable actuator activity. Both problems are slightly attenuated when temporal constraints are lifted, and hence leading to the problem of path following. Path following is the task of forcing a vehicle to converge to and follow a desired spatial path, without any temporal specifications [10]. In some missions the vehicle is still required to track a desired temporal speed profile. For instance, this objective occurs for example when an autonomous surface vessel must cover a certain area by performing a lawn mowing maneuver along desired trajectories with great accuracy, at speeds determined by the end-user.

The underlying assumption in path following control is that vehicles forward speed tracks in a the desired speed profile bound, while the controller directs the vehicle's orientation to move toward the path. Typically, smoother convergence to the path is achieved when path following strategies are used instead of trajectory tracking control laws, and the control signals are less likely to be pushed to the saturation. This interesting circle of ideas opens the door to more sophisticated strategies that naturally combine some of the attributes of trajectory tracking and path following, as first suggested in the pioneering work of Hauser and Hindman [42] and more recently pursued in works [43] and [44].

Nonlinear Path Following

The majority of AUV controller designs take a classic strategy followed by linear control techniques such as proportional integral derivative (PID) controller. However, AUVs are typically small multi-purpose underwater vehicles that are working around numerous operating points, which makes the classical linear control theory to be not applicable. Since nonlinear effects of hydrodynamic damping and lift, added mass, Coriolis and centripetal forces may produce degraded performances, the use of advanced control techniques in various AUV applications is drawing researchers attentions.

Path following requires the vehicle to reach and follow a desired path generally without any time constraint. This task is accomplished through controlling the forward speed and also through directing the vehicle's orientation towards its desired path. The task of path following is considered to be solved when the designed controller guarantees asymptotic convergence to the path. Some works have addressed the problem of path-following control for a nonlinear systems [45–48]. A new methodology has been proposed in [49] for the design of path-following systems within an autonomous marine craft in the presence of constant but unknown currents.

The work [49] explains the key ideas behind the development of the nonlinear algorithm. In [49], it is assumed that the main body-axis of a hypothetical vehicle is aligned with the net velocity vector. A nonlinear kinematic controller is then derived for the vehicle to steer it to a reference path in the presence of a constant and known ocean current. The work [49] followed this by the design of a linear estimator for the current that yields exponential asymptotic stability of the estimation errors to zero. The nonlinear control law is then modified to use the estimated values of the current instead of the real ones. In [49], it is assumed that the position and attitude of the marine craft, as well as its angle of side-slip, are accessible for measurement. Asymptotic convergence to the reference path is proven for the overall control system. Finally, the use of nonlinear dynamic inversions and applying backstepping technique are explained.

1.2.2 The Necessity of Fault Detection, Isolation and Reconfiguration in AUV Systems

A large amount of attention has been paid to the problem of Fault Diagnosis (FD) of AUVs to improve the reliability of these systems. Because of the communication delays in depth, AUVs should be able to diagnose the failure and decide how to reconfigure their control commands. Therefore, the early detection of the malfunctions and faults as well as their compensation are crucial both for the maintenance and for the mission reliability of these vehicles.

Faults are defined as any deviations from the normal behavior in the plant or its other relevant instruments. In fact, a fault detection system makes a binary decision that whether the system works in a normal condition or a fault happed to the monitored system. Fault isolation is the determination of location of a fault in the monitored system, whereas the fault identification identifies the type of the fault. A system with fault detection, isolation and

identification capabilities is known as Fault Detection, Isolation and Identification (FDII) system.

In case of Fault Recovery (FR), AUV missions generally use pre-scripted plans but these missions do not scale well with partially known or unknown environments. Pre-scripted plans, have intrinsically limited capability of exploiting observations that can be used as a basis for decision-making. Unexpected changes in the environment or the vehicle state, sensor limitations and hardware faults often affect mission performance. In addition, high-level priority changes can occur in real-time and accordingly, vehicles should be able to accommodate and act upon them. Fault detection, online mission planning and knowledge acquisition approaches are necessary but hardly sufficient. To promote persistent autonomy and successful mission execution, system control approaches should be backed by fault recovery capabilities and adaption on both the mission planning and execution levels [50].

The integration of FDI and recovery module is an important consideration in developing an FDIR methodology. According to [51], this integration divides to three subsections: 1) the open-loop approach in which the FDI module has no affect on the control function. Therefore, the control system effects the FDI operation through the residual signals. 2) Establishing a relation between the FDI information and the controller. In this way, once the fault is detected and isolated, the recovery system performs the fault accommodation task. The controller reconfiguration scheme performs the recovery task without changing the control structure. 3) A combination of control design and fault estimation resulted in a robust FDI approach, which imposes extra design constraints on the system. The work [51] explains how this approach results in a trade-off between the performance and robustness.

1.2.3 Fault-Tolerant Control of AUVs

The hazardous ocean environment presents many challenging problems in the event of system failures for AUVs. For any major failure of robots subsystems, the robot should rise to the surface and signal for retrieval. However, for any tolerable failure, the robot should be able to adjust for the failure and complete the assigned task. Therefore, an efficient and effective fault tolerant system becomes imperative for AUVs.

A fault-tolerant system consists of three steps: fault detection, fault isolation, and fault accommodation [52]. The fault detection process is a high-level function that monitors the overall systems (both hardware and software) for any signal that exceeds any preset tolerance or measured sensor values. Once a fault is detected, the fault isolation process determines the exact location of the fault. If the fault is evaluated to be tolerable, the fault accommodation process either accommodates or reconfigures the robots control architecture to successfully carry out the assigned task. Several methods for fault-tolerant control of AUVs have been discussed in the literature [53–57].

FTC approaches can be divided into passive and active FTCs. Passive FTC (PFTC) methods treat faults as sources of system uncertainty, and correspondingly the controller is designed for the worst case scenario. Moreover, they do not rely on the fault detection, since the resulting controller is passively resilient to all the faults considered at the design stage [58]. Active FTC methods (AFTC) only react to faults when necessary. This way, once the fault is detected, the controller can be either selected from a set of pre-defined controllers, or it can be computed online. Since AFTC methods are not designed for the worst case scenario, they can yield better performance than PFTC methods. However, the effectiveness of AFTC relies heavily on the reliability of the fault detection stage.

In contrast to PFTC systems, AFTC methods react to system component failures

actively by reconfiguring control actions so that the stability as well as an acceptable performance of the entire system can be maintained. It should be mentioned that in certain circumstances, degraded performance are accepted too [59–61]. AFTC systems are also referred to as self-repairing [62] [63], reconfigurable [64], restructurable [65] [66], or self-designing [67] control systems by some researchers. Figure 1.2 illustrates the main components of an AFTC system.

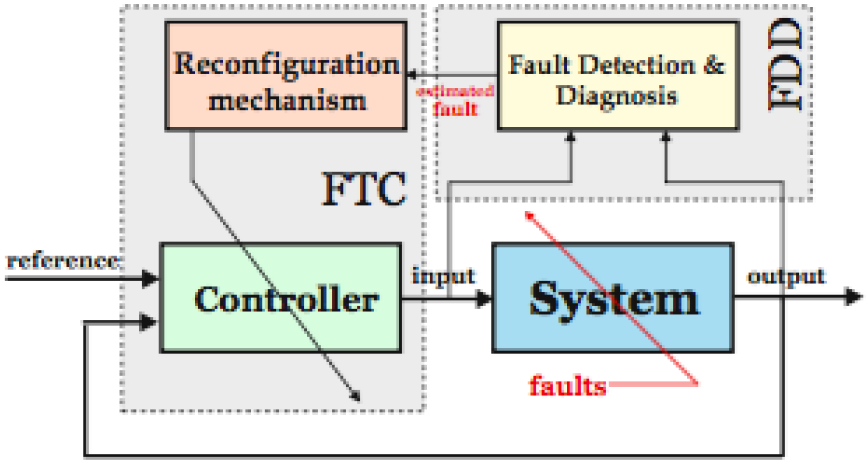


Figure 1.2: Main components of an active FTC system [4].

Fault recovery methods can be categorized by the fault recovery approaches [68]: observer-based methods, parity-relation approaches, optimization-based approaches, stochastic approaches, system identification approaches, nonlinear approaches hybrid system approaches and computational intelligence methods.

In [69], the problem of adaptive FTC was investigated for a class of nonlinear systems with unknown actuator actuator faults. In [69], a design of the normal and fault tolerant controllers is analyzed first. Then, a novel neural networks-based FTC scheme with fault alarm is proposed by using the implicit function theorem. The proposed scheme minimizes the time delay (due to fault diagnosis) between the fault occurrence and the accommodation, and reducing the time delay effects on system performance. The FTC scheme in [69]benefits

from the properties of the PFTC scheme as well as the traditional AFTC scheme's properties. Furthermore, the work [69] requires no additional FDI model which is necessary in the traditional AFTC scheme.

According to [5], in practice, a combination of several methods may be more appropriate to achieve the best overall FTCS. In this regard, a reconfigurable control technique rarely relies on a single control design technique, and instead uses a combination of different control structures and control design algorithms (See Figure 1.3). A technical review on the history and new developments in AFTC systems can be found in [5].

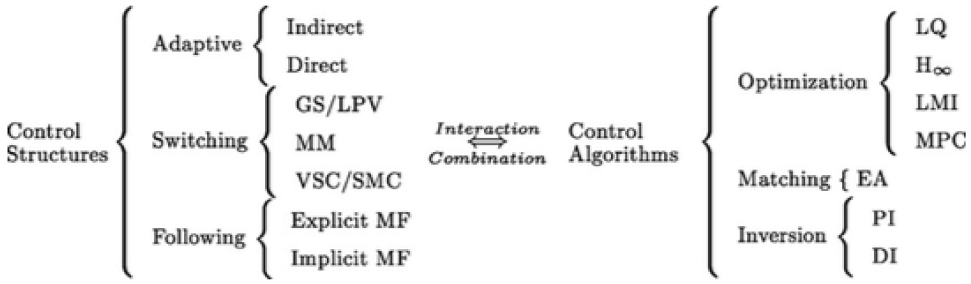


Figure 1.3: Combination of reconfigurable control algorithm in AFTCS [5].

From a more comprehensive view, fault-tolerant policies can also be categorized into two sets: reactive fault tolerance policies and proactive fault tolerance policies [70]. Reactive fault tolerance policies minimize the impact of failures on application execution when the failure occurs. On the other hand, proactive fault tolerance policies aim at predicting failures and move running systems away from conditions that are predicted to fail. The reactive approach is the solution which is most often selected by fault tolerant systems. Typically, system's mission is periodically checked and in case of a failure, the whole control system or a part of it will be restarted. Because this approach is input-output (I/O) intensive, it is not suitable for certain classes of execution platforms such as large-scale systems [71–73].

1.2.4 Computational Intelligence

Computational Intelligence (CI) is the study of intelligent agents [74]. An intelligent system can adapt to changes in its goals and also in the environment, since it learns from experience and can make appropriate decisions within its given perceptual limitations. The main difference between the Artificial Intelligence (AI) and CI is that, AI is not real. For instance, an artificial cherry is not real but, a synthetic cherry may not be a natural cherry while it is a real one and that is how the CI differs.

There is a class of intelligent agents that is somehow more intelligent than humans, and that is organizations [74]. Consider ant colonies that are a prototypical example of organizations. Each individual ant may not be very intelligent, but an ant colony can act more intelligently than any individual ant. The ant colony can discover food and exploit it very effectively as well as adapt to any changes in circumstances. Similarly, corporations can develop, manufacture, and distribute products where the sum of the skills require is much more than any individual could understand. This concept is referred to as synergy. The human society, if viewed as an agent, is probably the most known intelligent agent [74].

CI means that there is a level of abstraction in which one can interpret reasoning as symbol manipulation, so this level can explain an agent's actions in terms of its inputs [74]. Consider an agent as a black box, at any time, the agent has: 1) prior knowledge about the world, 2) past experience that it can learn from, 3) goals that it must try to achieve or values about what is important, and 4) observations about the current environment and itself, as its inputs. The agent's action at any time is the black box output.

Figure 1.4 illustrates a graphical overview of the subsections of the AI domain [6].

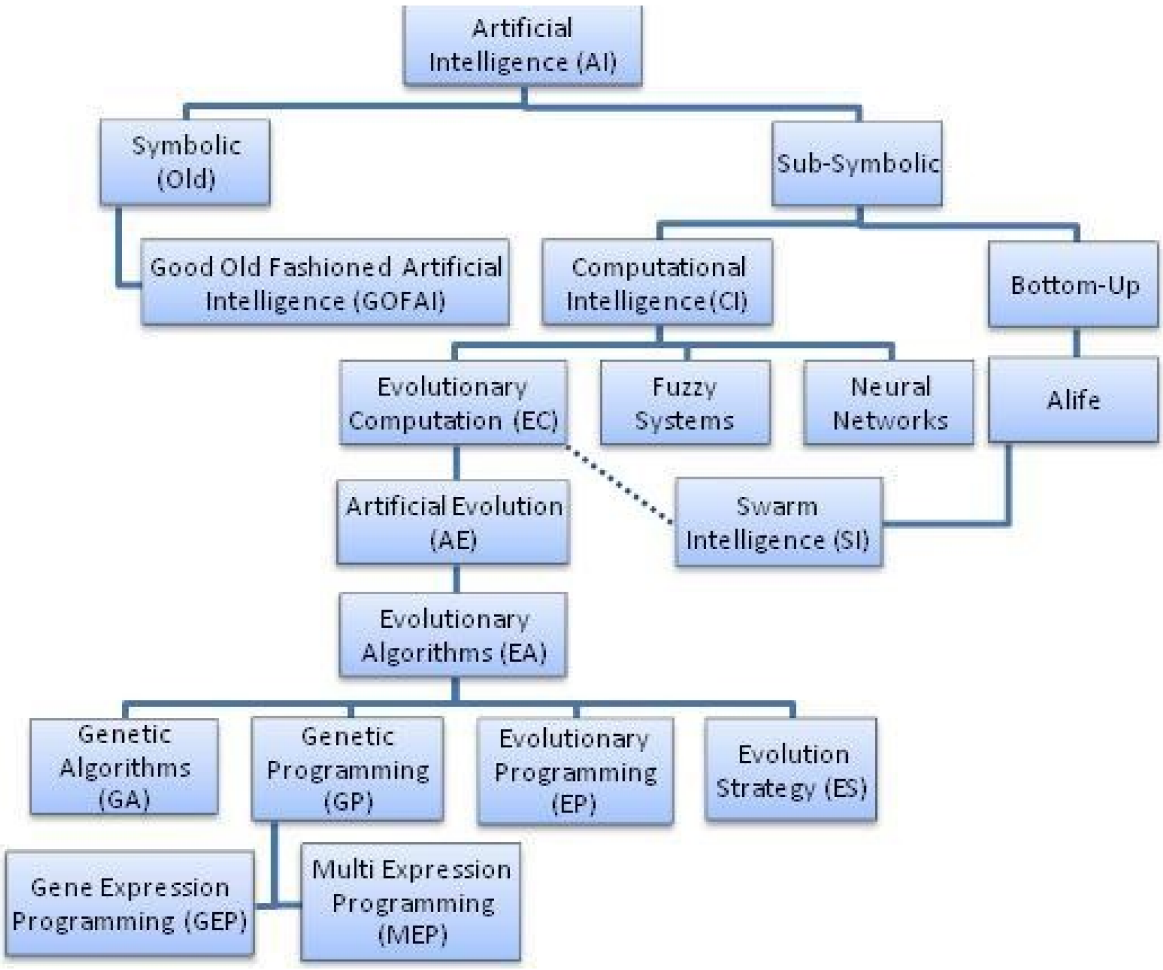


Figure 1.4: Computational Intelligence Hierarchy [6].

Computation Intelligence (CI) Applications

The theories on representation and reasoning are only applicable for the automation of problem solving tasks. CI has many applications such as medical diagnosis, scheduling factory processes, robots for hazardous environments, chess playing, autonomous vehicles, natural language translation systems, and cooperative systems [74]. After investigating essential features of such applications, the following four main tasks should be considered to study principles behind intelligent reasoning and action: 1) modeling the environment, 2) evidential reasoning or perception, 3) taking action and 4) learning from past experience. According to [6], neural network, fuzzy systems and evolutionary computation approaches are also considered as branches of CI.

A consequence of the increase in complexity of the task and physical hardware in our application is to observe an ever-widening gap between our mathematical model and the corresponding practical application. Consequently, the need for research towards the development of approaches having the capability of self-organization under the changing conditions of the task and environment is evident. The work [75] discusses the Variable Structure Systems (VSS) theory in CI.

Traditionally, AI methods are utilized in the literature to deal with the high-level programming. Several programming and control architectures have been developed for high-level controls of mobile robots, particularly for AUVs. Some architectures, such as planning-based systems, are not suitable for real-time operations in systems of reasonable complexity. On the other hand, few other approaches such as behavior-based systems were developed to address real-time concerns and also provide flexibility, however many of them lack a rigorous set of definitions as well as an associated systems analysis. The focus of intelligent control architectures has been on the use of technologies such as adaptation and learning. However, in order to facilitate safe execution of missions in complex environments, few works consider

real-time operations, automatic code-generation, and semi-automatic verification of safety and progress at every design stage. To this end, the work [76] proposed a model-based and hierarchical architecture. From control perspective, an AUV can broadly be divided into lower level control that is concerned with continuous dynamics, and high-level control that is typically discrete and event/time-driven [76].

1.3 Model Predictive Control (MPC) [1]

Nonlinear effects from lift and hydrodynamic damping, added mass, Coriolis and centripetal forces may produce highly degraded performance. Therefore, AUV applications using advanced control techniques are constantly drawing researchers' attention. Meanwhile, model predictive control (MPC) has been extensively studied for more than four decades [1]. Other names for the MPC are rolling-horizon planning, receding-horizon control, dynamic matrix control, and dynamic linear programming. The mid-seventies to mid-eighties is considered to be the true birth of MPC [77]. Due to its capabilities of handling nonlinearities and directly enforcing constraints, MPC finds an increasing number of applications across various fields. This control method has been applied in a broad range of applications such as in chemical process and industrial control [78] [79], control of queuing systems [80], supply chain management [81], stochastic control in economics and finance [82], dynamic hedging [83] and revenue management [84].

From the first algorithms of MPC, they all shared the same structural features. All MPC algorithms obtain a sequence of future control variables based on the optimization of the future system behavior. The first part of the obtained optimal sequence is then applied to the system, and the entire procedure is repeated after a short time interval [7]. MPC is an advanced optimization-based control method that entails extensive online computation of real-time solution to the underlying optimization problems [85].

According to [86], the flexibility of the MPC to plant size and the automated setup are stated as two important advantages of MPC. In addition, MPC controller is capable of dealing with the scalarization of multi-objective optimization problems, which means it might be possible to tackle two or more control goals simultaneously. Subsequently, the AUV path tracking (and even formation) problem can be formulated into an MPC scheme.

The common control objective used in the MPC is the Linear-Quadratic (LQ) objective (cost) function. The LQ objective function along with a linear prediction model and linear constrains give rise to a so called Quadratic programming (QP) optimization problem that can be solved within a finite number of numerical operations [1]. A QP problem is a convex optimization problem with a unique global minimum if the problem is feasible. Moreover, a QP problem ensures that the resulting MPC algorithm is robust for the process control.

The MPC strategy is an important class of the optimal control theory that was developed in 1960s and later [87]. MPC is an iterative optimization technique in which at each sampling time k, it measures or estimates the current state, and then obtains the optimal input vector by solving an optimization problem. The main goal of the optimization problem is to compute a new control input vector, u_k, while taking process constraints into consideration. An MPC algorithm consists of the following items:

(1) A cost function, or a control objective, J_k, which is a scalar criterion to measure the difference between future outputs, $y_{k+1|L}$, and some future reference, $r_{k+1|L}$, while taking the cost of the control input u_k into account. Hence, the objective, which is a measure of the process behavior over the prediction horizon L is minimized with respect to the future control vectors, $u_{k+1|L}$. In this process, the first u_k is used for control only. Figure 1.5 illustrates the general MPC control strategy.

(2) Some constraints, which can be simply treated on the process far more efficient than in conventional control systems (such as the PID-control). This is one of the main motivation behind selecting the MPC [87]. Input amplitude constraints and input rate

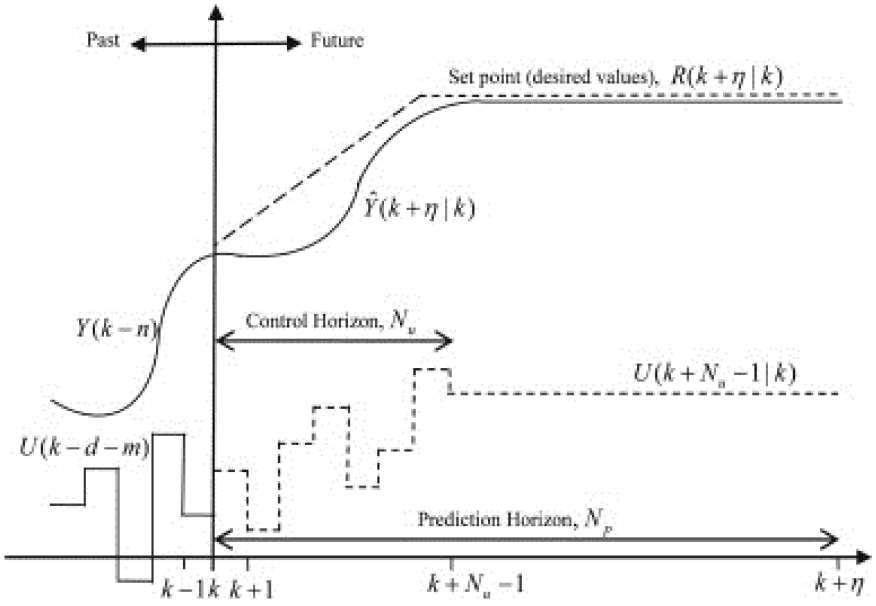


Figure 1.5: General MPC control strategy [7].

of change constraints, known as manipulated variables [86], control variable constraints and terminal constraints lead to an inequality constraint that is added to the optimization problem.

(3) Prediction Model (PM) that is a part of the optimization problem, describing the relationship between the future outputs and the future control inputs. Di *et al.* [87] claimed that the main drawback of MPC is that a model for the process is required. Recently, a model-free MPC by using the subspace identification methods got great attentions [88].

According to [72] and from stability point of view, the MPC framework is particularly appropriate for controlling systems subject to faults due to the fact that the actuator can take advantages of the predicted evolution of the system in order to update the input. To guarantee robust stability of the closed-loop system, MPC controllers must include a set of stability constraints. Different schemes can be found in the literature. The interested reader is referred to see [89] for a review on MPC stability results.

If applicable, MPC algorithms based on linear models should be used due to its low

computational complexity [90] [91]. Various nonlinear MPC techniques have been developed since behavior of many technological dynamic systems are nonlinear [91–93]. In fact, the major limitation of linear MPC is that the plant behavior is described by a linear dynamic model, and hence unsuitable for both moderately as well as highly nonlinear processes with large operating systems [92]. The structure of the nonlinear model and the way it is used in an online manner affect the accuracy, the computational burden, and the reliability of the nonlinear MPC. Fundamental (first-principle) models are usually not suitable for online control strategy although potentially very precise. Such models are consisted of nonlinear differential and algebraic equations which have to be solved in an online manner within MPC. This type of approach is usually requires performance runtime monitoring since fundamental models can be very complex and may lead to numerical problems. Moreover, development of fundamental models is difficult in many applications. If a process exhibits significantly nonlinear behavior, the classical PID controller and the MPC algorithm based on a linear model are unable to control the system efficiently. Linear Quadratic Regulator (LQR) method, which guarantees the operation of a dynamic system at minimum cost, lacks the constraints on input, output, and state variables, while MPC brings them directly into account [94].

The work [95] is proposed as a benchmark setup for several MPC methods for nonlinear and piecewise affine systems. Corona *et al.* [95] presented a description of the methods to be compared, and collected the comparison results in a table. In particular, the trade-offs between complexity and accuracy of the solution, as well as computational aspects are highlighted in [95]. Practically, there exist several drawbacks in choosing nonlinear MPC [95–98] such as higher online computation time, and a larger online memory. According to [87] and [99], a nonlinear MPC is generally not guaranteed to converge within a reasonable computing time. Also, a nonlinear optimization problem often has flaws in local minima and convergence problems. Hence, a nonlinear MPC method may not be robust for some

cases within an online process control.

To reduce the severe performance runtime problems associated with the nonconvexity, a suboptimal approach for continuous-time systems is proposed in [100] that employs an initial feasible solution that is improved iteratively in lieu of optimization. The work [99] extends these results by showing that under mild conditions, feasibility rather than optimality is sufficient for stability. Also, for nonlinear discrete-time systems, under mild condition, feasibility is sufficient to establish the stability of suboptimal fixed horizon versions of MPC. Motivated from [101], a performance runtime efficient nonlinear MPC algorithm with the nonlinear prediction linearized optimization was applied to the temperature control of a yeast fermentation reactor based on a neural model. On the other hand, the work [95] presents the linear methods with the minimum position and velocity overshoot values indicating with how far the vehicle overtakes the reference. Also, from the control cost perspective, linear methods has lower control costs. Adding up the mentioned literature, the importance of using a methodology which is a trade-off between linear and nonlinear MPC is feasible.

1.3.1 MPC-based Hybrid Control Methods

The advantages of making a system hybrid with CI methods are significant. Timed and hybrid automata have proved to be a successful modeling framework for formal verification [51, 102–107].

A good example of the fusion with CI-based methodologies is the integration of CI and sliding mode control (SMC) available in [103]. A pure SMC suffers from some drawbacks. First, the chattering problem (the high frequency oscillations of a controller's output due to the high speed switching) should be considered for the establishment of a sliding mode. In real-world implementations, chattering is highly undesirable as it may excite unmodeled

high-frequency plant dynamics leading to unforeseen instabilities. In addition, an SMC is vulnerable to measurement noise since its input depends on the sign of a measured variable that is very close to zero [108]. Moreover, the SMC may employ unnecessarily large control signals to overcome parametric uncertainties. Finally, it is difficult to calculate the equivalent control exists that needs a complete knowledge of plant dynamics. Kaynak *et al.* [103] discuss how some intelligence can be incorporated in SMCs by using CI methodologies.

In [104], a new control strategy using MPC is developed and simulated on an AUV to track the reference heading provided by a guidance system. MPC is chosen because of several reasons such as the ability to handle constraints in a natural way. A novel approach for the implementation of nonlinear MPC is proposed using GAs [105]. The proposed method in [105] formulates the MPC as an optimization problem. Application to two types of nonlinear models are studied, namely Hammerstein and Wiener Models. The simulation results are illustrated for two chemical processes to demonstrate the performance of the proposed scheme.

In [102], an MPC problem with fault-tolerance capabilities is formulated within the hybrid system framework. In particular, the mixed logical dynamic form is considered to represent hybrid systems. Using this approach, a hybrid model of the system to be controlled is obtained, which includes inherent hybrid phenomena and possible modes caused by fault occurrences. This allows to adapt the system model online by taking into account the fault information provided by a fault diagnosis and isolation module. In this way, the controller can cope with the considered faults. Additionally, different implementation schemes and fault-tolerance evaluation procedures for hybrid MPC (HMPC) considering fault-tolerance capabilities are proposed and discussed [102]. Finally, to exemplify the implementation of the proposed approach along with considering actuator fault tolerance, the proposed approach is applied to portion of the sewer network of Barcelona.

In the work [106], an asymptotically stable combined kinematic/torque control law is

developed using a backstepping approach to accommodate the complete dynamics of robots. Moreover, a neural network is introduced to approximate the dynamics of the robots using online weight tuning.

1.3.2 Fault-Tolerant MPC

MPC is potentially a promising tool for fault tolerant control applications , due to its prominent capabilities such as constraint handling, flexibility to changes in process dynamics, and the applicability in nonlinear dynamics [5, 109–112]. Since MPC recalculates the control signal at each sampling time, any change in process dynamics can be reflected simply into the control signal calculation. A fundamental question about MPC is its performance robustness to model uncertainty and noise. When we say that a control system is robust we mean that, the performance specifications are met for a specified range of model variations and a class of noise signals. [113].

Fault-tolerance aspect in control methods requires an effort at every stage and in all aspects of system design [60]. Most of the literature in control systems only consider problems which are based on mathematical models of the plant. A Fault-Tolerant Control (FTC) method should ideally be accompanied by a systematic integrated approach. The FTC strategy should start with an understanding of the system structure, the reliability of different components, the types of redundancy available (or to be generated) and the types of the controller function that are available.

There exists two main AFTC approaches in redesigning or recovering the controller to become fault-tolerant, that are: fault accommodation and control reconfiguration. The fault accommodation task is performed by adapting controller parameters according to the dynamical properties of the faulty system. In this recovery approach, the input and output signals of the system remain the same like in the fault-free case. If the fault accommodation

task does not perform sufficiently, the complete control loop has to be reconfigured in the fault reconfiguration task, a new control configuration is selected where alternative input and output signals are used [114]. It should be mentioned that in the process of fault detection, accommodation and controller reconfiguration, a percentage of degradation in the performance of the controller is feasible.

In [115], Roofigari *et al.* have developed a novel control scheme based on coupled modeling of a satellite controlled by MPC, and then they studied the local-level recovery accommodation of faulty agent performance by using local fault information. The work [115] also shows the enhancement of the recovery performance by reducing the oscillatory behavior.

In [116], Sedaghati *et al.* considered the problem of limited information availability in underwater applications that are controlled based on the MPC technique. An active fault recovery scheme is proposed and its performance is compared in tracking and formation keeping in the presence of actuator faults. In [116], a virtual vehicle formation coordination method is considered to achieve individual and cooperative objectives depending on their significance to individuals. Gopinathan *et al.* [117] used an MPC strategy as a basic control law within the framework of Multiple Models, Switching and Tuning (MMST) to design a reconfigurable flight control system. The main feature of the overall controller is that due to the use of MPC, it can explicitly consider hard constraints on control inputs. It can also achieve an acceptable flight performance in the presence of control effector freezing. In order to obtain an effective reconfigurable control design, a new parameterization of the aircraft model in the presence of control effector freezing is suggested. It turned out that such a parameterization is well suited for use within an MPC framework. The overall Multiple Model Predictive Control (MMPC) scheme quickly identifies the nature and time instant of the failure, and then carries out an automatic reconfiguration of the control law by achieving an acceptable flight performance. The properties of the reconfigurable controller in [117]

are evaluated through simulations of an F/A-l8A aircraft carrier landing maneuver in the presence of critical control effector failures.

In [118], an actuator FTC scheme combining tube-based MPC and set-theoretic FDI approached was proposed. The work [118] presents a passive fault detection scheme by using invariant sets and an active fault isolation scheme by relying on MPC and tubes. The use of tube-based MPC and set-theoretic FDI is interesting due to their relatively low computational complexity, robustness of the FDI, and their proper combination to implement the proposed active Fault Isolation (FI) strategy. Thus, the proposed fault-tolerant MPC scheme has robust FDI performance, low computational complexity, and less conservative FI conditions. The key idea of the proposed FTC scheme in [118] is to design the input and state sets for an active FI. The sets are chosen by offline trial and error as a practical method, which can be improved if a systematic design method can be proposed for the input and state FI sets in the future. It should be emphasized that the main drawback about the proposed FTC scheme in [118] is that it cannot detect all faults. Thus, for undetectable faults, the PFTC ability of this scheme can still tolerate them to some extent even though a possible degree of performance degradation may appear. Advantages of the proposed FTC scheme lie in its relatively simple structure and less conservative active FI.

In a recent series of papers, Lyapunov-based model predictive control (LMPC) schemes for nonlinear systems are proposed as a way to guarantee the closed-loop stability [72, 73, 119]. These LMPC methods, known as proactive MPC, are based on uniting receding horizon control with Control Lyapunov Functions (CLFs). This method has been a popular topic in the communication systems and aerospace control systems communities for the last 10 years [120]. Recent studies deal with the design of FT controllers for nonlinear systems subject to sensor faults such as time-varying measurement delay in the feedback loop or data loss. Recent proposed LMPC scheme is applied on nonlinear systems use nominal model of the system along with the faulty measurement to estimate the current state, when

measurement fault occurs. Then, when no measurement is available, due to the fault, the resulting estimate is applied to evaluate the LMPC controller instead of setting the control input to zero or to the last available parameter value.

When it comes to AUV application we come across difficulties in case of implementing such method on our proposed MPC algorithm in Chapter 3. First, proactive approaches need a huge data available. Resulted from the available data and based on the analysis of the design of normal and passive fault tolerant controllers, such fault-tolerant control schemes are proposed. Since such data history is not available in our application, proactive scheme seems to be not feasible. Second, in case of actuator faults the actuator should implement the last optimal input trajectory evaluated by the controller. Since this requires that the controller must stores the last evaluated optimal control input trajectory in its memory, the complexity of the system increases, undesirably. Moreover, to the best of our knowledge, there is no guarantee, so far, about consequent faults to be reconfigured or accommodated in the proactive schemes.

1.4 State Estimators in Nonlinear Systems

After Recursive Least Squares (RLS) used commonly for parameter estimation of linear systems [121], Kalman filter became the optimal parameter estimator, particularly, in tracking applications [122]. Kalman filter uses the hypotheses of Gaussian measurement and process noises and the linearity of state and measurement equations [12]. Recently, there has been a great deal of interest to use the online fault estimators [123–125]. Few of the literature have investigated moving horizon estimation (MHE) for nonlinear systems [126].

The Extended Kalman Filter (EKF) is being extensively used as a standard technique for recursive estimation for nonlinear systems and machine learning applications. Few of these applications include estimating states of a nonlinear dynamic system, identification

of the nonlinear system parameters (e.g., learning the weights of a neural network), and dual estimation (e.g., the Expectation Maximization (EM) algorithm) where both states and parameters are simultaneously estimated. The work [127] provides the reader with an overview about EKF and also the advantages of Unscented Kalman Filters (UKF), which is another variant of the KF. The parameter estimation block, in [128], uses either MHE or UKF to estimate the fault parameters. Using parameterized model to calculate the control signal, the MPC adopts the input and output information of the process in an online manner at each time step. In [128], an MPC based fault tolerant controller has been integrated with an MHE and/or UKF for fault parameters estimation to form an active FTC system. Simulation results in [128] suggest that the MPC-based fault tolerant controller provides prominent fault tolerant capabilities to the control of quad-rotor helicopter with constrained nonlinear dynamics. Furthermore, it satisfies all system constraints and performs control redistribution effectively in an optimal manner. It is also seen that the parameters of the fault converge faster using MHE when compared with the UKF-based fault estimator. However, the computation time of MHE is slightly higher than that of UKF.

In the work [129], a robust MPC controller for constrained discrete-time nonlinear systems with additive uncertainties is presented. This strategy follows the general MPC formulation that it is based on the nominal prediction of the states. It also considers a terminal cost and a terminal constraint on the state. Assumptions on the design parameters of the MPC controller are given in order to guarantee robust constraints satisfaction. The design in [129] is based on computation of the bound on the mismatch between the nominal prediction and the uncertain evolution of the system. The drawback of the work [129] is that over-conservative bounds may be obtained since the design is based on the Lipschitz continuity of the system as well as the open-loop nature of the predictions.

EKF can suffer from suboptimal performance and sometimes divergence due to errors introduced by the first-order approximation of the true nonlinear dynamics [130] [131]. To

overcome the above limitations, the works [130] [131] utilized a multi-layer feed-forward (static) neural network due to the excellent universal approximating properties and also the availability of effective online adaptation algorithms.

In the work [132], an FTC methodology is presented to handle possible sensor and/or actuator faults, any abrupt changes in model parameters and unmeasured disturbances. The work [132] integrated a bank of adaptive unscented Kalman filters (AUKFs) and fuzzy-based decision making (FDM) schemes in the proposed FDI module. The proposed FDI approach in [132] recursively corrects the measurement vector as well as the model used for both state estimation and output prediction in an MPC formulation. For robustness of the proposed FTC system in [132], H_∞ optimal robust controller and an MPC are combined via a fuzzy switch that is used for switching between MPC and robust controller. Therefore, the FTC system is able to maintain the closed loop stability in the face of abrupt changes in model parameters and unmeasured disturbances. The proposed FTC methodology in [132] can handle soft faults due to bias and drift in sensors and actuators and any model mismatch that cannot be isolated as faults by the FDI module. Also, it facilitates recovery of the closed loop performance after the faults have been isolated leading to an offset free behavior in the presence of sensor/actuator faults that can be either abrupt or drift change in biases.

The particle filters can be utilized as an alternative for real-time applications approached by model-based Kalman filter techniques [133–135]. Recursive implementations of Monte Carlo-based statistical signal processing are known as particle filters. According to [133], promising results in case of highly nonlinear models, or in the presence of non-Gaussian noise, especially in applications where computational power is rather cheap and the sampling rate is moderate. According to [136], there are still many results to be conducted be sure about the convergence of the empirical distributions generated by particle filtering methods. Similarly, Crisan *et al.* [136] proved that the crucial uniform convergence

results rely on strong assumptions on the dynamic models making particle filters inapplicable for most real-world problems.

UKF uses several sigma points (which if treated as elements of a discrete probability distribution has mean and covariance equal to the given mean and covariance) to calculate the mean and covariance of random variables [137]. These sigma points propagate through the true nonlinear system. The posterior estimation is then calculated by the average of the sigma points. The UKF is essentially a kind of Quasi-Monte Carlo method that has the ability of processing nonlinearities [138]. The UKF is only valid when the posterior distribution can be closely approximated by a Gaussian distribution [139]. The drawbacks of UKF are as follows [140]:

- It preserves the linear update structure of the Kalman filter which is optimal only in linear Gaussian systems.
- It uses second order moments which is only valid for Gaussian distributions.
- The number of sigma points is small and may not represent adequately complicated distributions.

The work [141] shows that a UKF implementation on our application gives a minor improvement and further discusses with the amount of heading uncertainty we are faced with, EKF does not suffer abrupt catastrophic inconsistency.

The work [142] presents a Cubature Kalman filter (CKF [143]) as an alternative for UKF. Gustafsson *et al.* [142] claim that the unscented transform may have a negative weight for the center point. This might cause problems when implementing the UKF such as using the square root form. On the other hand, the CKF has a similar set of sigma points. The points have positive weights, and the central point is left out. Authors in [142–144] concluded that apart from all the advantages, CKF may marginally improve our system's

performance at the expense of a reduced numerical stability and an increased computational cost.

1.4.1 Dual Extended Kalman Filter (dual-EKF):

Dual-EKF was proposed by Wan and Nelson in [145]. This variant provides a boot strapping procedure for combined state and parameter estimation using two EKFs [146]. Therefore, the state and parameter estimation problems are split, although the two problems cannot be entirely separated due to their inherent interdependencies. Thus, the system is provided with measurement updates on states and parameters of its system and reconfigures. One of the advantages of this technique is that there exists the possibility to switch off the parameter estimator, once a sufficiently good set of estimates has been obtained. The work [12] shows the state-space formulation of the dual-EKF algorithm makes it applicable to a much wider variety of contexts than has been explored in the literature. The simultaneous estimation of the state and disturbance not only improves state observer robustness, but also helps to compensate for disturbances in the controller according to the work [147].

The work [12] illustrated dual-EKF as a fundamental method for solving a range of fault-tolerant predictive control problems related to signal processing. In [12] modeled the AUV application by bringing a number of examples to illustrate the performance of the dual-EKF methods and its ability to capture the underlying dynamics of a noisy time series. Interested reader is referred to read [12] for more information on dual estimation algorithms.

1.4.2 Observer-based Nonlinear Model Predictive Control

In Section 1.3, the importance of nonlinear model predictive control (NMPC) is highlighted. Applications of NMPC for AUV are less frequent. A dual observer in [147], which combines

a state and a perturbation observer, aims to solve the problem of being sensitive to external disturbance. The work [148] considered the elevator angle constraints and proposed a controller based on MPC with artificial bee colony algorithm in which a classical linear state observer is used. In [149], a novel discrete-time proportional and integral observer is used to estimate the states, output, input, and disturbance together. Zhang *et al.* [150] proposed a new reduced-order observer with multi-constrained thoughts by using specific system decomposition. However, these results only use linear models.

In contrast, most practical systems are nonlinear and therefore nonlinear models are required. Authors in [151] proposed a nonlinear observer based on dynamic model of AUV, which is used to estimate the vehicles velocity. The work [152] investigated a nonlinear feedback control algorithm with a nonlinear state observer, but the error dynamics stability was not considered in observer design. The work [153] extended an adaptive state observer to a class of nonlinear systems. However, due to the selected special Lyapunov matrix, there is a reduction of the accuracy of state estimation.

As it is mentioned earlier, the MPC needs state variables to achieve desired output tracking in the minimization of cost function, so the role of observer-based MPC is significant in practical implementation [154]. So, the motivation of this thesis is to address the aforementioned problems and deficiencies, and the aim is to design an observer-based MPC with nonlinear AUV kinematics and dynamics model.

1.5 Combining Control Algorithms with Neural Network (NN)

Neural Network (NN) properties such as, learning, nonlinear mapping, parallel processing are attractive for underwater control problems [155]. Fujii *et al.* [156] investigated an application to control problems of underwater robots and developed an NN-based adaptive control system called SONCS (Self-Organizing Neural-net Control System). The work [156] constructs a neural network as a feedback controller based on a back propagation method proposed in [157]. Yuh [158] described an NN control system using a recursive adaption algorithm with a critic function (reinforcement learning approach). The special feature of the controller proposed in the work [158] is that a system directly adjusts itself in an online manner with no explicit model for vehicle dynamics.

The works [159] [160] employed an NN method as well as a PID control strategy to develop a robust adaptive controller. In [161], an NN combined with variable structure method is used to develop the position controller. A neural learning control design is presented in [162] for trajectory tracking of ocean ships in the presence of unmodeled dynamics as well as environmental disturbances.

According to [163], a combined system structure providing an input to an NN in AUV equipment will be useful for enhancing mission control operation. Motivated from the statistical learning theory, the work [11] presents a one-layer recurrent NN for the support vector machines (SVM) learning in pattern classification and regression. In [11], the SVM learning problem is first converted into an equivalent QP formulation, and then a one-layer recurrent neural (RNN) network for SVM learning is proposed. For large scale and real-time optimization problems, RNN emerged as a promising computational approach. The work [164] presents a one-layer NN for solving convex optimization problems. In [165], another one-layer neural network is presented for pseudo-convex optimization problems.

These RNNs are shown to perform well in terms of convergence property as well as model complexity.

Liu *et al.* [160] aim to perform a group of vessels to automatically position themselves in a desired time-varying formation. Liu *et al.* [160] use a dynamic surface control technique as well as distributed adaptive controllers to track a reference position by using the information of neighboring vessels. The work [160] uses iterative updating law of NN to accurately identify dynamical uncertainty and time-varying ocean disturbances. In [160], two types of adaptive laws are proposed and validated, namely 1) direct iterative updating laws based on the velocity tracking errors and, 2) composite iterative updating laws based on tracking errors and prediction errors. Then, the work [160] employed Lyapunov-Krasovskii functions to analyze the stability of the closedloop network. Due to the use of a distributed control strategy in [160], the information exchanges among agents are reduced. Also, the mixed uncertainty that includes the internal model uncertainty as well as external time-varying ocean disturbances is compensated. Liu *et al.* [160] claimed that their proposed controller in comparison with previous literature is 1) easier to implement in digital processors, as they used derivativefree laws and, 2) they achieved a faster adaptation and improved performance by using iterative neural control laws.

Li *et al.* [166] proposed an FDI strategy based on a Dynamically Driven Recurrent Neural Network (DDRNN), to be used in situations when there are actuator failures in satellite's attitude control system. The architecture in [166] is designed to consist of two DDRNNs. The first DDRNN diagnoses the presence of a faulty thruster and the second DDRNN identifies the faulty actuator. In [166], it is shown that in the presence of external disturbances and noise, the proposed scheme is more robust as compared to a scheme that is based on a single feed-forward back-propagation NN or a single DDRNN scheme. The fact that using global feedback is likely to reduce the memory requirement significantly, is one of the privileges of the work [166].

1.5.1 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) is an NN model initiated in the 80's for modeling time series. The structure of this network is similar to feed-forwrad neural network, with the distinction that the connection topology has cycles. Figure 1.6 presents the RNN as a class of artificial NN where connections between units form a directed cycle.

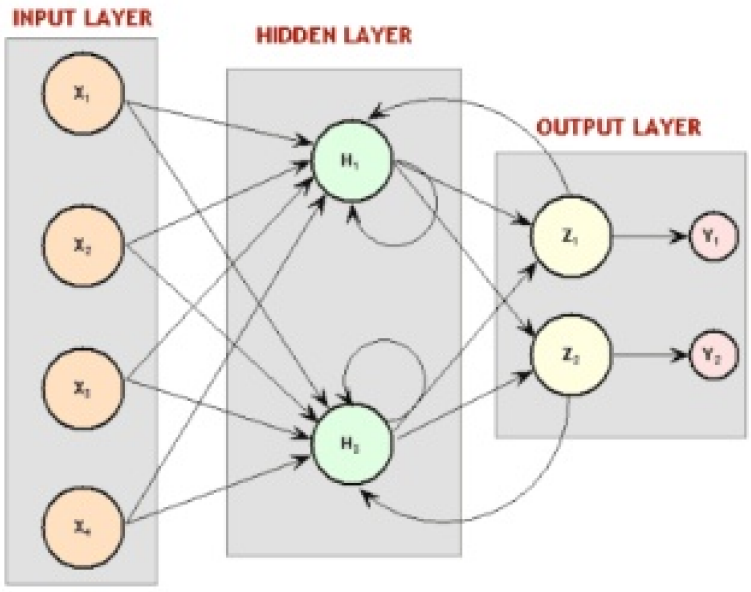


Figure 1.6: One-layer Recurrent Neural Network structure [8].

The existence of a recurrent hidden state, whose activation at each time is dependent on that of the previous time, has profound impacts. The desired features which motivates us to use RNN in this work are as follows [167]:

- An RNN may develop a self-sustained temporal activation dynamics along its recurrent connection pathways, even in the absence of input. Mathematically, this renders an RNN to be a dynamical system, while feed-forward networks are functions.

- If driven by an input signal, an RNN preserves a nonlinear transformation of the input

history in its internal state. In other words, an RNN has a dynamical memory, and hence it is able to process temporal context information.

RNNs emerged as a promising approach for addressing the issue of heavy online computational tasks. Neural optimization has an inherent nature of parallel and distributed information processing and its hardware implementation is available. Various NNs are proposed to solve linear programming, quadratic programming (QP), general convex programming, and pseudoconvex optimization problems. These networks are categorized based on their duality and projection methods, namely the projection NN, the general projection NN, the simplified dual network, the improved dual NN, the RNNs for non-smooth optimization problems, the one-layer RNNs with discontinuous activation functions, and the delayed projection NNs [167].

Neural networks are used in various domains such as pattern recognition, signal processing, system parameter identification, and automated diagnostics *etc.*. Two classes of network have received much attention in underwater control systems that are: 1) the multilayer static NN which is a memoryless mapping of inputs to outputs and, 2) the recurrent network which has associative memory of its present state [163]. The static network is successful in identification of patterns while the recurrent networks are applied to optimization problems in which emergent dynamic behavior can be mapped.

Since RNN possess many desirable properties such as real-time information processing, it received tremendous interest for optimization, control and signal processing [168]. There are several RNN approaches to solve quadratic programming problems. Kennedy and Chua in [169] presented a primal neural network for solving the quadratic problem obtained from MPC. The network proposed in the work [169] contained a finite penalty parameter which converged to an approximate solution. To overcome the penalty parameter Xia *et al.* in [170] proposed a primal-dual neural network with a two-layer structure for solving some convex

QP problems. The works [168–170] cannot be shown to have a finite-time convergence and exponential convergence to the optimal solution of their corresponding quadratic problems. Therefore, designing NN control framework with a low complexity and a fast convergence rate, especially in real time application, is of importance.

There exist literature on MPC controllers which are based on using RNN [2, 171, 172]. These MPC-RNN approaches (sometimes known as Neurodynamics-based MPC approaches) are developed to improve system's computational efficiency as well as the control performance. In case of applying such control approaches on an AUV, the dynamical uncertainty and time-varying environment disturbances in the agent dynamics can be compensated by NN using recurrent updating laws. In the work [171], a simplified network is used for solving real-time quadratic optimizations in various MPC approaches. A two-layer neural network is applied for solving reformulated minimax optimization problems of robust MPC approaches in [172]. In the work [2] Wang *et al.* applied an RNN for solving the QP problem in real-time. The shortcomings of the work [2] (detailed in Section 2.9) are another motivation for this thesis to improve the MPC-RNN approach.

Recurrent networks can be single or multiple layered. Considering the trade-off between less system complexity in the AUV application and gaining a superior performance in terms of global convergence as well as parallel computational implementability, one-layer RNN is integrated into our proposed hybrid control framework.

1.6 The Procedure of Choosing Methodology

Choosing between model-based techniques and model-free ones is highly related to the application of the system. A model-based technique has the merits such as low cost, high reliability and easy realization for AUVs. However, it is difficult to build an accurate model for autonomous underwater vehicles due to the effect of model error, measurement noise,

outer disturbance and *etc.* [173]. In model-free approaches, a variety of techniques have been sought to assist fault detection, isolation and recovery [174].

Artificial Intelligent-based methods (such as NNs and SVMs) as a broad category of data driven approaches, can handle highly nonlinear problems but their drawback is that, they require huge number of training data which is often not available in practice [175]. Perhaps the major issue with data-driven models is whether the causality among the variables are modeled and if so, what variables are related causally. A model causally relates two variables if it correctly shows that a change of a certain magnitude in one will result in a change of a certain magnitude of the other. In data-driven models, causality among variables is determined entirely by the nature of the data and by the structure of the empirical model. If independent variations are not present in certain manipulated variables, then no causality information for effects of those individual variables will be present in the data, nor in any model built from them.

The work [174] looks at recent advances in the use of data-driven models built from such historical data for monitoring, fault diagnosis, optimization and control. Latent variable models are used because they provide reduced DOF models for high DOF systems. They also provide unique, interpretable and causal models, all of which are necessary for the diagnosis, control and optimization of a process. The drawback of such data-driven models is the need for a routine plant data that are of a very different nature from typical R&D data collected usually under designed experiments. Only few works simultaneously take advantage of mathematical model of a system as well as the adaptive nature of intelligent techniques especially NNs in a hybrid frame work [176].

1.7 Thesis Objectives and Contributions

This section presents the problem statement, the proposed methodology, and the contributions of the work developed in this thesis to solve the problem.

1.7.1 The Problem Statement

In this thesis, fault tolerant control of an autonomous underwater vehicle with uncertainties is addressed by using a hybrid method of Model Predictive Control and Recurrent Neural Network for control and fault recovery purposes. The main objective is to develop a fault tolerant strategy so that the autonomous underwater vehicle with uncertainties follow the desired trajectory while meeting a set of requirements and bounds on position, orientation, linear and rotational velocity, and actuator efforts. These requirements, which are depicted to met in the autonomous underwater vehicle's mission, aim to minimize the control cost along with the performance runtime of the system while the performance of the autonomous underwater vehicle in path tracking remains satisfactory. Another objective of this research is to address the fault detection-recovery task in order to overcome the loss-of-effectiveness fault in the actuators of the autonomous underwater vehicle.

1.7.2 Methodology

The problem of trajectory tracking and nonlinear path following have been discussed in Section 1.2.1. Section 1.2.2 states the importance and necessity of fault tolerant control as well as the advantages of using a combination of several methods to achieve the best overall fault tolerant control system. The procedure to choose a model-based approach which accounts for system's nonlinearity and uncertainties is presented in Section 1.6.

Motivated by the literature given in Section 1.3, the model predictive algorithm is selected as the base method for the control of an autonomous underwater vehicle with

uncertainties considering the designed mission objectives. Model predictive control is introduced as a promising tool for fault tolerant control applications, due to its prominent capabilities such as constraint handling, flexibility to changes in process dynamics, and the applicability in nonlinear dynamics. In Section 1.3 the importance of using a methodology which is a trade-off between nonlinear and linear model predictive control is highlighted. Then, in Section 1.4.2, an observer-based nonlinear Model Predictive Control is explained as a feasible solution of the problem stated in this research. Also, the benefits of formulating the model predictive control problem which has fault-tolerance capabilities within the hybrid system framework are discussed.

In Section 1.5, several literature given on advantages of combining the system controller with neural networks, particularly, learning, nonlinear mapping, and parallel processing. Then, recurrent networks demonstrated as a class of network that has received attention in underwater control systems. Coming to a trade-off between less system complexity in the autonomous underwater vehicle's application and gaining a superior performance in terms of global convergence and parallel computational implementability, one-layer recurrent neural network got selected to maintain our proposed hybrid control framework to address the designed problem of this research.

The approach proposed in this thesis falls into a hybrid of nonlinear Model Predictive Control and Recurrent Neural Network control methodology to benefit from both a priory mathematical model information of system and the adaptation capability of recurrent neural networks. Therefore, a performance-runtime efficient nonlinear Model Predictive Control is developed to control an autonomous underwater vehicle. Since, the algorithm requires solving an online quadratic programming problem, a recurrent neural network is employed to guarantee obtaining the optimal solution of model predictive control in each sampling time. The main feature of the overall developed controller is that due to the use of model predictive controller, it can explicitly consider hard constraints on control inputs,

and achieve acceptable path tracking performance. Also, since the controller benefits from the adaptability of recurrent neural network, it shows to be more tunable as a step toward addressing the problems of unachieved goals resulted from choosing small prediction horizon numbers and the necessity of remaining cost-to-go estimation issue.

Considering the above mentioned, the model-based approach chosen in this study calls for the need of a recursive estimation for nonlinear system of an autonomous underwater vehicle with uncertainties. In Section 1.4, the pros and contras of using Extended Kalman filter in comparison with other nonlinear estimation filters are discussed. Although extended Kalman filter has its set backs, but considering the problem stated in this thesis and the objectives this research sought, applying this estimation filter technique is feasible.

Moreover, the fault tolerant properties that the dual extended Kalman filter provided the system with, have explained in Section 1.4.1. Then, in the second part of this thesis, motivated by the literature on dual extended Kalman filter methods in Section 1.4.1, the developed hybrid controller, integrated with dual extended Kalman filter to accomplish the objectives of an autonomous underwater vehicle's mission through the designed path tracking that faces the vehicle with severe loss-of-effectiveness actuator faults. The integration of dual extended Kalman filter with the developed model predictive control and recurrent neural network control method yields an active fault-tolerant control scheme that is applied to an autonomous underwater vehicle with uncertainties. In Chapter 4, the mentioned active fault-tolerant control system faults are detected and identified by a fault detection identification scheme, and the controllers are reconfigured accordingly, online in a single frame.

1.7.3 Thesis Contributions

The contributions of the work developed in this research to solve the aforementioned problems are listed below.

- In Chapter 3, a hybrid Model Predictive Control (MPC) and Recurrent Neural Network (RNN) scheme for the tracking control problem of an AUV is developed while considering the application's constraints in path following as well as improving the performance runtime. The developed scheme considers the constraints on system states and meets the objectives of this research according to the system's mission requirements such as lower tracking error cost for the controller, while considering the performance runtime as an important constraint. A comparison section including simulations in various scenarios and the corresponding discussions are provided to evaluate the developed control method.

- In Chapter 4, an active recovery fault-tolerant hybrid control is developed by integrating the MPC, RNN and dual extended Kalman filter (dual-EKF). The developed method meets the objectives of the desired mission, namely, the capability to handle constraints on the system states, improving the control cost of the controller and performance runtime of the system while effectively dealing with severe actuator fault scenarios. Since, the proposed method uses a dual-EKF, it updates its control distribution matrix entities at each time step. This way, under the faulty conditions a control recovery action is taken in order to keep the performance of the faulty AUV dynamics close to the healthy AUV dynamics. Similarly, a comparison section including simulations of various scenarios and discussions is provided to evaluate the developed active fault-control method.

1.8 Thesis Outline

This thesis is organized as follows.

In Chapter 2, the background information required for obtaining the dynamic equations of underwater vehicles, MPC method and the extended Kalman filter is provided. MPC principles as a preliminary to our proposed control method is presented and the nonlinear discrete-time form of the system is used for applying to the MPC framework. Also, some background information about fault types presented. The Quadratic Programming problem explained and the Recurrent Neural Network (RNN) that is emerged as a promising approach for addressing the issue of online convex QP problem solving, is also provided.

In Chapter 3, the proposed nonlinear hybrid control method using an EKF is presented as an efficient performance-runtime NMPC algorithm, which requires solving a QP problem via an RNN in an online manner. For the evaluation of the proposed control algorithm in the designed mission of this research, three control systems have been considered in this chapter. 1) Linear MPC with KF state estimation, 2) Nonlinear MPC with EKF state estimation, and 3) the developed MPC-RNN with EKF state estimation. The simulation results and discussions regarding an AUV trajectory tracking is presented for all three scenarios.

In Chapter 4, the developed methodology in Chapter 3 is integrated with a dual-EKF that can effectively deal with fault detection and accommodation in case of loss-of-effectiveness actuator faults. Then, different scenarios are designed to evaluate the developed approach. First, three different scenarios are designed to demonstrate the importance of applying FTC on our developed control scheme from Chapter 3. Then, three different scenarios are designed as a comparison between the developed approach of this chapter with nonlinear MPC AFTC approach faced with loss-of-effectiveness actuator faults. Simulation results and discussions are presented at the end of this chapter.

Chapter 5 presents conclusion as well as remarks for our developed methodologies.

Few suggestions and potential future work are also provided.

Chapter 2

Background Information

2.1 Introduction

This chapter provides the information needed for deriving dynamic equations of underwater vehicles, Extended Kalman Filter (EKF) and Model Predictive Control (MPC) method. Moreover, an overview of the Recurrent Neural Network (RNN) proposed in work [11] to solve the strict convex quadratic programming problems, is explained. A description of the kinematic and kinetic model of a 6 degree of freedom (DOF) AUV is provided and a reduced order dynamical model is obtained after analyzing our problem, stated in Section 1.7.1, in the horizontal plane. Then, the effect of ocean currents on the model is considered by means of applying a 2 dimensional current model for the submerged body.

The principles of MPC are presented as a preliminary to our proposed control method. The nonlinear discrete-time definition of the system is used whithin the nonlinear MPC (NMPC) framework. For the NMPC algorithm to work, a nonlinear state estimator is needed for which the EKF is an appropriate tool considering the literature in Section 1.4 and our mission objectives in Section 1.7.1.

Some background information suggests that the RNN is emerging as a promising

approach to address the issue in time-consuming online task of solving the quadratic problem obtained from the MPC.

2.2 Dynamical Model

This section considers a 6-DOF model [13] and its corresponding reduced order. The reduced order models are often used since most craft do not have actuation in all DOF. Therefore, a 3-DOF dynamic model that stabilizes the surge, sway and yaw motions is explained in this chapter. This horizontal plane model is suitable for dynamic positioning systems, trajectory tracking control systems and path following systems [13], [177].

Some assumptions that should be considered when modeling an AUV are:

1) The environment in which the AUV is running through, is shallow water. This assumption reduces the hydrodynamics noise that is caused by wind, rain, currents and other environmental sea-life ambient noise. It should be noted that this assumption is made to reduce the ambient noise although ceratin terms for uncertainties are still considered.

2) The vehicle is a rigid body of constant mass ($\dot{m} = 0$). In other words, the vehicle's mass and mass distribution, which is assumed to be homogeneous, do not change during the operation.

3) The center of mass should be considered very close to the center of buoyancy. This way, modeling and controlling the AUV in software would be easier although its power consumption would be far greater.

4) Control surface assumptions: We assume that the control fins do not stall regardless of the angle of attack. We assume an instantaneous fin response, meaning that the vehicle's actuator time response is small in comparison with the time response of the vehicle's attitude.

5) The vehicle is deeply submerged in a homogeneous, unbounded fluid. In other

words, the vehicle is located far from the free surface (no surface effects, *i.e.* wave-making loads), walls and bottom.

6) The vehicle does not experience memory effects. The simulator neglects the effects of the vehicle passing through its own wake.

7) Note that during the straight and level motion, the vehicle operates at a roll offset of negative five degrees ($\theta = -5$) due to the propeller torque. As a result, we never get pure vertical or horizontal-plane motion. Therefore, the vehicle's roll is small enough that it is still possible to identify the vehicle's behavior in pitch and yaw movements. Reynolds number dependencies of the desired motions are not taken into consideration.

2.2.1 Equations of Motion for 6-DOF AUV Model

The AUV equations consist of kinematics (the geometric aspects of motion), rigid-body dynamics (the vehicle inertia matrix) and mechanics (forces and moments causing motion) [178]. Two coordinate frames should be considered in modeling the AUV as a rigid body subject to external forces and torques while moving in a fluid environment.

The Earth-fixed coordinate frame {U} composed of the orthonormal axes (X_U, Y_U, Z_U) and the body-fixed coordinate frame {B} (also known as the moving coordinate frame) composed of the axes (X_B, Y_B, Z_B). The {B} frame is fixed to the vehicle and its axes are aligned with the principal axes of inertia. Figure 2.1 illustrates the cordinate frames, motion components and few navigation terminologies briefly. The interested reader can find more details in [9, 13, 177, 178].

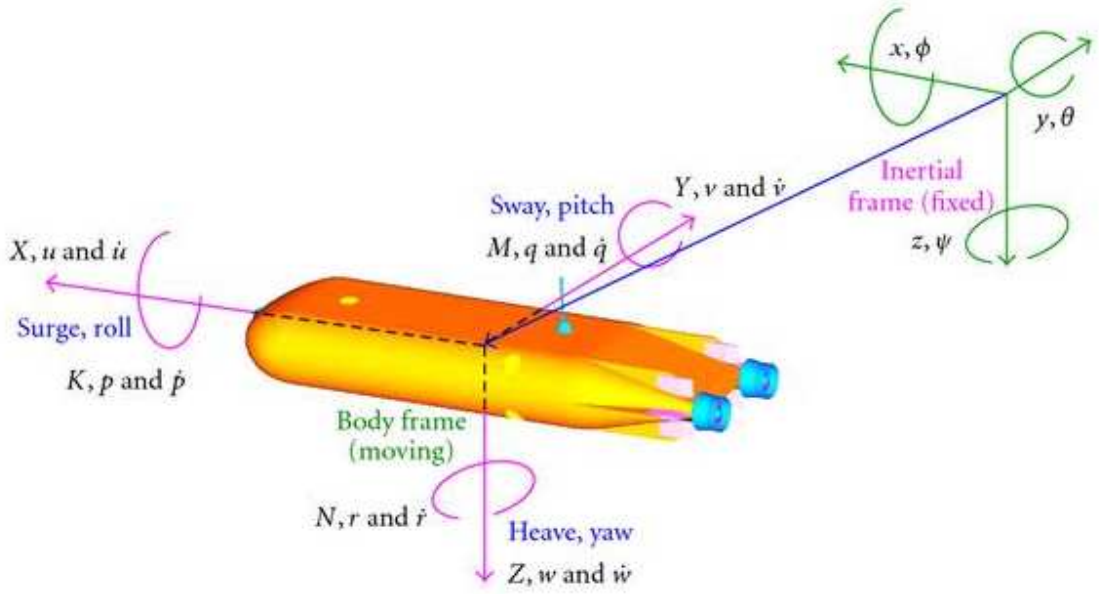


Figure 2.1: Coordinate frames and AUV motion variables [9].

Table 2.1: 6DOF Motion Components [13].

Motion components used for 6DOF marine vehicles and flight systems				
DOF	Description	Linear and Angular Velocity	Position and Euler Angles	External Forces
1	Motion along the x-axis *(surge)*	u	x	X
2	Motion along the y-axis (*sway*)	v	y	Y
3	Motion along the z-axis (*heave*)	w	z	Z
4	Rotation about the x-axis (*roll*)	p	ϕ	K
5	Rotation about the Y-axis (*pitch*)	q	θ	M
6	Rotation about the Z-axis (*yaw*)	r	ψ	N

Generally, a 6-DOF coordinate system is necessary to describe any variation in position and orientation including three position coordinates (x, y, z) and three Euler orientation angles (ϕ, θ, ψ). These six components known as surge, sway, heave, roll, pitch, and yaw are illustrated in Table 2.1. The position and velocity vectors are given in the following form:

$$\eta = [\eta_1^T, \eta_2^T]^T \tag{2.1a}$$

$$\nu = [\nu_1^T, \nu_2^T]^T \tag{2.1b}$$

$$\tau = [\tau_1^T, \tau_2^T]^T \tag{2.1c}$$

where $\eta_1 = [x\, y\, z]^T$ denotes the position of the origin of $\{B\}$ expressed in $\{U\}$ (barycenter coordinates in inertial coordinate system [179]), $\eta_2 = [\phi\, \theta\, \psi]^T$ denotes the orientation of $\{B\}$ with respect to $\{U\}$, $\nu_1 = [u\, v\, w]^T$ denotes the linear velocity of $\{B\}$ relative to $\{U\}$,

$\nu_2 = [p\, q\, r]^T$ denotes the angular velocity of $\{B\}$ relative to $\{U\}$, $\tau_1 = [X\, Y\, Z]^T$ denotes the forces acting on $\{B\}$, and $\tau_2 = [K\, M\, N]^T$ denotes the moments acting on $\{B\}$.

2.2.2 Kinematics

A kinematic equation is used to describe the motion of the vehicles body without considering the causes of motion. This equation describes the relation between the body-fixed velocity and the position vector η.

The corresponding equation in the north-east-down (NED) coordinate frame is expressed as follows:

$$\dot{\eta} = J(\eta)\nu, \tag{2.2}$$

where $J(\eta)$ is the Jacobian matrix transforming the velocities from the body-fixed to the earth-fixed frame. The elements of $\dot{\eta}$ are given below

$$\begin{bmatrix} \dot{\eta}_1 \\ \dot{\eta}_2 \end{bmatrix} = \begin{bmatrix} J_1(\eta_2) & 0 \\ 0 & J_2(\eta_2) \end{bmatrix} \begin{bmatrix} \upsilon_1 \\ \upsilon_2 \end{bmatrix},$$

where $J_1(\eta_2)$ and $J_2(\eta_2)$ denote the coordinate transform matrix (rotation matrix and the angular velocity transformation) and they are defined as follows:

$$J_1(\eta_2) = \begin{bmatrix} c(\theta)c(\psi) & -c(\phi)s(\psi) + s(\phi)s(\theta)c(\psi) & s(\phi)s(\psi) + c(\phi)s(\theta)c(\psi) \\ c(\theta)s(\psi) & c(\phi)c(\psi) + s(\phi)s(\theta)c(\psi) & -s(\phi)c(\psi) + c(\phi)s(\theta)s(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix},$$

$$J_2(\eta_2) = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi)/c(\theta) & c(\phi)/c(\theta) \end{bmatrix},$$

where $c()$, $s()$, and $t()$ denote cos(), sin(), and tan(), respectively. It should be noted that the fact that the matrix $J_2(\eta_2)$ is not defined for pitch angles $\theta = \pm 90°$ does not impose a problem in our model since our desired motion in the horizontal plan is limited to $\pm 30°$.

In kinematics, the full-order nonlinear dynamic equation of motion of a vehicle is expressed as follows

$$M\dot{\nu} + C(\nu)\nu + D(\nu) + g(\eta) = \tau \tag{2.3}$$

where M is the inertia matrix, $C(\nu)$ is the matrix of Coriolis and centripetal terms, $D(\nu)$ is the damping matrix, $g(\eta)$ is the vector of gravitational forces and moments, and $\tau = [\tau_1\, \tau_2]$ is the vector of control inputs (force/torque). Both M and $C(\nu)$ include added mass as well as the rigid-body as can be seen in equations (2.4) and (2.5),

$$M \triangleq M_{RB} + M_A \tag{2.4}$$

$$C(\nu) \triangleq C_{RB}(\nu) + C_A(\nu) \tag{2.5}$$

where rigid-body and added mass effects in mass and Coriolis terms are respectively illustrated by M_{RB}, $C_{RB}(\nu)$, M_A, and $C_A(\nu)$. In fact, $D(\nu)$ is composed of four other matrices, namely the potential damping, the skin friction, wave drift damping and damping due to vortex shedding as can be noted in equation (2.6) as:

$$D(\nu) \triangleq D_P(\nu) + D_S(\nu) + D_w(\nu) + D_M(\nu) \tag{2.6}$$

where $D_P(\nu)$ is the radiation-induced potential damping due to forced body oscillations, $D_S(\nu)$ is the linear skin friction due to laminar boundary layers and quadratic skin friction due to turbulent boundary layers, $D_w(\nu)$ is the wave drift damping, and $D_M(\nu)$ is the damping due to vortex shedding (Morisons equation).

Due to the fact that the hydrodynamic damping matrix is real, non-symmetrical and strictly positive for a rigid body moving through an ideal fluid [178], hence,

$$D(\nu) > 0 \quad \forall \nu \in \mathbb{R}^6.$$

The reader is referred to [178] for detailed matrices effect on $D(\nu)$ and the vector of control inputs (force/torque) τ.

The vehicle coefficients (lumped into the elements of M_{RB} , C_{RB}, and τ) are adopted and calculated based on the theory as well as empirical data [178].

2.2.3 Horizontal Motion of a Dynamically Positioned Underwater

A dynamically positioned underwater ($U = 0$) is described by the motion components in surge, sway and yaw, and consequently $\nu = [u, v, r]^T$ and $\eta = [x, y, \psi]^T$. This implies that the dynamics associated with motions in heave, roll and pitch are neglected ($w = p = q = 0$) as seen in equations 2.2 and 2.3 are

$$M\dot{\nu} + C(\nu)\nu + D(\nu) + g(\eta) = \tau$$
$$\dot{\eta} = J(\psi)\nu$$

where

$$J(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

and η is the vector of position and orientation in the inertial frame, $J(\psi)$ is the rotation matrix that is reduced to one principle rotation about z-axis and $\tau = [\tau_u \, \tau_v \, \tau_r]$ is the matrix of forces and moments that act on the surge, sway, and yaw dynamics, respectively, Moreover,

$g(\eta)$ is the vector of unknown nonlinear uncertainties that take model uncertainties into account.

The rigid-body mass and inertia M_{RB}, rigid-body Coriolis and centripetal $C_{RB}(\nu)$, added mass M_A, and added Coriolis, and centripetal $C_A(\nu)$ matrices are given by

$$M_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_G \\ 0 & mx_G & I_z \end{bmatrix}, \tag{2.7a}$$

$$C_{RB}(\nu) = \begin{bmatrix} 0 & 0 & -m(x_G r + v) \\ 0 & 0 & mu \\ m(x_G r + v) & -mu & 0 \end{bmatrix}, \tag{2.7b}$$

$$M_A = \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -Y_{\dot{r}} & -N_{\dot{r}} \end{bmatrix}, \tag{2.7c}$$

$$C_A(\nu) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}} v + Y_{\dot{r}} r \\ 0 & 0 & -X_{\dot{u}} u \\ -Y_{\dot{v}} v + Y_{\dot{r}} r & X_{\dot{u}} u & 0 \end{bmatrix}, \tag{2.7d}$$

Hence, applying equations (2.4), (2.5), (2.7a), and (2.7b) into equations (2.7c) and (2.7d)

leads to the matrices of M and $C(\nu)$ as follows:

$$M = \begin{bmatrix} m - X_{\dot{u}}u & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_G - Y_{\dot{r}} \\ 0 & mx_G - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix}, \tag{2.8a}$$

$$C(\nu) = \begin{bmatrix} 0 & 0 & -(m - Y_{\dot{v}})v - (mx_G r - Y_{\dot{r}})r \\ 0 & 0 & (m - X_{\dot{u}})u \\ (m - Y_{\dot{v}})v + (mx_G + Y_{\dot{r}})r & -(m - X_{\dot{u}})u & 0 \end{bmatrix}, \tag{2.8b}$$

where x_G, y_G and z_G denote the body-fixed coordinates of the vehicle's center of gravity on the surge, sway, and yaw axes, respectively, I_z denotes the moment of inertia of the vehicle about the yaw axis.

Note that in the above equations, the body's inertia tensor I_0, known as an arbitrary body-fixed coordinate system $X_0Y_0Z_0$ with the origin O in the body-fixed frame. The inertia tensor I_0 is defined as follows:

$$I_0 \triangleq \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yz} & I_y & -I_{yz} \\ -I_{zx} & -I_{zy} & I_z \end{bmatrix}, \qquad I_0 = I_0^T > 0 \tag{2.9}$$

In equation (2.9), diagonal entities are the moments of inertia about the origin axis and off diagonal entities are the products of inertia. The vehicle cross-products of inertia I_{xy}, I_{xz} and I_{yz} are assumed to be small and are neglected in the terms of equation (2.3). Similarly, equation (2.3) does not include zero-valued coefficients. Moreover, the coincidence of the center of gravity and the center of added mass leads to a simplified M and $C(\nu)$, and also

results in the decoupling of surge from sway and yaw.

In [178], it is assumed that damping in surge is decoupled from the sway and the yaw, and hence the linear damping D_{ln} is the only effective damping component. Therefore, in case of the linear damping, the matrix D_{ln} is given as follows:

$$D_{ln} = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix}$$

The nonlinear damping matrix D_n is modeled as follows:

$$D_n = \begin{bmatrix} -X_{|u|u}|u| & 0 & 0 \\ 0 & -Y_{|v|v}|v| - Y_{|r|v}|r| & -Y_{|r|v} \\ 0 & -N_{|v|v}|v| - N_{|r|v}|r| & -N_{|r|v}|r| - N_{|r|r}|r| \end{bmatrix}$$

For more information on matrices D_{ln} and D_n, the interested reader is referred to the work [178].

Finally, the detailed kinematics and dynamics equations of motion in the surge, sway, and yaw degrees of freedom is defined in equation (2.10)

$$\dot{x} = u\cos(\psi) - v\sin(\psi) \tag{2.10a}$$

$$\dot{y} = u\sin(\psi) + v\cos(\psi) \tag{2.10b}$$

$$\dot{\psi} = r \tag{2.10c}$$

$$m_u\dot{u} - m_v vr + d_u u = \tau_u \tag{2.10d}$$

$$m_v\dot{v} - m_u ur + d_v v = \tau_v \tag{2.10e}$$

$$m_r\dot{r} + (m_v - m_u)uv + d_r r = \tau_r \tag{2.10f}$$

where $m_u = m - X_{\dot{u}}$, $m_v = m - Y_{\dot{v}}$, $m_r = I_z - N_{\dot{r}}$, $d_u = -X_u - X_{|u|u}|u|$, $d_v = -Y_v - Y_{|v|v}|v|$

and $d_r = -N_r - N_{|r|r}|r|$. The interested reader is referred to [116] for additional information on the parameters in the equation (2.10).

2.3 Uncertainties in the Model

The possible uncertainties that are counted as an effect on the dynamical model include vehicle's initial conditions as well as the ocean currents. The most significant uncertainty is the vehicle's state at the start of each experiment objective. In fact, we are unable to measure currents, wave effects, and non-axial vehicle velocities, which would have all affected the vehicle's motion during open-loop maneuvers.

There exists few uncertainties that are caused by human mistakes or flaws in the hardware assembling process of AUV. Therefore, it is feasible to allocate additional terms for these uncertainties too. For instance, although the alignment of vehicle fins is checked before each experiment mission, it is difficult to keep the vehicle control fins from knocking during vehicle's transportation and lunch. Therefore, fin misalignment as large as five degrees is inevitable [180].

Any uncertainty in the water-column temperature and salinity comes from two sources, namely the sensors inherent accuracy and the aliased environmental variability. These variations occur in the water column both spatially and temporally. Most of the works done on oil and gas commercial AUV models include less uncertainty terms, unlike the reality of working in the shallow (less than 200 meters depth) water.

In [17], a Total Propagated Uncertainty (TPU) modeling for AUVs is described, which is fundamentally similar to the Hare-Godin-Mayer model [24] [181]. The TPU model provides an estimate of the total horizontal uncertainty (THU) and the total vertical uncertainty (TVU) for every seafloor depth value. This way, elements contributing to the calculation of seafloor depth at a specific location are considered and their corresponding uncertainties are

separately measured or estimated and propagated using the law of propagation of variances. Therefore, the total horizontal and total vertical uncertainty estimates are produced. For simplicity in this model, it is assumed that component uncertainties are uncorrelated. In practice, the TPU values are used to characterize the quality of the data in order to assist in decision making about the suitability of the data for its intended purpose [87]. Next, the effect of ocean currents is applied to the dynamical model via a 3 dimensional (3-D) current model for submerged body [178].

2.3.1 The 3-D Current Model

Considering the vertical profile $V_z(z)$ and the hull draft T, the average current velocity V_c over the draft of the vehicle can be evaluated as follows:

$$V_c = \frac{1}{T}\int_0^T V_z(z)\mathrm{d}z \tag{2.11}$$

The earth-fixed fluid velocity V_c^E can be related to V_c by the angle of attack α and the sideslip angle β. These two angles describe the orientation of V_c about the y and z axis as follows:

$$\begin{aligned} u_c^E &= V_c\cos(\alpha)\cos(\beta) \\ v_c^E &= V_c\sin(\beta) \\ w_c^E &= V_c\sin(\alpha)\cos(\beta) \end{aligned}$$

where u_c^E, v_c^E, and w_c^E denote the earth-fixed fluid velocities along X, Y, and Z axes.

2.3.2 The 2-D Current Model

Considering the horizontal motion of the vehicle, the 3-D expressions in equation (2.12) reduce to two components which are described by only two parameters, that are the average current speed V_c and the direction of current β. Figure 2.2 shows the definition of average velocity V_c and direction β of the current for the vehicle.

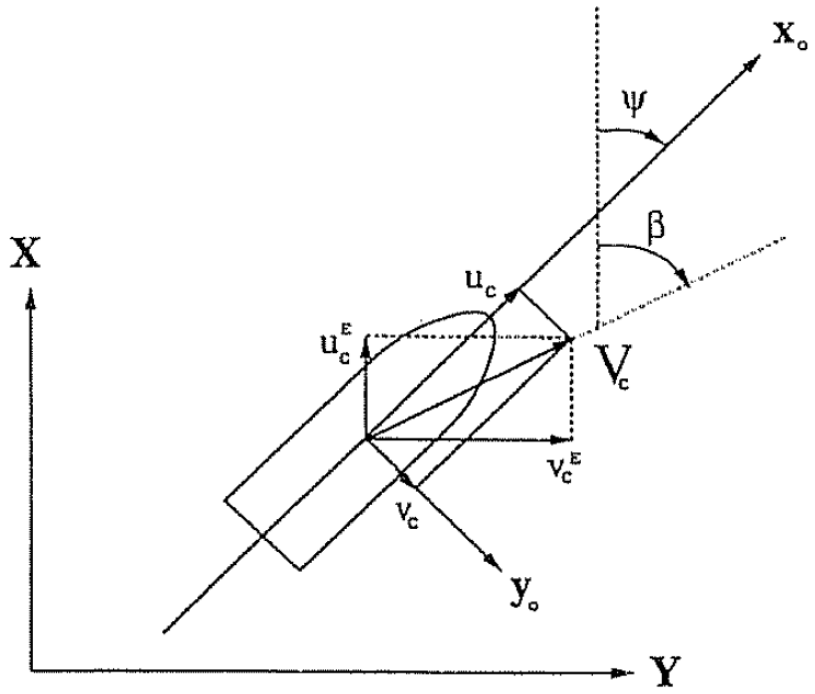


Figure 2.2: Orientation of the vehicle related to the current [10].

Finally, the 2-D current model is given as follows:

$$u_c^E = V_c \cos(\beta)$$

$$v_c^E = V_c \sin(\beta)$$

Since in 2-D motion $\alpha = \theta = 0$, hence, the average current speed u_c and v_c are computed

as follows:

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} u_c^E \\ v_c^E \end{bmatrix} \tag{2.14}$$

Integrating equation (2.13) into equation (2.14) yields:

$$u_c = V_c \cos(\beta - \psi)$$
$$v_c = V_c \sin(\beta - \psi)$$

2.3.3 Considering the Effect of Ocean Currents

Generally, there are two methods that consider the effect of ocean currents in our application. The first method introduces the dynamic equation in terms of the relative velocity. This way, the earth-fixed current velocity is modeled as a Gauss-Markov process as follows:

$$\dot{V}_c(t) + \mu_0 V_c(t) = \omega(t) \tag{2.16}$$
$$V_{c,min} \leq V_c(t) \leq V_{c,max}$$

where $\omega(t)$ is a zero mean Gaussian white noise and $\mu_0 \geq 0$ is a constant. According to [10], it is sufficient to choose $\mu_0 = 0$ in most cases, which simply corresponds to a random walk defined as the time integration of white noise. The constraint of the equation (2.16) limits the process in order to simulate realistic ocean currents. Therefore, the dynamical model of AUV with relative velocity, ν_r, is given as follows:

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

where $\nu = [\nu_u - \nu_{uc}^B, \nu_v - \nu_{vc}^B, \nu_r]$, and the body-fixed current velocity $\nu_c^B = [\nu_{uc}^B \ \nu_{vc}^B \ 0]$ denotes the current velocity components in surge, sway and yaw, respectively. Using the transposed

Euler angle rotation matrix, ν_c^B is obtained as follows

$$\nu_c^B = J(\psi)^{-1}\nu_c^E \tag{2.17}$$

Hence, the kinematic model for AUV in terms of relative velocity ν_r would be

$$\dot{\eta} = J(\psi)\nu = J(\psi)\nu_r + \nu_c^E \tag{2.18}$$

In the second method, which is used as a basis for derivation of model based controllers for path following and tracking of AUVs, the effect of ocean currents is considered through input channels in the simulation step.

Due to the effect of second-order disturbances, ocean currents as drift forces in X, Y, and Z axes are modeled by slowly varying drift $d_c = [d_{cx}\ \ d_{cy}\ \ 0]$ (d_{cx} and d_{cy} are drift along X and Y axes), such that:

$$\dot{d_c} = \omega_d \tag{2.19}$$

where ω_d describes constant, but unknown, environmental forces acting on the system. ω_d is a vector of zero mean Gaussian white noise process, According to the superposition law, second-order disturbances are combined with the original model to yield the following:

$$M\dot{\nu} + C(\nu)\nu + D(\nu) + g(\eta) = \tau + J(\psi)^{-1}d_c, \tag{2.20}$$

$$\dot{\eta} = J(\psi)v, \tag{2.21}$$

Also, first-order disturbances, which describe the high frequency oscillatory motion of a marine vehicle, are included in the measurement equations implicitly.

Apart from what mentioned in this section, any fluctuations in fraction drag or pressure drag numbers results in the change of the whole model since they are related to the Reynolds number and compressibility effects.

2.4 Fault Types in AUV Actuators

Faults are classified based on several criteria, such as the time characteristics of faults, physical locations in the system and the effects of faults on system performance [182]. From the time-dependency perspective, faults are categorized as abrupt (stepwise), incipient and intermittent faults. Faults can also be classified based on their locations. A fault can occur in actuator, sensor or plant component. In terms of induced effects of the faults on system performance, faults can be either additive or multiplicative.

Actuator faults have serious consequences on the AUV system performance and may lead to system malfunction or failure. Actuator faults are commonly modeled by incorporating their effect through multiplicative modeling that is defined as an abrupt change of the control input u to u_f, (the actual input generated by the faulty actuator). The effectiveness coefficient Γ matrix of the actuator control parameters is expressed as follows [116]:

$$\begin{cases} \Gamma u & \Gamma = 1, \forall t \geq 0 & \text{No Failure} \\ \Gamma u & 0 < \epsilon < \Gamma < 1, \forall t \geq t_f & \text{Loss of Effectiveness (LOE)} \\ \Gamma u & \Gamma = 0, \forall t \geq t_f & \text{Float} \\ \Gamma u + u^{Lock} & \Gamma = 0, \forall t \geq t_f & \text{Lock In Place (LIP)} \\ \Gamma u + u^{min} \text{ or } u^{max} & \Gamma = 0, \forall t \geq t_f & \text{Hard Over Failure (HOF)} \end{cases} \tag{2.22}$$

where u denotes the controller input, t_f is the time that a fault occurs, u_f is the actual input that is generated by the faulty actuator and u^{Lock} is a constant level of actuation being between the minimum and maximum possible actuation limits. Γ is represented by

the multiplicative matrix as follows:

$$\Gamma = \begin{bmatrix} \gamma^1 & 0 & 0 \\ 0 & \gamma^2 & 0 \\ 0 & 0 & \gamma^3 \end{bmatrix}, \tag{2.23}$$

where $0 < \gamma^k < 1$, $k = 1, \ldots, 3$ denotes the effectiveness factor of the forces and torque in surge, sway and yaw motions.

Later, in Chapter 4 we use the matrix Γ to implement the effects of actuators' loss-of-effectiveness in various scenarios in order to evaluate our developed AFTC method.

2.5 Discrete Extended Kalman Filter (EKF)

For nonlinear MPC algorithms to work, a nonlinear state estimator is needed and the Extended Kalman filter (EKF) is an appropriate tool for state estimation and data reconciliation of nonlinear systems. In most practical applications of interest, the system dynamics and the measurement equations are given by

$$\dot{x} = f(x, u) + \omega \tag{2.24a}$$

$$y = hx + v \tag{2.24b}$$

where

$$f(x, u) = \begin{bmatrix} M\nu^{-1}[\tau - C(\nu)\nu - D(\nu) - g(\eta)] \\ J(\eta)\nu \end{bmatrix}$$

where f and h are known nonlinear functions and ω and v represent the white noise signals of uncorrelated Gaussian random vectors with zero means and covariance matrices Q_c and

R_c. Then, in a discrete-time nonlinear model is given as follows:

$$x(k) = F(x(k-1), u(k-1)) + \omega(k-1) \tag{2.26a}$$

$$y(k) = Hx(k) + v(k) \tag{2.26b}$$

The nonlinear system in equations (2.26a) and (2.26b) are subject to the following constraints:

$$u_{min} \leq u(k) \leq u_{max} \tag{2.27a}$$

$$\Delta u_{min} \leq \Delta u(k) \leq \Delta u_{max} \tag{2.27b}$$

$$y_{min} \leq y(k) \leq y_{max} \tag{2.27c}$$

where $x(k) \in \mathbb{R}^n$ is the state vector, $u(k) \in \mathbb{R}^m$ is the input vector, $y(k) \in \mathbb{R}^p$ is the output vector, $F(.)$ and $H(.)$ are nonlinear functions, and $u_{min} \leq u_{max}$, $\Delta u_{min} \leq \Delta u_{max}$, $y_{min} \leq y_{max}$ are vectors of lower and upper bounds. As a result, nonlinearity can come in either through process model, equation (2.26a), and/or through the measurement model, equation (2.26b). In practice, the system model in equation (2.26a) is of continuous-time nature. However, the measurements in equation (2.26b) are available through the common digital data-acquisition systems at discrete measurement time instants. Therefore, an efficient formulation of the algorithm is needed to be made for a real-time practical application in order to minimize the filter process time, while obtaining a reasonable accuracy in the filter implementation.

The EKF computes the state estimates at each sampling instance by using the Kalman filter on the linearized approximation of the nonlinear system model. If the noise is white Gaussian and a large neighborhood exists in which the linearizion is a reasonable approximation of true model, then the optimal linear estimate will be an accurate approximation

of the nonlinear state estimate. For example, the work [183] uses the first-order EKF implementation in which the nonlinear system is linearized around the current state estimate using the first-order Taylors series approximation. In [183], Bhonsale *et al.* present an open-source python-based simulation environment, known as SolACE, which enables even non-experts to easily formulate the control (and estimation) problems. Summarizing the different steps needed for the efficient implementation of the discrete time EKF is presented in the Algorithm 1, where $\hat{}$ denotes the estimation of x.

Algorithm 1: discrete Extended Kalman Filter [184].

Input: $\hat{x}(k-1)$ `// a nominal reference trajectory from equation (2.26a) without system noise`

Output: $\hat{x}(k), P_k$ `// The optimal process state estimate and its corresponding error covariance matrix`

Initialization

Time update ("predict");

Project the state ahead; `//` $\bar{\hat{x}}(k) = \bar{\hat{x}}(k-1) + T_s f(\bar{\hat{x}}(k-1))$

foreach *EKF Process* **do**

Update the system covariance matrix

$A_k = \frac{\partial f}{\partial x}|_{x=\hat{x}(k)}$

$\phi = I + T_s A_k$

$Q_d = (\phi Q_c \phi^T + Q_c)\frac{T_s}{2}$

Project the error covariance ahead

$(P_k)^i = (\phi P_{k-1}\phi^T + Q_d)$

Mesurement update ("correct")

Compute the Kalman gain

$H_k = \frac{\partial h}{\partial x}|_{x=\hat{x}(k)}$

$K_k = \bar{P}_k H^T (H\bar{P}_k H^T + R_c)^{-1}$

Update estimate with measurement Y_k

$\hat{x}(k) = \bar{\hat{x}}(k) + K_k(Y_k - H\bar{\hat{x}}(k))$

Update the error covariance

$P_k = \bar{P}_k - K_k H \bar{P}_k$

end

The EKF algorithm 1 gives the optimal system state estimate $\hat{x}(k)$ and its corresponding error covariance matrix as the main two outcomes. The EKF algorithm requires the measurement covariance matrix R_c, the system covariance matrix Q_c, the observation matrix H_k, and the state transition matrix A_k. In fact, A_k and H_k are the Jacobin matrices, derived from the actual nonlinear state-space models in equations (2.26a) and (2.26b), which depend on the most recent state estimation. The measurement error covariance matrix, R_c, indicates the intrinsic quality of the available measuring devices. The larger the covariance, the more quickly older data are discarded. Thus, increasing the corresponding element in matrix R_c for a non-measured variable forces the EKF to estimate the variable using the other output sensor values.

The first-order Euler integration technique is used for numerical integration of the system model from one sample time to the next. The time propagation equation for the state covariance matrix P_k can be solved using the transition matrix technique [185]. This method preserves both the symmetry and the positive definitions of matrix P_k, and yields an adequate performance. Consequently, any possible time-varying dynamic variations in the process or measurement model equations can be introduced in the state estimation procedure.

2.6 Model Predictive Control Formulation

The optimization problem that is solved at each step of MPC is actually a planning exercise which meant to ensure that the current action does not neglect the future. In Linear MPC problem, assuming the discrete-time linear dynamical model of the system is given by

$$x(k+1) = Ax(k) + Bu(k) \tag{2.28}$$

$$z(k) = Cx(k) \tag{2.29}$$

where $x(k) \in \mathbb{R}^n$ is the state vector, $u(k) \in \mathbb{R}^m$ is the input vector, $z(k) \in \mathbb{R}^p$ is the output vector. Then, the MPC can be formulated by introducing the following open-loop optimization problem at every time interval k,

$$\min_{u(k)} J(u(k), x(k))$$

subject to:

$$x(k+p+1|k) = Ax(k+p|k) + Bu(k+p|k) \tag{2.30a}$$

$$z(k+p|k) = Cx(k+p|k) \tag{2.30b}$$

$$z_{min} \leq z(k+p|k) \leq z_{max}, p = 0, \ldots, N_u - 1 \tag{2.30c}$$

$$u_{min} \leq u(k+p|k) \leq u_{max}, p = 0, \ldots, N - 1 \tag{2.30d}$$

Therefore, the performance index is defined as:

$$\min J(u(.), x(k)) = \sum_{p=1}^{N_u - 1} [z^T(k+p|k) Q_p z(k+p|k) \tag{2.31}$$

$$+u^T(k+p|k) R_p u(k+p|k)] + x^T(k+N_u|k) Q_N x(k+N_u|k) \tag{2.32}$$

where $Q \in \mathbb{R}^{p \times p}$ and $Q_N \in \mathbb{R}^{p \times p}$ are positive semi-definite, and $R \in \mathbb{R}^{m \times m}$ is positive-definite penalty matrices, N and N_u denote the prediction horizon ($1 \leq N$) and the control horizon ($0 < N_u < N$), respectively. In Section 1.3, the literature on several approaches for solving the minimization problem in 2.32 through QP problem formulation of MPC is provided. Also, in the literature there are no sufficiently fast and reliable optimization algorithms for nonlinear MPC problems. More details on the MPC is available in [186]. The reliable optimization algorithms that would be able to determine the global optimal solution within a predefined time (each time interval) are not practical in online control applications. Chapter 3 deals with this problem using an observer-based nonlinear prediction linearized

optimization MPC-RNN algorithm in the AUV path-tracking problem.

2.7 Nonlinear Prediction in Model Predictive Control

Nonlinear prediction is the first step in the nonlinear model predictive control approach [186]. Considering the conventional nonlinear MPC formulation at each consecutive sampling instant, k, a set of future control increments is obtained as follows:

$$\Delta u(k) = [\Delta u(k|k) \ \dots \ \Delta u(k+N_u-1|k)]^T \tag{2.33}$$

and the following quadratic cost function is typically used.

$$J(k) = \sum_{p=1}^{N} Q_p(y^{ref}(k+p|k) - \hat{y}(k+p|k))^2 + \sum_{p=1}^{N_u-1} R_p(\Delta u(k+p|k))^2 \tag{2.34}$$

where $\Delta u(k+j|k)$ denotes the input increment, and $\Delta u(k+p|k) = u(k+p|k) - u(k+p-1|k)$, $Q_p > 0$ and $R_p > 0$ are weighting matrices, it is assumed that $\Delta u(k+p|k) = 0$ for $p \geq N_u$, The objective is to minimize differences between the reference trajectory of output signal $y^{ref}(k+p|k)$ and predicted values of the output $\hat{y}(k+p|k)$ over the prediction horizon $N > N_u$, as well as considering that fact that the excessive control increments should be penalized. In fact, only the first element of the determined sequence in equation (2.33) is applied to the system.

$$u(k) = \Delta u(k|k) + u(k-1) \tag{2.35}$$

At the next sampling instant, $k+1$, the prediction is shifted one step forward, the output measurement is updated, and finally the whole procedure is repeated. Since problem constraints have to be usually taken into account, future control increments are determined

from the following optimization problem.

$$\min_{\Delta u(k|k)\,\ldots\,\Delta u(k+N_u-1|k)} J(k) \tag{2.36}$$

subject to the following constraints:

$$u_{min} \leq u(k+p|k) \leq u_{max}, p = 0, \ldots, N_u - 1 \tag{2.37a}$$

$$\Delta u_{min} \leq \Delta u(k+p|k) \leq \Delta u_{max}, p = 0, \ldots, N_u - 1 \tag{2.37b}$$

$$y_{min} \leq \hat{y(k+p|k)} \leq y_{max}, p = 0, \ldots, N_u - 1 \tag{2.37c}$$

The general prediction equation for $p = 1, \ldots, N$ is given by

$$\hat{y}(k+p|k) = y(k+p|k) + d(k) \tag{2.38}$$

where quantities $y(k+p|k)$ are calculated from a dynamic model of the system. The Dynamic Model Control (DMC) type disturbance model is used in which the unmeasured disturbance $d(k)$ is assumed to be constant over the prediction horizon [91] [187]. $d(k)$ is estimated from equation (2.39)

$$d(k) = y(k) - y(k|k-1) \tag{2.39}$$

where $y(k)$ is measured whereas $y(k|k-1)$ is calculated from the dynamic model.

Prediction vectors $\hat{y}(k+p|k)$ are nonlinear functions of future control moves [188]. In this case, the nonlinear MPC optimization problem, described in equation (2.34), should be solved in an online manner at each sampling instant. Although in theory such an approach seems to be potentially very precise, it has a limited practical applicability. It is necessary to emphasize that the difficulty of nonlinear MPC optimization problems is two folded. First , it is nonlinear and high performance-runtime demanding. Second, it may be non-convex

and even multi-modal [91].

Assumption [187]: The output prediction $\hat{y}(k)$ is expressed as the sum of a forced trajectory, which depends only on the future and the free trajectory $y^0(k)$, which depends only on the past,

$$\hat{y}(k) = M_d(k)\Delta u(k) + y^0(k) \tag{2.40}$$

where $\hat{y}(k)$ are vectors of length rN and presents the output prediction, r is the number of outputs and N is the prediction horizon, $\Delta u(k)$ is the future input moves that is in the form of equation (2.33), $\Delta u(k)$ is a vector of length mN_u while m is the number of inputs and N_u is the control horizon. The free trajectory $y^0(k)$ is as follows:

$$y^0(k) = [y^0(k+1|k) \ \dots \ y^0(k+N|k)] \tag{2.41}$$

Motivated from [187], the dynamic matrix $M_d(k)$ of dimensionality $rN \times mN_u$ is comprised of step-response coefficients of the linearized model.

$$M_d(k) = \begin{bmatrix} m_1(k) & 0 & \dots & 0 \\ m_2(k) & m_1(k) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ m_N(k) & m_{N-1}(k) & \dots & m_{N-N_u+1}(k) \end{bmatrix} \in \mathbb{R}^{rN_u \times mN_u} \tag{2.42}$$

where each $m_i(k)$ is, basically, the step-response coefficient of the linearized model and has the dimensionality of $r \times m$. Both the free trajectory $y^0(k)$ and the dynamic matrix $M_d(k)$ are calculated online from the system current states.

There exists differences between the suboptimal prediction calculated from equation (2.40) and the optimal prediction determined from the nonlinear model. However, using the suboptimal prediction turns the optimization problem in equation (2.34) to a QP problem

task as follows:

$$\min_{\Delta u(k)} \parallel y_{ref}(k) - M_d(k)\Delta u(k) - y^0(k) \parallel_Q^2 + \parallel \Delta u(k) \parallel_R^2 \tag{2.43a}$$

subject to:

$$u_{min} \leq J^{NPL}\Delta u(k) + u^{NPL}(k) \leq u_{max} \tag{2.43b}$$

$$-\Delta u_{max} \leq \Delta u \leq \Delta u_{max} \tag{2.43c}$$

$$y_{min} \leq M_d(k)\Delta u(k) + y^0(k) \leq y_{max} \tag{2.43d}$$

where y_{ref}, y_{min} and y_{max} are vectors of length N, space and denote the reference trajectory, minimum constraint, and maximum constraint output vectors, respectively. Vectors of minimum and maximum control inputs, and input increments are denoted by u_{min}, u_{max}, and Δu_{max}, respectively, and they are vectors of length N_u, $u^{NPL}(k)$ is an auxiliary vector length Nu and J^{NPL} is an auxiliary matrix, J^{NPL} is the all ones lower triangular matrix of dimensionality $N_u \times N_u$, Q and R are matrices of the sizes $N \times N$ and $N_u \times N_u$, respectively, R is the matrix of weights in MPC given by $R = diag(\lambda_0, \ldots, \lambda_{N_u-1})$.

2.8 Quadratic Programming Problem Solving

One of the tasks in the process of MPC optimization is to solve the QP problem, obtained from the corresponding cost function. Motivated from the works [189] on nonlinear systems, an RNN proposed to solve a strict convex QP problem and its related piecewise equations applied to a Support Vector Machines (SVM). Compared with the existing neural networks for QP, the proposed NN has a one-layer structure with a low model complexity. Moreover, the proposed NN is shown to have a finite-time convergence and exponential convergence.

Considering the following QP problem

$$\begin{aligned} &\min \frac{1}{2}x^T Qx + c^T x \\ &\text{s.t.} \quad b^1 \leq Ax \leq b^2, d^0 \leq x \leq h^0 \end{aligned} \tag{2.44}$$

where $Q \in \mathbb{R}^{n\times n}$ is an asymmetric and positive-definite matrix, $A \in \mathbb{R}^{m\times n}$, $h^0, d^0 \in \mathbb{R}^n$, $b^1, b^2 \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$. Since the objective function is strictly convex, the problem 2.44 has a unique optimal solutionn. The work [189] used a standard optimization technique in which the equation (2.44) is transformed into a piecewise formulation. Let

$$e = \begin{pmatrix} A \\ I \end{pmatrix}, \quad d = \begin{pmatrix} b^1 \\ d^0 \end{pmatrix}, \quad h = \begin{pmatrix} b^2 \\ h^0 \end{pmatrix}$$

where $I \in \mathbb{R}^{n\times n}$ is an identity matrix. Then the problem (2.44) can be re-written as

$$\begin{aligned} &\min \frac{1}{2}x^T Qx + c^T x \\ &\text{s.t.} \quad d \leq Ax \leq h \end{aligned} \tag{2.45}$$

where $d = [d_1, \ldots, d_{n+m}]^T$ and $h = [h_1, \ldots, h_{n+m}]^T$. Consider the Lagrangian formulation of equation (2.45) as follows:

$$L(x, y, \eta) = \frac{1}{2}x^T Qx + c^T x - u^T(ex - \eta), \tag{2.46}$$

where $u \in \mathbb{R}^{n+m}$ is referred to as the Lagrange multiplier and $\eta \in X = \{u \in \mathbb{R}^{n+m} | d \leq u \leq h\}$. Bazaraa et al. [189] in their saddle point theorem, show that x is an optimal solution of equation (2.45) if and only if there exist u^* and η^* satisfying the following condition.

$$L(x^*, u, \eta^*) \leq L(x^*, u^*, \eta^*) \leq L(x, u^*, \eta)$$

That is

$$\begin{aligned}\frac{1}{2}(x^*)^T Q x^* + c^T x^* - u^T(ex^* - \eta^*) &\leq \frac{1}{2}(x^*)^T Q x^* + c^T x^* - (u^*)^T(ex^* - \eta^*) \\ &\leq \frac{1}{2}(x)^T Q x + c^T x - (u^*)^T(ex - \eta) \\ \forall x \in \mathbb{R}^n, u \in \mathbb{R}^{n+m}, \eta \in X & \end{aligned} \tag{2.47}$$

From the first inequality in equation (2.47) it is obtained that

$$(u - u^*)^T(ex^* - \eta^*) \geq 0 \ \forall u \in \mathbb{R}^{n+m} \tag{2.48}$$

Then $ex^* = \eta^*$. From the second inequality in equation (2.47), it is obtained that

$$f(x^*) - f(x) \leq (u^*)^T(\eta^* - \eta) \ \forall x \in \mathbb{R}^n, \eta \in X, \tag{2.49}$$

where $f(x) = \frac{1}{2}x^T Q x + c^T x - (u^*)^T ex$ and for $i = 1, \ldots, n+m$ which is contradictive when $\eta^* = \eta$. Thus for any $x \in \mathbb{R}^n$, we have $f(x^*) - f(x) \leq 0$ and

$$(u^*)^T(\eta^* - \eta) \geq 0\,, \quad \forall \eta \in X.$$

Using the projection formulation from [190], it can be seen that the above inequality can be equivalently represented as

$$\eta^* = P_X(\eta^* - u^*) \tag{2.50}$$

where $P_X(u) = [P_X(u_1), \ldots, P_X(u_{n+m})]$ and for $i = 1, \ldots, n+m$,

$$P_X(u_i) = \begin{cases} d_i & u_i < d_i, \\ u_i & d_i \leq (u_i \leq h_i, \\ h_i & u_i > h_i, \end{cases} \tag{2.51}$$

On the other side, $f(x^*) \leq f(x)$ implies that $\nabla f(x^*) = Qx^* + c - e^T u^* = 0$. Thus x^* is an optimal solution of equation (2.45) if and only if there exist u^* and η^* such that (x^*, u^*, η^*) satisfies

$$\begin{cases} ex = \eta, \\ Qx + c - e^T u = 0, \\ \eta = P_X(\eta - u). \end{cases}$$

Substituting equations (2.48) and (2.49) into the equation (2.50) we have

$$eQ^{-1}(e^T u - c) = P_X(eQ^{-1}(e^T u - c) - u).$$

Then x^* is an optimal solution of equation (2.45) if and only if there exists u^* such that (x^*, u^*) satisfies

$$\begin{cases} eQ^{-1}e^T u + q = P_X(eQ^{-1}e^T u - eQ^{-1}c - u), \\ x = Ru + a, \end{cases}$$

where $R = Q^{-1}e^T$ and $a = -Q^{-1}c$. Therefore, let u^* be a solution of the piecewise equation,

$$Wu + q = P_X(Wu + q - u), \tag{2.52}$$

where $W \in \mathbb{R}^{(n+m)\times(n+m)}$ is a matrix and $q \in \mathbb{R}^{(n+m)}$ is a vector. If $W = eQ^{-1}e^T$ and $q = -eQ^{-1}c$, then $x^* = Ru^* + a$ is the optimal solution of equation (2.44). Hence, it can see that the optimal solution of equation (2.44) can be obtained by solving the piecewise equation (2.52).

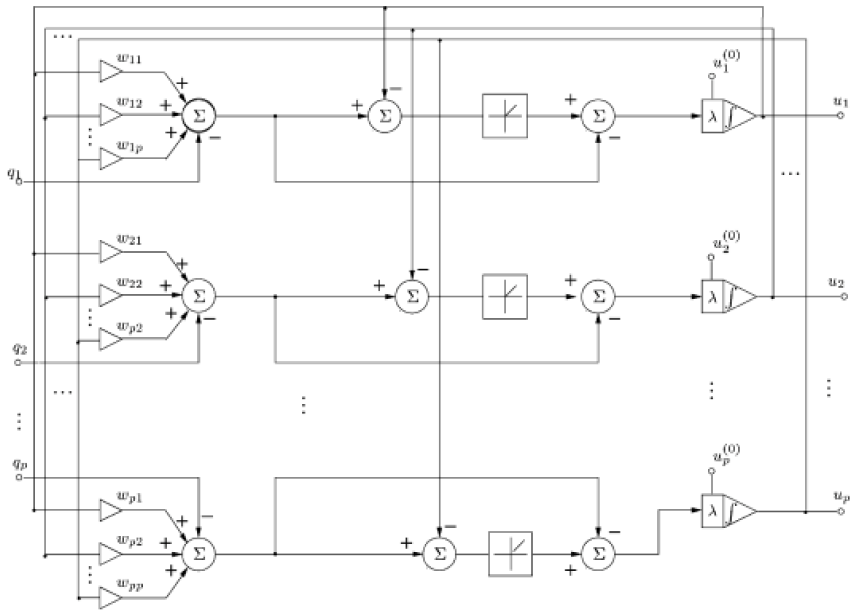


Figure 2.3: Architecture of the state equation defined in equation (2.53) [11].

2.8.1 One-Layer Recurrent Neural Network for Solving Quadratic Programing Problem

In [191], an RNN is proposed that solves both equation (2.44) and its piecewise equation $Wu + q = P_X(Wu + q - u)$. The RNN's dynamical equation is defined as

$$\frac{du}{dt} = \lambda\{P_X(Wu + q - u) - Wu - q\}, \quad \text{(state equation)} \tag{2.53}$$

$$x(t) = Ru(t) + a, \quad \text{(output equation)} \tag{2.54}$$

where $\lambda > 0$ is a scaling constant, $u(t) \in \mathbb{R}^{(n+m)}$ is the state variable, $x(t) \in \mathbb{R}^n$ is the output variable, and R, W, q, a are defined in equation (2.52). Figure 2.3 shows that the state equation (2.53) can be implemented by an analog circuit with a single-layer structure, where $p = m + n$. The circuit consists of $(n + m)^2 + n + m + 1$ summers, $n + m$ integrators, and $(n + m)^2$ weighted connections.

$P_X(z_i)$ as the projection operator may be implemented by using a linear piecewise

function. Following results for the convergence of the proposed RNN in [11], are proved.

- The proposed RNN has a globally convergent state trajectory and it is convergent to the solution of piecewise equation (2.52) within a finite time, if W is symmetric and semi-definite, and is globally exponentially convergent if W is symmetric and definite.

- Within a finite time, the output trajectory of the proposed RNN converges globally to a unique optimal solution of the equation (2.44), if $W = EQ^{-1}E^T$ and $q = -EQ^{-1}c$. Moreover, the output trajectory has a bounded convergence rate

$$\| x(t) - x^* \|^2 \leq \frac{\gamma_{RNN}}{\lambda_{RNN}(t - t_0)}, \quad \forall t > t_0 \tag{2.55}$$

  where $\| \cdot \|$ denotes the l_2 norm, $\| x(t) - x^* \|$ is the future control increment $\Delta u(k)$ from equation (2.33), γ_{RNN} is a positive constant, and $\lambda_{RNN} > 0$ is a scaling constant [191].

The reader is referred to [11] for more details and the comparison among existing NNs and the proposed one. Also, the works [11], [191], and [192] provide the reader with the proof of convergence for the proposed RNN.

2.9 Shortcomings of Previous MPC-RNN Approach in [2]

Wang *et al.* [2] applied an RNN for solving the QP problem in real-time. The first shortcoming of the work in [2] which motivated us to improve the MPC-RNN approach is that, the work was based on a bilinear model whereas in our case, the model is nonlinear as mentioned in the statement of the problem of this thesis.

Wang *et al.* [2] claimed to solve the equation (2.34) as follows:

First, according to the model, Wang *et al.* obtained the following sequence

$$
\begin{aligned}
x(k+1|k) &= f(x(k|k-1)) + g(x(k|k-1)))(u(k-1) + \Delta u(k|k)) \\
x(k+2|k) &= f(x(k+1|k-1)) + g(x(k+1|k-1)))(u(k-1) + \Delta u(k|k) + \Delta u(k+1|k)) \\
&\vdots \\
x(k+N|k) &= f(x(k+N-1|k-1)) \\
&\quad + g(x(k+N-1|k-1))(u(k-1) + \Delta u(k|k) + \cdots + \Delta u(k+N_u-1|k)),
\end{aligned}
$$

Then, the following vectors were defined:

$$\bar{y}^{ref}(k) = [y^{ref}(k+1) \ \ldots \ y^{ref}(k+N)]^T \tag{2.57a}$$

$$\bar{\hat{y}}(k) = [\hat{y}(k+1|k) \ \ldots \ \hat{y}(k+N|k)]^T \tag{2.57b}$$

$$\bar{u}(k) = [u(k|k) \ \ldots \ u(k+N_u-1|k)]^T \tag{2.57c}$$

$$\bar{x}(k) = [x(k+1|k) \ \ldots \ x(k+N|k)]^T \tag{2.57d}$$

$$\Delta\bar{u}(k) = [\Delta u(k|k) \ \ldots \ \Delta u(k+N_u-1|k)]^T \tag{2.57e}$$

where $\bar{y}^{ref}(k)$ denotes the reference trajectory vector of output signal that is known in advance, $\bar{\hat{y}}(k)$ denotes the predicted output vector, $\bar{u}(k)$ denotes the inputs vector, $\bar{x}(k)$

denotes the system states vector, and $\Delta\bar{u}(k)$ denotes the input increment vector,

In the proposed procedure by the work [2], the predicted output $\bar{\hat{y}}(k)$ is expressed in the following form:

$$\bar{\hat{y}}(k) = \tilde{C}\bar{x}(k) = \tilde{C}(G\Delta\bar{u}(k) + \tilde{f} + \tilde{g})$$

where $\tilde{C}, G, \tilde{f}$ and $\tilde{g}$ are matrices that are defined as follows:

$$\tilde{C} = \begin{bmatrix} C & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & C \end{bmatrix} \in \mathbb{R}^{N\times Nn},$$

$$G = \begin{bmatrix} g(x(k|k-1)) & \dots & 0 \\ \vdots & \ddots & \vdots \\ g(x(k+N-1|K-1)) & \dots & g(x(k+N-1|k-1)) \end{bmatrix} \in \mathbb{R}^{Nn\times N_u m},$$

$$\tilde{f} = \begin{bmatrix} f(x(k|k-1)) \\ f(x(k+1|k-1)) \\ \vdots \\ f(x(k+N-1|k-1)) \end{bmatrix} \in \mathbb{R}^{Nn},$$

$$\tilde{g} = \begin{bmatrix} g(x(k|k-1))u(k-1) \\ g(x(k+1|k-1))u(k-1) \\ \vdots \\ g(x(k+N-1|k-1))u(k-1) \end{bmatrix} \in \mathbb{R}^{Nn},$$

Hence, the original optimization problem (equation (2.34) becomes:

$$\min \parallel \bar{y_{ref}}(k) - \tilde{C}\tilde{f} - \tilde{C}\tilde{g} - \tilde{C}G\Delta\bar{u}(k) \parallel_Q^2 + \parallel \Delta\bar{u}(k) \parallel_R^2 \tag{2.58a}$$

s.t.

$$\bar{u}_{min} \leq \bar{u}(k-1) + \tilde{I}\Delta u(k) \leq \bar{u}_{max} \tag{2.58b}$$

$$\Delta\bar{u}_{min} \leq \Delta\bar{u}(k) \leq \Delta\bar{u}_{max} \tag{2.58c}$$

$$\bar{\hat{y}}_{min} \leq \tilde{C}\tilde{f} - \tilde{C}\tilde{g} - \tilde{C}G\Delta\bar{u}(k) \leq \bar{\hat{y}}_{max} \tag{2.58d}$$

where

$$\tilde{I} = \begin{bmatrix} I & 0 & \dots & 0 \\ I & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \dots & I \end{bmatrix} \in \mathbb{R}^{N_u m \times N_u m}.$$

Then, Wang *et al.* [2] rewrote the equation (2.34) as a time-varying QP problem as follows:

$$\min \frac{1}{2}\Delta\bar{u}^T W \Delta\bar{u} + C^T \Delta\bar{u} \tag{2.59}$$

s.t.

$$l \leq E\Delta\bar{u} \leq h$$

where the coefficients in equation (2.59) are defined as follows:

$$W = 2(G^T\tilde{C}^TQ\tilde{C}G + R) \in \mathbb{R}^{N_um\times N_um}$$

$$c = -2G^T\tilde{C}^TQ(\bar{r}(k) - \tilde{C}\tilde{g} - \tilde{C}\tilde{f}) \in \mathbb{R}^{N_um}$$

$$E = [-\tilde{I} \quad \tilde{I} \quad -\tilde{C}G \quad \tilde{C}G \quad \tilde{I}]^T \in \mathbb{R}^{(3N_um+2Np)\times N_um}$$

$$l = [-\infty \quad \Delta\bar{u}_{min}]^T \in \mathbb{R}^{3N_um+2N}$$

$$h = [b \quad \Delta\bar{u}_{max}]^T \in \mathbb{R}^{3N_um+2N}$$

$$b = \begin{bmatrix} -\bar{u}_{min} + \bar{u}(k-1) \\ \bar{u}_{max} - \bar{u}(k-1) \\ -\bar{\hat{y}}_{min} + \tilde{C}\tilde{g} + \tilde{C}\tilde{f} \\ \bar{\hat{y}}_{max} - \tilde{C}\tilde{g} - \tilde{C}\tilde{f} \end{bmatrix} \in \mathbb{R}^{2N_um+2N}$$

Wang *et al.* [2] claimed that, the solution to their QP problem gives optimal control increment vector $\Delta\bar{u}(k)$ whose first Δu can be used to calculate the optimal control input.

The formulation of authors in the work [2], have major issues to be improved. The state predictions in the first step's equations (2.56), are obtained using the information at time step $k-1$ (both f and g are calculated utilizing the state information at time $k-1$). This is incorrect because by applying the control input at time k could change the state profile drastically, hence the state predictions using this framework can be totally incorrect. The substantial effect of this wrong estimation is specially observable when we have $g(x) = 0$ for some x. Therefore, the whole matrix G would be zero and QP optimization would produce $\Delta u = 0$. Moreover, the proposed approach in [2] lacks an applicable method that searches global solution, other than local, in case of applying to nonlinear systems.

2.10 Conclusion

In this chapter, the backgroundl related to this thesis is provided. After bringing important concepts in designing AUVs, the full-order and reduced-order dynamics and modeling of underwater vehicle are adapted from [10]. Then, uncertainties in the model and fault types in AUV application are presented. Also, a discrete extended Kalman filter algorithm followed by model predictive method formulation are introduced. The Recurrent Neural Network adopted in [11] to solve the quadratic programming problem and its peicewise formulation are explained. It is shown that the proposed neural network has a finite-time convergence as well as an exponential convergence for the optimal solution of the piecewise equation. Moreover, the shortcomings of previous litrature on the MPC-RNN based control approaches are discussed in this chapter. Next chapter deals with the problem regarding the MPC optimization using the proposed observer-based MPC approach for an AUV system.

Chapter 3

A Performance-Runtime Efficient Observer-based Nonlinear Model Predictive Control

3.1 Introduction

This chapter aims to bring an alternative solution to Model Predictive Control (MPC) of nonlinear systems. The developed methodology in this chapter benefits from the accuracy of nonlinear optimization approaches in MPC, lower complexity of linear MPC optimization approaches. Moreover, the developed methodology is performance runtime efficient and has adaptivity of computational intelligent hybridized control methods.

In Chapter 1, the importance of using model based methodology has been discussed. In Chapter 2, the nonlinear dynamic model of an autonomous underwater vehicle is described. Then, the fault types regarding AUV actuator are explained. Chapter 1 stated that MPC is explicitly a function of the model that can be modified in real-time (and plan time). Therefore, MPC explicitly accounts for system constraints and can easily handle nonlinear

and time-varying plant dynamics [186]. Also, in Chapter 1 and Chapter 2, the difficulties of working with nonlinear MPC optimization problems are presented.

After bringing the literature on hybrid control approaches containing MPC method, the benefits and importance of development and improvement in this field are demonstrated in Chapter 1. Moreover, in Section 2.9 shortcomings of previous hybrid of MPC and Recurrent neural network control algorithms are presented.

Previous chapters given the literature on nonlinear systems' estimators. Particularly, in Section 1.4.2 observer-based nonlinear MPC algorithm is presented. Due to the AUV application, the objectives of this research, Extended Kalman Filter (EKF) is selected (as stated in Section 1.4). In Chapter 2 motivated by [11], a one layer RNN for solving the convex QP problem and its piecewise formulation are suggested. The RNN is shown to have a finite-time convergence and exponential convergence.

The contribution of this chapter is to formulate the MPC optimization problem with nonlinear prediction from the nonlinear system model, acquiring its corresponding quadratic problem and using the RNN approach, (explained in Section 2.8) to solve the QP problem obtained from the MPC formulation.

In this approach, the system states are gained from Extended Kalman Filter that is applied to the Dynamic Matrix (explained in Section 2.7). The proposed method in this chapter requiressolving a QP problem in an online manner which guarantees finding a control input within each time interval. The aforementioned objectives are shown to be met via the RNN approach motivated from [11]. Consequently, our proposed control method in this chapter combines RNN with the MPC method to avoid high complexity of solving the QP problem as well as reaching a faster convergence time for the system. For the evaluation of the proposed control algorithm in this chapter, three control schemes are considered in Section 3.3 for the same AUV trajectory tracking and path following problem. Our

developed control scheme is compared with the nonlinear MPC method using Levenberg-Marquardt algorithm [193] as well as the linear MPC scheme. Finally, the corresponding simulation results and discussions are presented.

3.2 Formulating the Problem

In this chapter, an efficient algorithm is developed by employing state observers that can be applied to multi-variable systems such as AUV. The developed algorithm utilizes the EKF and it merely requires solution of an online QP problem.

As explained in Section 2.7, in the conventional nonlinear MPC formulation, a set of future control increments is calculated at each consecutive sampling instant k as follows:

$$\Delta u(k) = [\Delta u(k|k) \ \dots \ \Delta u(k+N_u-1|k)]^T, \tag{3.1}$$

and the following quadratic cost function is typically used.

$$J(k) = \sum_{p=1}^{N} Q_p(y^{ref}(k+p|k) - \hat{y}(k+p|k))^2 + \sum_{p=1}^{N_u-1} R_p(\Delta u(k+p|k))^2. \tag{3.2}$$

It is assumed that $\Delta u(k+p|k) = 0$ for $p \geq N_u$, where N and N_u are prediction horizon ($1 \leq N$) and control horizon ($0 < N_u < N$), respectively, $y^{ref}(k+p|k)$ is the reference trajectory of output signal, and $\hat{y}(k+p|k)$ is predicted values of the output over the prediction horizon $N > N_u$.

Objective Minimize the differences between the reference trajectory of output signal $y^{ref}(k+p|k)$ and predicted values of the output $\hat{y}(k+p|k)$ over the prediction horizon $N > N_u$, as well as considering the fact that the excessive control increments should be penalized. $\Delta u(k+j|k)$ denotes the input increment, $\Delta u(k+pk) = u(k+pk) - u(k+p-1k)$, $Q_p > 0$ and $R_p > 0$ are weighting matrices.

In fact, only the first element of the determined sequence in equation (3.1) is applied to the system $u(k) = \Delta u(k|k) + u(k-1)$. At the next sampling instant, $k+1$, the prediction is shifted one step forward, the output measurement is updated, and the whole procedure is repeated. Since problem constraints have to be taken into account, future control increments are determined from the following optimization problem:

$$\min_{\Delta u(k|k) \ldots \Delta u(k+N_u-1|k)} J(k)$$

subject to the following constraints:

$$u_{min} \leq u(k+p|k) \leq u_{max}, p = 0, \ldots, N_u - 1$$
$$\Delta u_{min} \leq \Delta u(k+p|k) \leq \Delta u_{max}, p = 0, \ldots, N_u - 1$$
$$y_{min} \leq \hat{y}(k+p|k) \leq y_{max}, p = 0, \ldots, N_u - 1$$

The general prediction equation for $p = 1, \ldots, N$ is

$$\hat{y}(k+p|k) = y(k+p|k) + d(k)$$

where $y(k+p|k)$ is calculated from a dynamic model of the system. The Dynamic Model Control (DMC) type disturbance model is used in which the unmeasured disturbance $d(k)$ is assumed to be constant over the prediction horizon [91] and [187]. $d(k)$ is estimated as given below:

$$d(k) = y(k) - y(k|k-1)$$

where $y(k)$ is measured while $y(k|k-1)$ is calculated from the dynamic model.

Prediction vectors $\hat{y}(k+p|k)$ are nonlinear functions of future control moves [188]. In such a case, the nonlinear MPC optimization problem (described in equation (3.2)), has to

be solved in an online manner at each sampling instant. In theory, such an approach seems to be potentially very precise, but it has a limited practical applicability.

Then motivated from [187], the output prediction $\hat{y}(k)$ is expressed as the sum of a forced trajectory that depends only on the future and the free trajectory $y^0(k)$, which depends only on the past. $\hat{y}(k)$ is calculated as follows:

$$\hat{y}(k) = M_d(k)\Delta u(k) + y^0(k),$$

where $\hat{y}(k)$ are vectors of length rN presenting an output prediction, r is the number of outputs and N is the prediction horizon, $\Delta u(k)$ is the future input moves and it is in the form of equation (3.1), $\Delta u(k)$ is a vector of length mN_u while m is the number of inputs and N_u is the control horizon. The free trajectory $y^0(k)$ is as follows:

$$y^0(k) = [y^0(k+1|k) \ \dots \ y^0(k+N|k)]$$

The dynamic matrix $M_d(k)$ of dimensionality $rN \times mN_u$ is comprised of step-response coefficients of the linearized model as provided below.

$$M_d(k) = \begin{bmatrix} m_1(k) & 0 & \dots & 0 \\ m_2(k) & m_1(k) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ m_N(k) & m_{N-1}(k) & \dots & m_{N-N_u+1}(k) \end{bmatrix} \in \mathbb{R}^{rN_u \times mN_u}$$

where each $m_i(k)$ is the step-response coefficient of the linearized model and has $r \times m$ dimensionality. Both the free trajectory $y^0(k)$ and the dynamic matrix $M_d(k)$ are calculated online from the current states of system.

The approach this research selected based on the literature is using EKF, discussed in Section 2.5 and Algorithm 1, considering the current state of the system to build both the

free trajectory $y^0(k)$ and the dynamic matrix $M_d(k)$. Therefore, in the linearized model, A and C matrices are obtained using the equations in the Algorithm 1 for A_k and H_k. The matrix B of the linearized model is obtained from $B_k = \frac{\partial f}{\partial u}|_{x=\hat{x}(k)}$. Then, the suboptimal prediction calculated from equation $\hat{y}(k) = M_d(k)\Delta u(k) - y^0(k)$ transforms the optimization problem in equation (3.2) to a quadratic programming problem task as follows:

$$\min_{\Delta u(k)} \parallel y_{ref}(k) - M_d(k)\Delta u(k) - y^0(k) \parallel_Q^2 + \parallel \Delta u(k) \parallel_R^2 \tag{3.4a}$$

subject to

$$u_{min} \leq J^{NPL}\Delta u(k) + u^{NPL}(k) \leq u_{max}$$

$$-\Delta u_{max} \leq \Delta u \leq \Delta u_{max}$$

$$y_{min} \leq M_d(k)\Delta u(k) + y^0(k) \leq y_{max}$$

where y_{ref}, y_{min} and y_{max} are vectors of length N, and present the reference trajectory, minimum constraint, and maximum constraint output vectors, respectively, u_{min}, u_{max}, and Δu_{max} are vectors of minimum and maximum control inputs, and input increments, respectively, Q and R are matrices of the sizes $N \times N$ and $N_u \times N_u$, respectively, R is the matrix of weights in MPC given by $R = diag(\lambda_0, \ldots, \lambda_{N_u-1})$, $u^{NPL}(k)$ is an auxiliary vector length N_u and J^{NPL} is an auxiliary matrix of dimensionality $N_u \times N_u$ as follows:

$$J^{NPL}(i,j) = \begin{cases} 1, \ i = j \ or \ i = j+3 \\ 0, \text{others} \end{cases} \tag{3.5}$$

By using the nonlinear model of system and EKF, the nonlinear prediction of MPC is gained. This thesis considered $x_{QP} = \Delta u^T(k)$ be a vector containing all decision variables of the MPC algorithm, and correspondingly the optimization problem (equation (3.4a)) is

rewritten in a standard QP form as follows:

$$\begin{aligned} \min \frac{1}{2} x_{QP}^T H_{QP} x_{QP} + f_{QP}^T x_{QP} \\ \text{s.t.} \quad M_d^T(k)\Delta u(k) = \hat{y}(k) - y^0(k) \\ A_{QP} x_{QP} \leq b_{QP} \\ -\Delta u_{max}^T \leq x_{QP} \leq \Delta u_{max}^T \\ \text{where} \quad \Delta u^T(k) = x_{QP}, \end{aligned} \tag{3.6}$$

The cost function in the QP problem equation (3.6) is defined by following equations which has a unique optimal solution, as the objective function is strictly convex.

$$H_{QP} = 2({M^T}_d(k) Q M_d(k) + R) \tag{3.7}$$

$$f_{QP} = -2({M_d}^T(k) y^{ref}(k) - y^0(k)) \tag{3.8}$$

where Q and R are matrices of the sizes $N \times N$ and $N_u \times N_u$, respectively. R is the matrix of wights in MPC given by $R = diag(\lambda_0, \ldots, \lambda_{N_u - 1})$. y_{ref} is a vector of length N, and denotes the reference trajectory. The constraints are defined as:

$$A_{QP} = \begin{bmatrix} -J^{NPL} \\ J^{NPL} \\ -CM_d(k) \\ CM_d(k) \\ -I_{N_u \times N_u} \\ I_{N_u \times N_u} \end{bmatrix}, \tag{3.9}$$

$$b_{QP} = \begin{bmatrix} -u_{min} + u_{k-1}(k) \\ u_{max} - u_{k-1}(k) \\ -y_{min} + y^0(k) \\ y_{max} - y^0(k) \\ \Delta u_{max} \\ \Delta u_{max} \end{bmatrix}, \tag{3.10}$$

where $C = I$ and J^{NPL} is an auxiliary matrix obtained from equation (3.5). using a standard optimization technique in which the equation (3.6) is transformed into a piecewise formulation. Let

$$E = \begin{pmatrix} A \\ I \end{pmatrix}, \quad d = \begin{pmatrix} b^1 \\ -\Delta u_{max}^T \end{pmatrix}, \quad h = \begin{pmatrix} b^2 \\ \Delta u_{max}^T \end{pmatrix},$$

where $I \in \mathbb{R}^{n \times n}$ is an identity matrix and A and b^2 are given by

$$A = \begin{pmatrix} A_{QP} \\ M_d^T \end{pmatrix}, \quad b^2 = \begin{pmatrix} \hat{y}(k) - y^0(k) \\ b_{QP} \end{pmatrix}$$

Then the problem (3.6) can be re-written as

$$\begin{aligned} \min \frac{1}{2} x_{QP}^T H_{QP} x_{QP} + f_{QP}^T x_{QP} \\ \text{s.t.} \quad d \leq E x_{QP} \leq h \end{aligned} \tag{3.11}$$

where $d = [d_1, \ldots, d_N]^T$ and $h = [h_1, \ldots, h_N]^T$. Consider the Lagrangian formulation of

equation (3.11) as follows:

$$L(x, y, \eta) = \frac{1}{2} x_{QP}^T H_{QP} x_{QP} + f_{QP}^T x_{QP} - u^T (E x_{QP} - \eta), \tag{3.12}$$

where $u \in \mathbb{R}^{n+m}$ is referred to as the Lagrange multiplier and $\eta \in X = \{u \in \mathbb{R}^{n+m} | d \leq u \leq h\}$. where $x_{QP} = \Delta u^T(k)$ denotes the vector containing all decision variables of the MPC algorithm, A_{QP} given in (3.9), $H_{QP} \in R^{N \times N}$ is positive-definite and $H_{QP} = 2({M^T}_d(k) Q M_d(k) + R)$, and $f_{QP} = -2({M_d}^T(k) y^{ref}(k) - y^0(k))$. Bazaraa et al. [189] in their saddle point theorem, show that x_{QP} is an optimal solution of equation (3.11) if and only if there exist u^* and η^* satisfying the following condition.

$$L(x_{QP}^*, u, \eta^*) \leq L(x_{QP}^*, u^*, \eta^*) \leq L(x_{QP}, u^*, \eta)$$

Similar to the Section 2.8, using the projection formulation form [190], it can be seen that the above inequality can be equivalently represented as

$$\eta^* = P_{X,QP}(\eta^* - u^*), \tag{3.13}$$

where the projection operator $P_{X,QP}(u) = [P_{X,QP}(u_1), \ldots, P_{X,QP}(u_N)]$ and for $i = 1, \ldots, N$,

$$P_{X,QP}(u_i) = \begin{cases} d_i & u_i < d_i, \\ u_i & d_i \leq u_i \leq h_i, \\ h_i & u_i > h_i, \end{cases}$$

Thus x_{QP}^* is an optimal solution of equation (3.11) if and only if there exist u^* and η^* such

that (x^*_{QP}, u^*, η^*) satisfies

$$\begin{cases} Ex_{QP} = \eta, \\ H_{QP}x_{QP} + f_{QP} - E^T u = 0, \\ \eta = P_{X,QP}(\eta - u). \end{cases}$$

Substituting the above first and second equations into the third equation, we have

$$EH^{-1}_{QP}(E^T u - f_{QP}) = P_{X,QP}(EH^{-1}_{QP}(E^T u - f_{QP}) - u).$$

Then x^*_{QP} is an optimal solution of equation (3.11) if and only if there exists u^* such that (x^*_{QP}, u^*) satisfies

$$\begin{cases} EH^{-1}_{QP}E^T u + q = P_{X,QP}(EH^{-1}_{QP}E^T u - EH^{-1}_{QP}f_{QP} - u), \\ x_{QP} = R_y u + a, \end{cases}$$

where $R_y = H^{-1}_{QP}E^T$ and $a = -H^{-1}_{QP}f_{QP}$. Therefore, let u^* be a solution of the piecewise equation,

$$Wu + q = P_{X,QP}(Wu + q - u), \tag{3.14}$$

where $W \in \mathbb{R}^{N\times N}$ is a matrix and $q \in \mathbb{R}^N$ is a vector. If $W = EH^{-1}_{QP}E^T$ and $q = -EH^{-1}_{QP}f_{QP}$, then $x^*_{QP} = R_y u^* + a$ is the optimal solution of equation (3.6). Hence, it can see that the optimal solution of equation (3.6) can be obtained by solving the piecewise equation (3.14).

Therefore, according to the Section 2.8.1, an RNN is proposed that solves both equation (3.6) and its piecewise equation $Wu + q = P_{X,QP}(Wu + q - u)$. The proposed RNN's

dynamical equation is defined as

$$\frac{du}{dt} = \lambda_{RNN}\{P_{X,QP}(Wu + q - u) - Wu - q\}, \quad \text{(state equation)} \tag{3.15}$$

$$x_{QP}(t) = R_y u(t) + a, \quad \text{(output equation)} \tag{3.16}$$

where $\lambda_{RNN} > 0$ is a scaling constant, $u(t) \in \mathbb{R}^N$ is the state variable, $x_{QP}(t) \in \mathbb{R}^n$ is the output variable, and R_y, W, q, a are defined in equation (3.14). In [191], presents that the one later RNN has a single- layer structure with totally $3N_u m + 2N$ neurons. In the Section 2.8.1, it is illustrated that our proposed RNN has a globally convergent state trajectory and it is convergent to the solution of piecewise equation (3.14) within a finite time, if W is symmetric and semi-definite, and is globally exponentially convergent if W is symmetric and definite. Moreover, within a finite time, the output trajectory of our proposed RNN converges globally to a unique optimal solution of the equation (3.6), if $W = EH_{QP}^{-1}E^T$ and $q = -EH_{QP}^{-1}c$. Moreover, the output trajectory has a bounded convergence rate

$$\| x_{QP}(t) - x_{QP}^* \|^2 \leq \frac{\gamma_{RNN}}{\lambda_{RNN}(t - t_0)}, \quad \forall t > t_0 \tag{3.17}$$

where $\| \cdot \|$ denotes the l_2 norm, $\| x_{QP}(t) - x_{QP}^* \|$ is the future control increment $\Delta u(k)$ from equation (3.1), γ_{RNN} is a positive constant, and $\lambda_{RNN} > 0$ is a scaling constant [191].

Hence, according to the work [191], given in the Section 2.7, the developed RNN provides the controller with the vector of future decisions, so that the controller applies the first element of this vector in order to control the AUV. The proposed RNN guaranteed to find the solution to the equation (3.2) at each time interval k (Section 2.8.1). The implementation of our proposed hybrid of MPC and RNN control method is described in Figure 3.1.

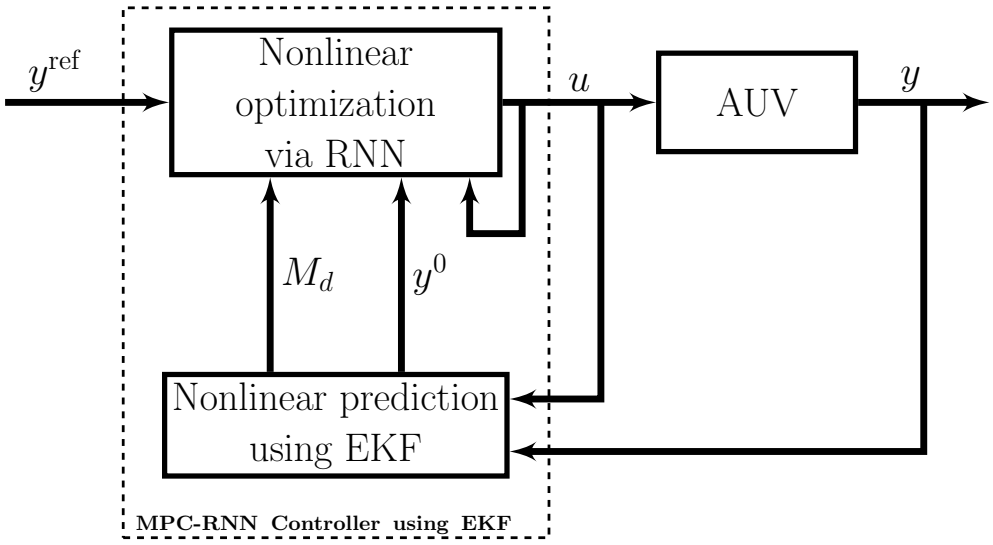


Figure 3.1: The developed MPC-RNN methodology.

3.2.1 The Overall Developed MPC-RNN Control Scheme

The following steps should be repeated at each sampling instant for the MPC-RNN algorithm to work:

- Let $k = 1$ and set the control time terminal T, prediction horizon N, control horizon N_u, control time terminal T, sampling interval T_s, weight matrices Q and R.
- Calculate the nonlinear free trajectory $y^0(k)$ given in equation (2.41) using EKF.
- Calculate the dynamic matrix $M_d(k)$. Set NN parameters, H_{QP}, f_{QP}, A_{QP}, R_y, q and W (given in equations (3.7), (3.8), (3.9), and (3.14)).
- Solve the convex quadratic minimization problem given in equation (3.6) to obtain the optimal control action Δu_k by using an RNN with $3N_u m + 2N$ neurons, where m denotes the number of system inputs and n denotes the number of system states.
- Apply the optimal input vector $u(k)$ given in equation (2.35).
- If $k < T$, set $k = k + 1$ and go to the second step; otherwise end.

The proposed method in this chapter achieves its stability by proper tuning of the prediction horizon and weighting coefficients λ_{RNN}.

3.3 Comparative Methods

This section uses the tracking dynamics of a nonlinear model to illustrate the application of this study. Simulation results are given and discussed to demonstrate the effectiveness of the proposed MPC-RNN scheme for AUV control application.

In previous chapters, it was discussed that the nonlinear MPC has higher online time, requiring a larger online memory. Also,higher number of variables are required (real and integer) refers to the programming features, results in higher computation time and memory requirements for implementing of nonlinear MPC. Moreover, a nonlinear MPC is generally not guaranteed to converge within a reasonable computing time. Hence, a nonlinear MPC method may not be robust for some cases within an online process control. On the other hand, the linear MPC is shown to have the least position and velocity overshoot values and from the control cost perspective, the most approximate behavior (including linear MPC) with lower cost.

In order to evaluate of the proposed nonlinear MPC-RNN control algorithm, three control systems are considered here:

- Linear MPC using Kalman Filter (KF) state estimation.
- Nonlinear MPC (NMPC)using Extended Kalman Filter (EKF) state estimation.
- MPC-RNN using EKF state estimation.

The first control algorithm is based on a discrete-time linearized dynamical model of the system using Taylor series expansion about a valid operating point $(\hat{x}(k), \hat{u}(k))$ given by

$$x(k+1) = A_k\hat{x}(k) + B_k\hat{u}(k) \tag{3.18}$$
$$y(k) = C\hat{x}(k)$$

where $x(k) \in \mathbb{R}^n$ is the state vector, $u(k) \in \mathbb{R}^m$ is the input vector, $y(k) \in \mathbb{R}^p$ is the output vector, A_k and C_k matrices are obtained using the equations in the Algorithm 1 for A_k and H_k, respectively. The matrix B_k is obtained from $B_k = \frac{\partial f}{\partial u}|_{x=\hat{x}(k)}$. The matrices A_k, B_k, and C are defined as follows:

$$A_k = \frac{\partial f}{\partial x}|_{(\hat{x}(k),\hat{u}_{lin}(k))} = \begin{bmatrix} 0 & 0 & -\hat{v}\cos(\hat{\psi}) - \hat{u}\sin(\hat{\psi}) & \cos(\hat{\psi}) & -\sin(\hat{\psi}) & 0 \\ 0 & 0 & \hat{u}\cos(\hat{\psi}) - \hat{v}\sin(\hat{\psi}) & \sin(\hat{\psi}) & \cos(\hat{\psi}) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{-d_u}{m_u} & \frac{m_v}{m_u}\hat{r} & \frac{m_v}{m_u}\hat{v} \\ 0 & 0 & 0 & \frac{-m_u}{m_v}\hat{r} & \frac{d_v}{m_v}\hat{r} & \frac{-m_u}{m_u}\hat{u} \\ 0 & 0 & 0 & \frac{m_u - m_v}{m_r}\hat{v} & \frac{m_u - m_v}{m_r}\hat{u} & \frac{-d_r}{m_r} \end{bmatrix}$$

$$B_k = \frac{\partial f}{\partial u}|_{(\hat{x}(k),\hat{u}_{lin}(k))} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m_u} & 0 & 0 \\ 0 & \frac{1}{m_v} & 0 \\ 0 & 0 & \frac{1}{m_r} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

where $\hat{u}(k) = (\hat{u}, \hat{v}, \hat{r})$ and the rest of parameters and equations are defined as follows:

$$\dot{x} = u\cos(\psi) - v\sin(\psi) \tag{3.19a}$$

$$\dot{y} = u\sin(\psi) + v\cos(\psi) \tag{3.19b}$$

$$\dot{\psi} = r \tag{3.19c}$$

$$m_u\dot{u} - m_v vr + d_u u = \tau_u \tag{3.19d}$$

$$m_v\dot{v} - m_u ur + d_v v = \tau_v \tag{3.19e}$$

$$m_r\dot{r} + (m_v - m_u)uv + d_r r = \tau_r \tag{3.19f}$$

where $m_u = m - X_{\dot{u}}$, $m_v = m - Y_{\dot{v}}$, $m_r = I_z - N_{\dot{r}}$, $d_u = -X_u - X_{|u|u}|u|$, $d_v = -Y_v - Y_{|v|v}|v|$ and $d_r = -N_r - N_{|r|r}|r|$.

The MPC implementation can be formulated by introducing an open-loop optimization problem at every time interval k [116]. In the linear MPC and in case of the unconstrained optimization, equation (2.34) is employed to solve the Riccati equation yielding:

$$\Delta u = ({M_d}^T Q M_d + R)^{-1} {M_d}^T Q(y_r - y_{free}) \tag{3.20}$$

where y_{free} is the free response generated by the KF and M_d^T is the transposed of the dynamic matrix M_d specified in equation (2.42). The step-response coefficients of the linearized model comprising the dynamic matrix M_d given by:

$$M_d(k) = \begin{bmatrix} B_k & 0 & \dots & 0 \\ A_k B_k & B_k & \dots & 0 \\ A_k^2 B_k & A_k B_k & B_k & \vdots \\ A_k^{N-1} B_k & A_k^{N-2} B_k & \dots & A_k^{N-N_u} B_k \end{bmatrix} \tag{3.21}$$

The second control method, NMPC, directly minimizes equation (2.34) at each instant. Therefore, it seems to be the most accurate control scheme. However, performing an

online nonlinear optimization of a non-convex problem is both time-consuming and unreliable. In this thesis, the Levenberg-Marquardt algorithm was chosen among the nonlinear least squared methods to optimize equation (2.34). A discrete EKF Algorithm 1 is also required to estimate the current states and predict the output over the prediction horizon. In the Algorithm 1, R_c is a diagonal matrix with the entries computed from measurement covariance. In other words, R_c is our estimation of the measured noise's power.

The third control algorithm which was used in this thesis, is the nonlinear MPC-RNN with the EKF state estimation. This algorithm is based on the online strategy of the nonlinear state space model specified in equation (2.26a), using EKF Algorithm 1. After discretization the model (A, B, C), the dynamic matrix M_d can be computed using equation (3.21). During each sampling interval, the RNN is applied for solving the formulated quadratic optimization problem. Within a sampling interval k, the convergence behaviors of the one-layer RNN is depicted in Figure 3.2. The output of RNN is $\Delta u(k)$ vector where the first element is feed into the control system. It is shown that the RNN can converge to the optimal solution in a very short time interval.

3.4 Simulation Results

To analyze and evaluate the effectiveness of the performance of the developed method a comparative simulation is conducted. During the following experiments, the AUV is forced to do a desired path tracking mission as well as considering the constraints on the variables. The performance runtime, control cost, position and orientation state errors regarding each control algorithm are compared.

Physical constraints of the AUV dynamic model is summarized in Table 3.1. These constraints are only considered in MPC-RNN and NMPC algorithms. In all three designed control cases, the results are obtained using $T = 20s$, $T_s = 0.2s$, the output penalty matrix

$Q = 10^4 \times I_{6\times6}$ and control penalty matrix $R = I_{3\times3}$. Also, the maximum thruster force along X and Y axes is assumed to be $400N$ and the maximum thruster torque of yaw axis is set to be $100Nm$. The inertia matrix M, matrix of Coriolis and centripetal terms $C(\nu)$, and the damping matrix $D(\nu)$ in equation (2.3) are as follows:

$$M = \begin{bmatrix} 25.8 & 0 & 0 \\ 0 & 33.8 & 1.0115 \\ 0 & 1.0115 & 2.76 \end{bmatrix},$$

$$C(\nu) = \begin{bmatrix} 0 & 0 & -(33.8v + 1.0115r) \\ 0 & 0 & 25.8u \\ 33.8v + 1.0115r & -25.8u & 0 \end{bmatrix}$$

$$D(\nu) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 7 & 0.1 \\ 0 & 0.1 & 0.5 \end{bmatrix}$$

The control objective is to force the AUV to follow a predefined trajectory of positions and orientations. The simplified dynamic model consists of six state variables and three input variables. AUV system states are the positions on X and Y axis and its orientation from Z axis. Considering the equation (2.1a), the initial position, orientation and velocity of the AUV are set to $\eta_0 = [0, 0, \frac{\pi}{15}, 0, 0, 0]^T$ and the desired tracking mission's surge, sway and yaw yields $\eta_d = [2.4, 2, \frac{\pi}{4}, 0, 0, 0]^T$.

The effect of environmental disturbances due to irrational ocean currents are gained from equation (2.19) and described as slowly varying drift forces acting on the input channels of AUV along X and Y axses.

In the EKF/KF initializations, the measurement covariance matrix $R_c = diag[5 \times 10^{-2}, 5 \times 10^{-2}, 5 \times 10^{-2}, 1 \times 10^{-2}, 1 \times 10^{-2}]^2$ and the system covariance matrix $Q_c = diag[5 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-5}, 1 \times 10^{-5}, 1 \times 10^{-5}]^2$. For the vehicle, the matrix of unknown nonlinear uncertainty from 2.3 is considered as diagonal matrix of square roots of matrix element from the measurement covariance matrix multiplied by $rand(1, p)$, where p is the number of system output states.

$$g(\eta) = diag[sqrt(R_c) \times rand(1, p)]$$

For the RNN initialization, $\lambda_{RNN} = 100$, $\Delta u_0 = 0 \times ones(N, 1)$, and $\mu_0 = 0$.

The simulations are conducted in MATLAB. The simulation results is an average of 20 simulations held on a computer equipped with a quad-cored processor operating at 2.67 GHz, and is operated by a 64-bit operating system.

Table 3.1: Physical constraints of the AUV dynamic model.

Discription	Variables	Min	Max
Roll Angular Velocity	p	$-\pi/6$	$\pi/6$
Pitch Angular Velocity	q	$-\pi/6$	$\pi/6$
Yaw Angular Velocity	r	$-\pi/6$	$\pi/6$
Roll Euler Angle	ϕ	$-\pi$	π
Pitch Euler Angle	θ	$-\pi/2$	$\pi/2$
Yaw Euler Angles	ψ	-2π	2π

3.4.1 First Experiment $N = 9$, $Nu = 1$

In the first experiment, the AUV is forced to follow a predefined trajectory of positions and orientations. The initial position, orientation and velocity of the AUV set to $\eta_0 = [0, 0, \frac{\pi}{15}, 0, 0, 0]^T$ and the desired tracking mission's surge, sway and yaw results in $\eta_d = [2.4, 2, \frac{\pi}{4}, 0, 0, 0]^T$. Prediction horizon assumed to be $N = 9$, the control horizon is $N_u = 1$ and number of neurons in the RNN architecture is 27.

During each sampling interval $T_s = 0.2s$, the RNN is applied to solve the formulated quadratic optimization problem. The convergence behaviors of the one-layer RNN is depicted in Figure 3.2. It is shown that the output of RNN ($\Delta u(k)$) converges to the optimal solution within the time interval.

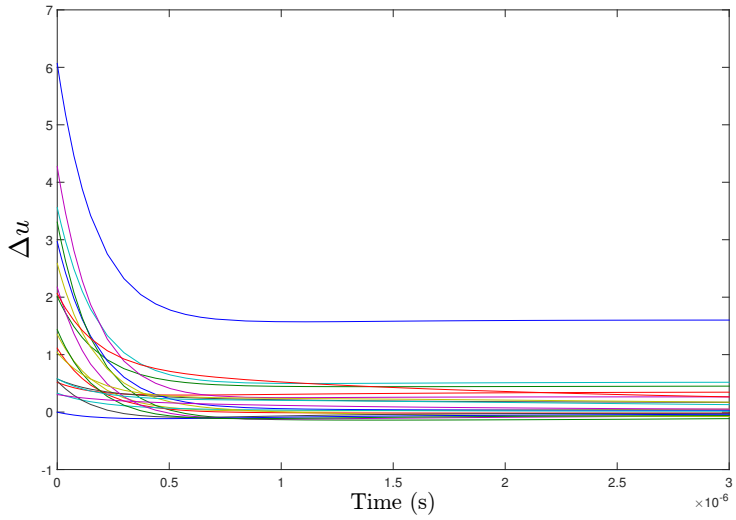
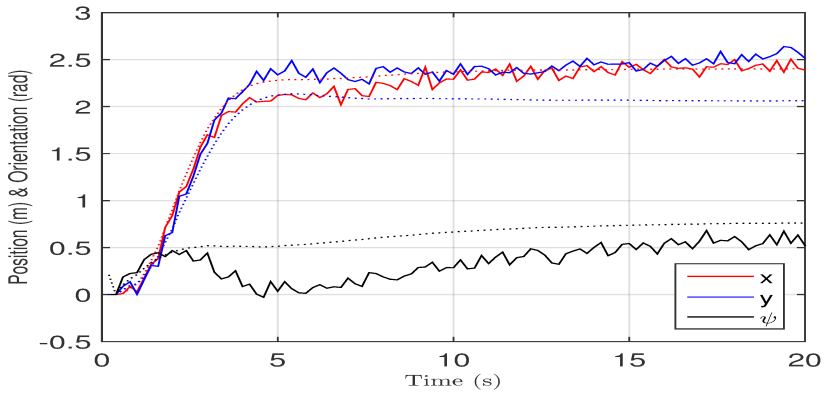


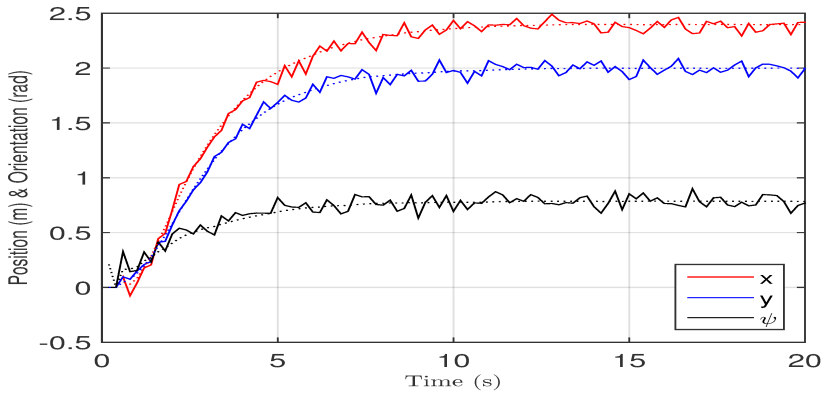
Figure 3.2: Convergence behaviors of the one-layer RNN.

The AUV reaches the pre-defined set-point via three given control schemes. The position and orientation (Euler angle) states are shown in Figure 3.3. More details on this comparative case is shown in the next 3 figures.

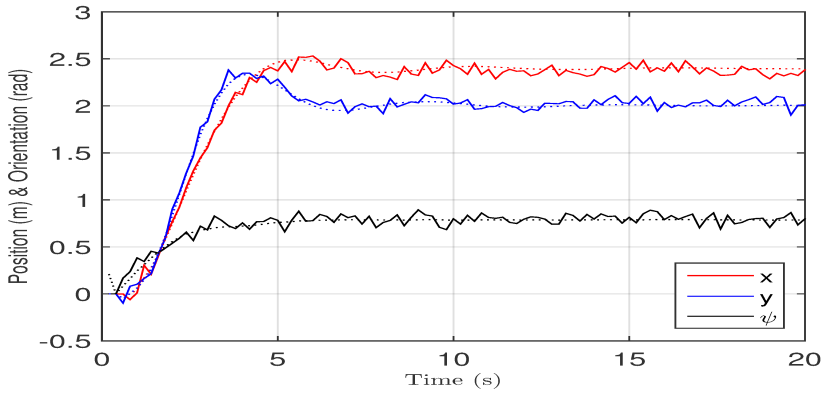
The position states along X and Y axis is shown in Figures 3.4 and 3.5. As can be seen, the developed method reaches the pre-defined set-point along X and Y axes faster



(a) Positions and orientation states of AUV along X and Y axes and around Z axis for linear MPC scheme using KF in the first experiment.



(b) Positions and orientation states of AUV along X and Y axes and around Z axis for nonlinear MPC scheme using EKF in the first experiment.



(c) Positions and orientation states of AUV along X and Y axes and around Z axis for our developed MPC-RNN scheme using EKF in the first experiment.

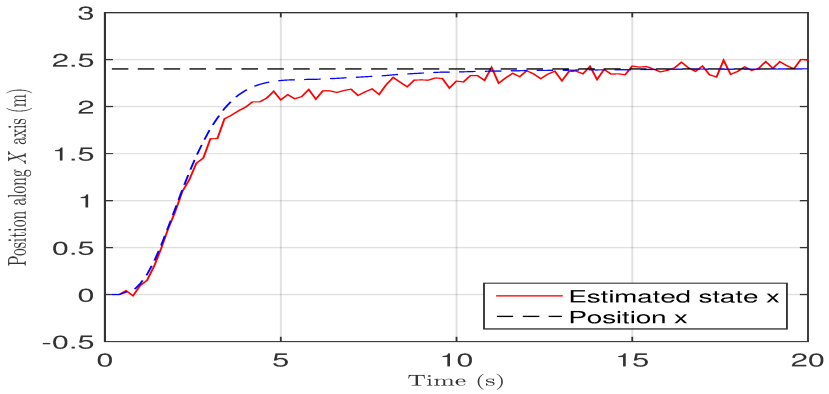
Figure 3.3: Comparison of the position and orientation (Euler angle) states in three control schemes. The measured states and the states values are shown for each position and orientation state. The desired tracking mission's surge, sway and yaw are $\eta_d = [2.4, 2, \frac{\pi}{4}, 0, 0, 0]^T$.

than the other two control methods. Moreover, using Kalman Filter in linear MPC, more differences between the measured and the value of state is shown in comparison with using EKF in the nonlinear MPC control method.

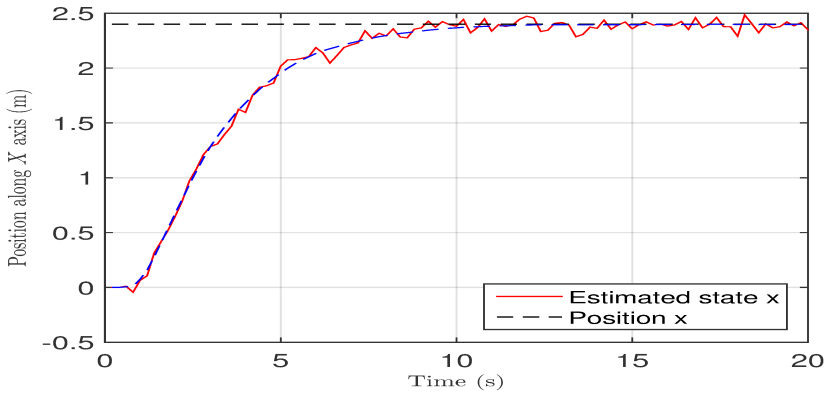
The orientation state around Z axis is shown in Figure 3.6. As can be seen, the developed method and the NMPC method is reached the pre-defined set-point around Z axis faster than linear MPC methods. Also, using Kalman filter in linear MPC, more differences between the measured and the value of state is shown in comparison with using EKF in nonlinear MPC control methods.

Control inputs of all three approaches are illustrated in Figure 3.7. In all three control methods, the optimal control inputs are found. It can be seen that our developed method has more oscillation than the other two control methods. However, our developed control method has the advantages of being more accurate (in comparison with linear MPC) as well as faster performance runtime (in comparison with NMPC method) during its path tracking mission.

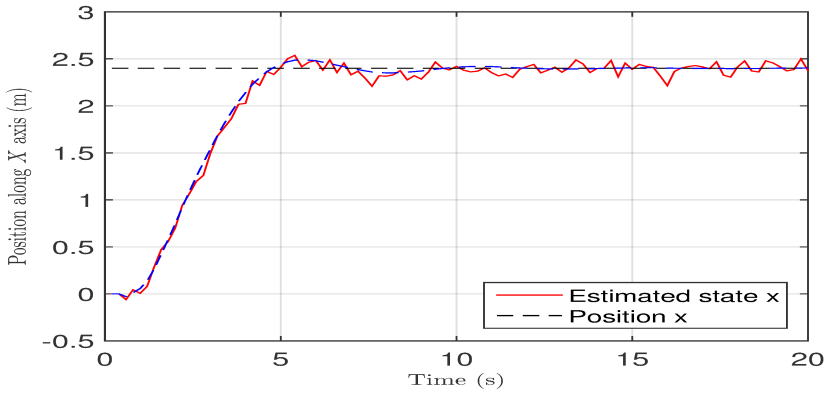
The linear and angular velocity states for AUV in all three control schemes are shown in Figure 3.8. Note that in this path following mission, the goal is to just control the position and orientation states. Linear and rotational velocity states are given to compare the overshoots. In Figure 3.8, it is shown that the NMPC method has better results in case of velocity states than the other two control schemes. Moreover, linear MPC has more oscillatory results compared to the other two control schemes.



(a) Position state along X axis for linear MPC scheme using KF in the first experiment.

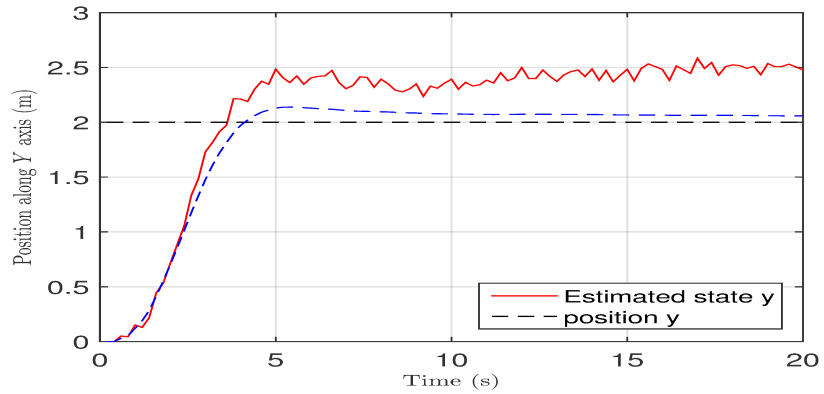


(b) Position state along X axis for nonlinear MPC scheme using EKF in the first experiment.

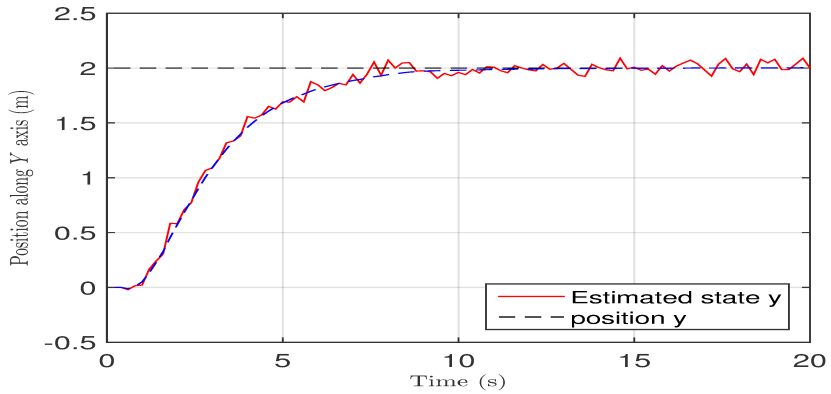


(c) Position state along X axis for our developed MPC-RNN scheme using EKF in the first experiment.

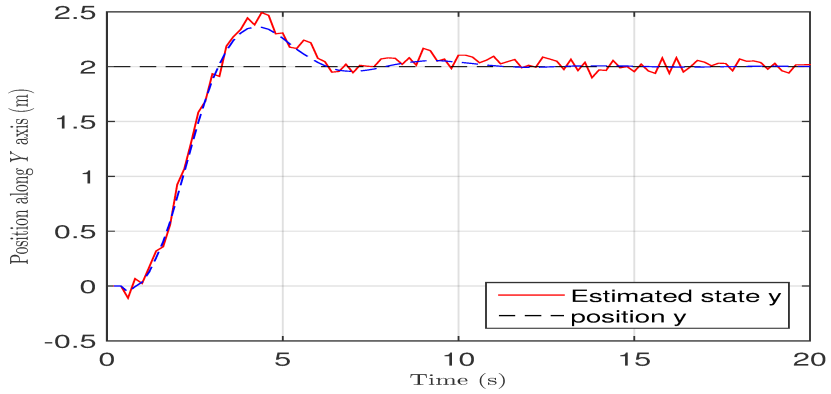
Figure 3.4: Comparison of the position state along X axis in three control schemes. The measured states and the states values are shown for position state along X axis. The desired tracking mission's surge, sway and yaw are $\eta_d = [2.4, 2, \frac{\pi}{4}, 0, 0, 0]^T$.



(a) Position state along Y axis for linear MPC scheme using KF in the first experiment.

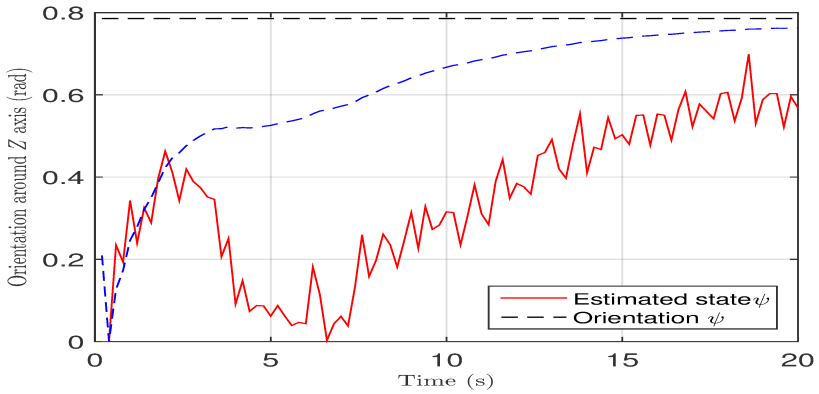


(b) Position state along Y axis for nonlinear MPC scheme using EKF in the first experiment.

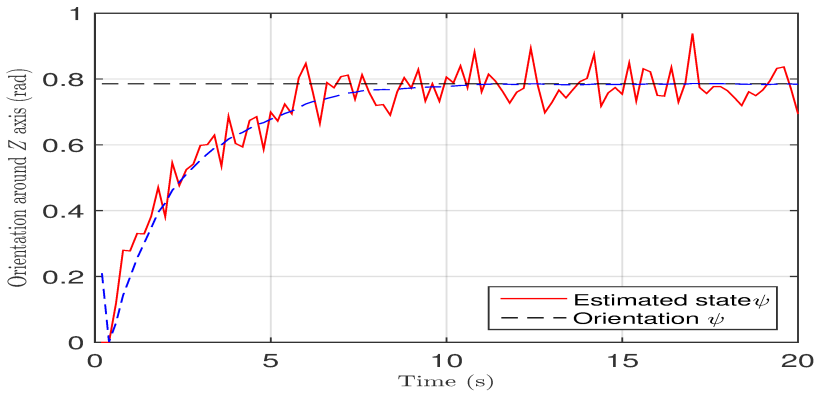


(c) Position state along Y axis for our developed MPC-RNN scheme using EKF in the first experiment.

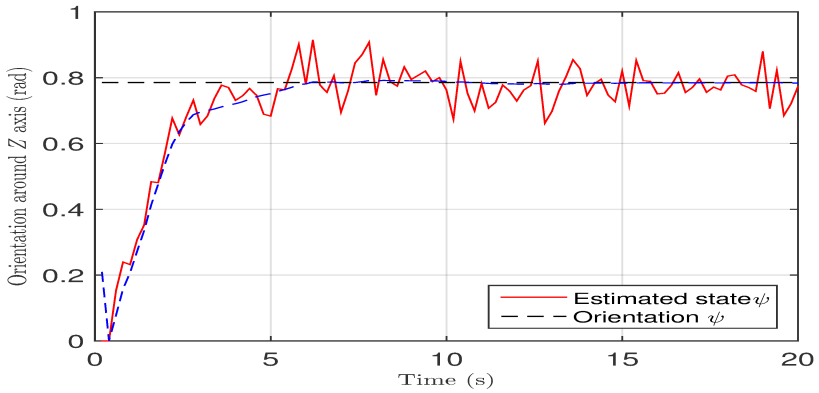
Figure 3.5: Comparison of the position state along Y axis in three control schemes. The measured states and the states values are shown for position state along Y axis. The desired tracking mission's surge, sway and yaw are $\eta_d = [2.4, 2, \frac{\pi}{4}, 0, 0, 0]^T$



(a) Orientation state around Z axis for linear MPC scheme using KF in the first experiment.

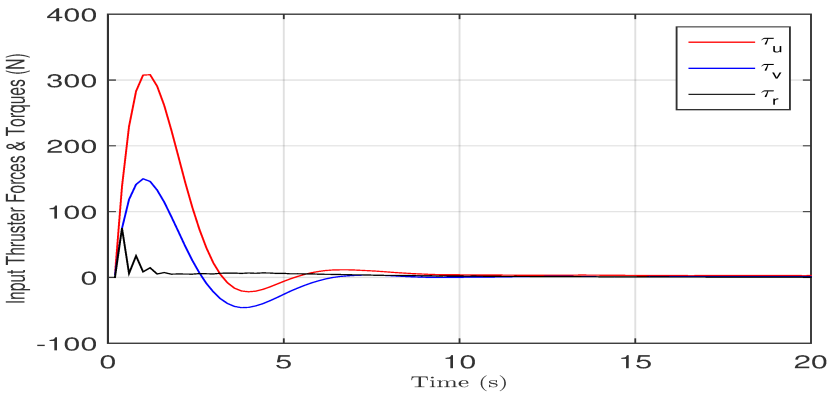


(b) Orientation state around Z axis for nonlinear MPC scheme using EKF in the first experiment.

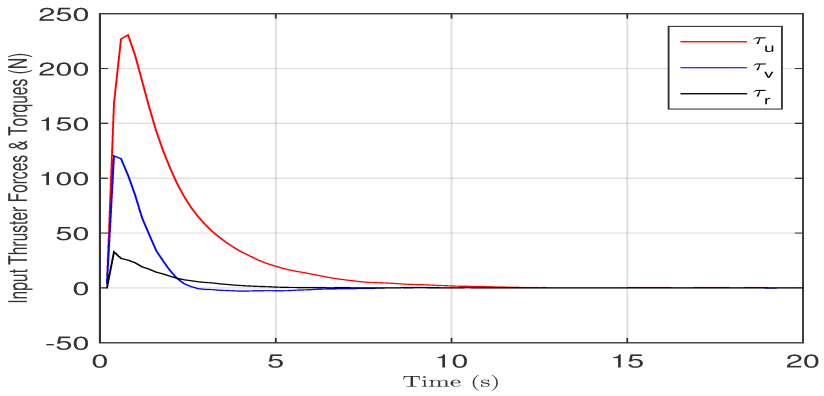


(c) orientation state around Z axis for our developed MPC-RNN scheme using EKF in the first experiment.

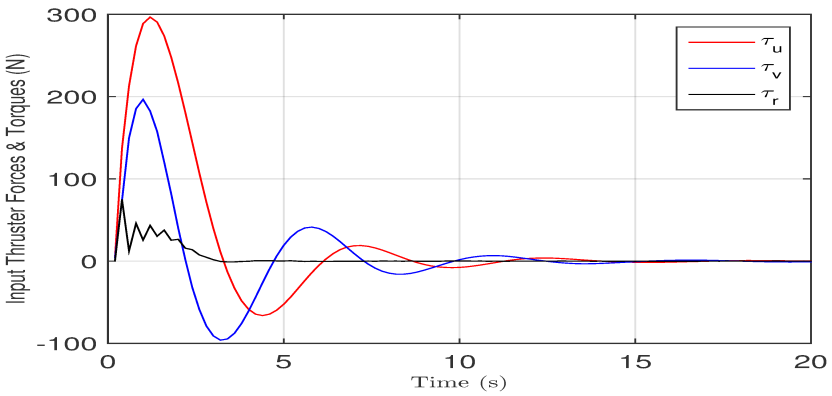
Figure 3.6: Comparison of the orientation state around Z axis in three control schemes. The measured states and the states values are shown for orientation state around Z axis. The desired tracking mission's surge, sway and yaw are $\eta_d = [2.4, 2, \frac{\pi}{4}, 0, 0, 0]^T$



(a) Control efforts of AUV for linear MPC scheme using KF in the first experiment.

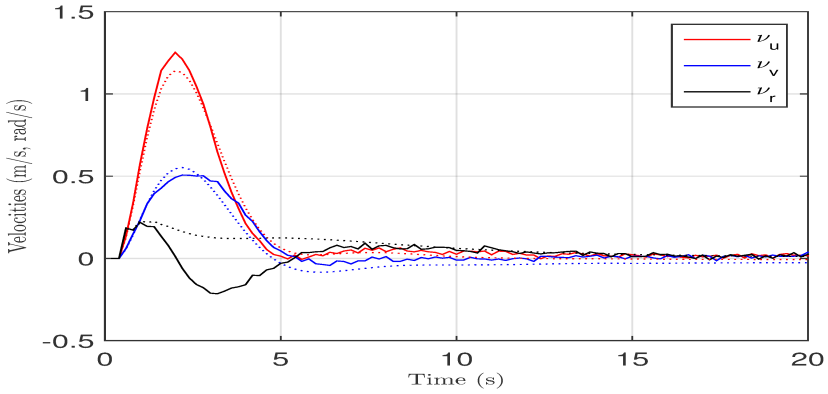


(b) Control efforts of AUV for nonlinear MPC scheme using EKF in the first experiment.

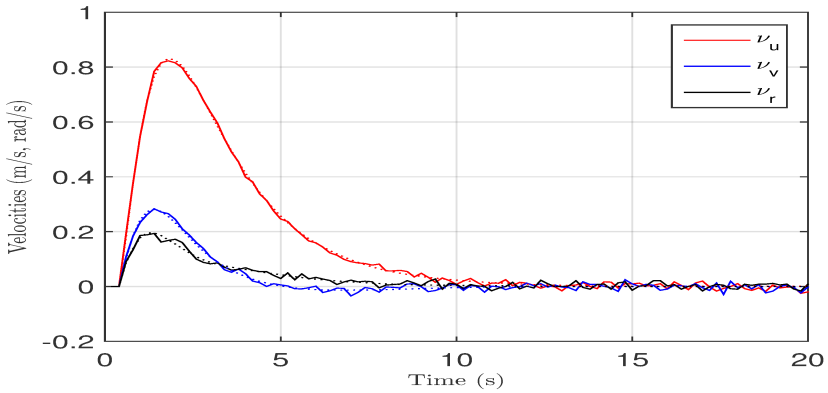


(c) Control efforts of AUV for our developed MPC-RNN scheme using EKF in the first experiment.

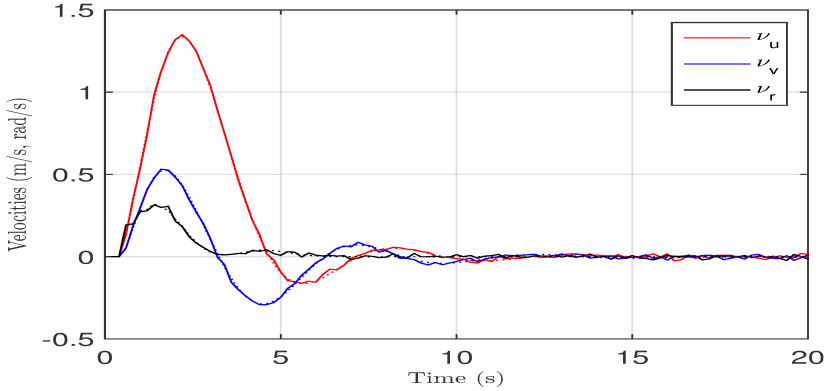
Figure 3.7: Comparison of control effort results in three control schemes.



(a) Linear and angular velocity states of AUV for linear MPC scheme using KF in the first experiment.



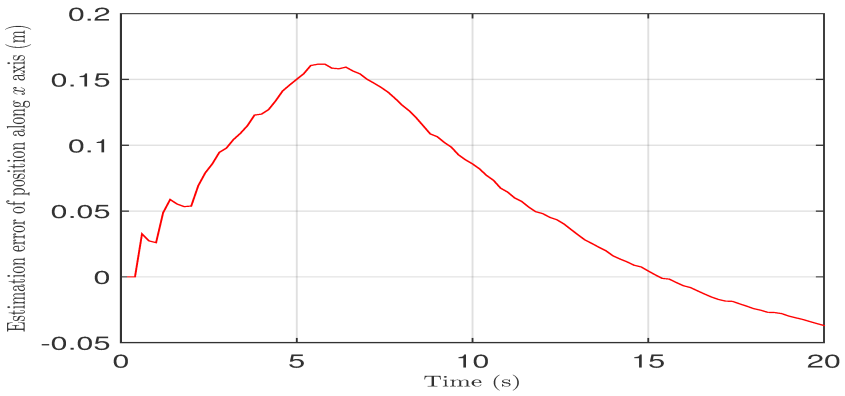
(b) Linear and angular velocity states of AUV for nonlinear MPC scheme using EKF in the first experiment.



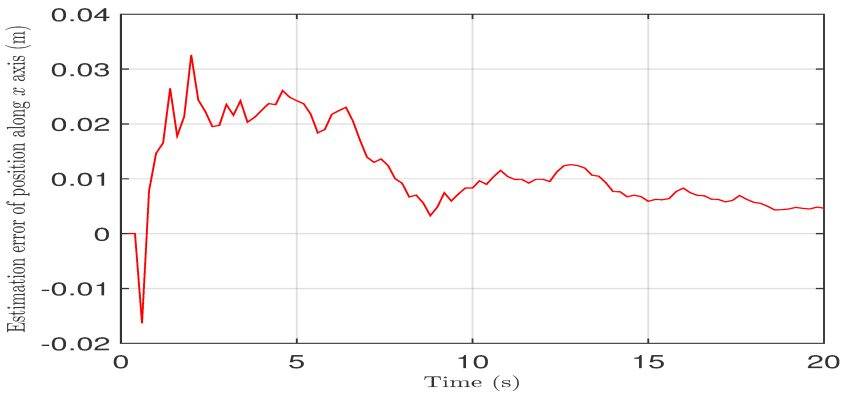
(c) Linear and angular velocity states of AUV for the MPC-RNN scheme using EKF in the first experiment.

Figure 3.8: Comparison of the linear and angular velocity states in three control schemes.

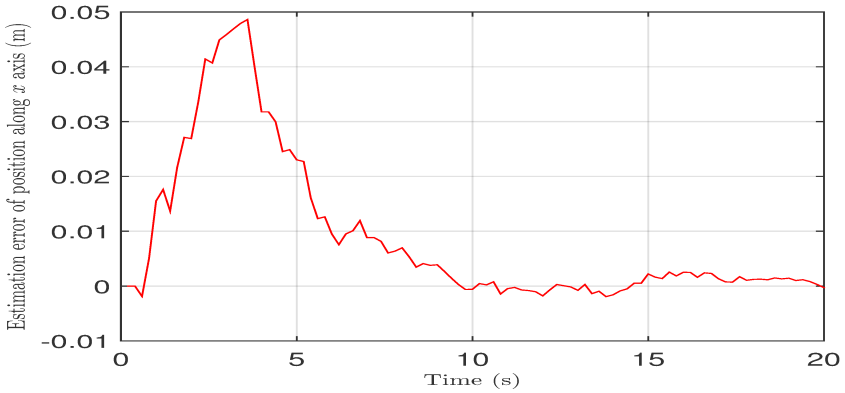
Estimation of Error Signals in the First Experiment: The estimation of error signals in the first experiment with same conditions are illustrated for all three control schemes. During the trajectory tracking and path following, the prediction horizon and control horizon are assumed to be $N = 9$ and $N_u = 1$, respectively. Figures 3.9 to 3.11 illustrate the comparison of estimation error signals in surge, sway and yaw motions among three control schemes. Linear MPC using KF has the highest estimation errors while our developed control methodology performs close to the NMPC approach. Both NMPC and MPC-RNN control methods, use the EKF algorithm.



(a) Estimation error signals of position tracking along X axis for linear MPC scheme using KF in the first experiment.

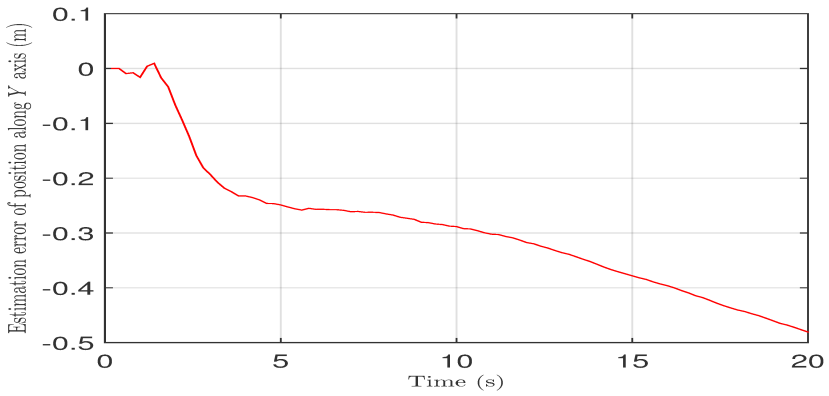


(b) Estimation error signals of position tracking along X axis for nonlinear MPC scheme using EKF in the first experiment.

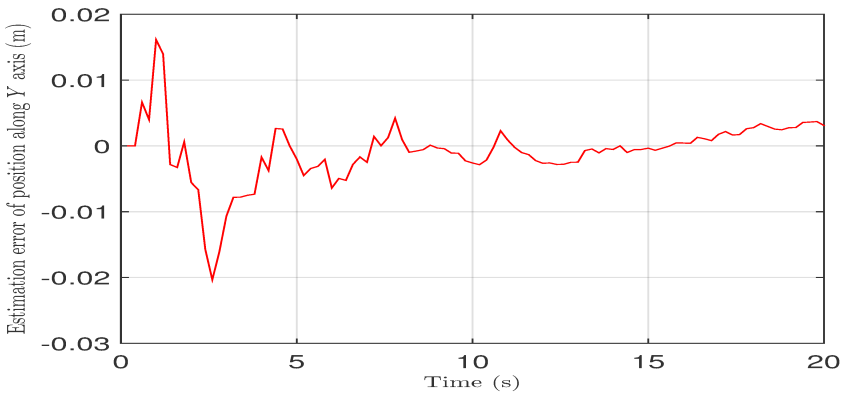


(c) Estimation error signals of position tracking along X axis for our developed MPCL-RNN scheme using EKF in the first experiment.

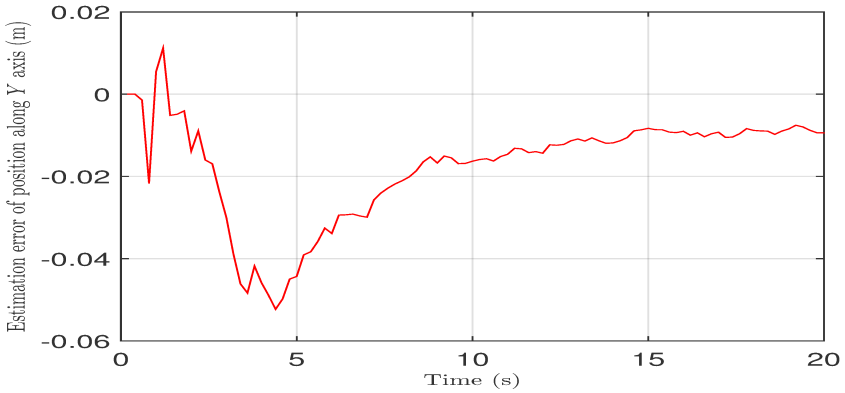
Figure 3.9: Comparison of the estimation error signals of position tracking along X axis (surge).



(a) Estimation error signals of position tracking along Y axis for linear MPC scheme using KF in the first experiment.

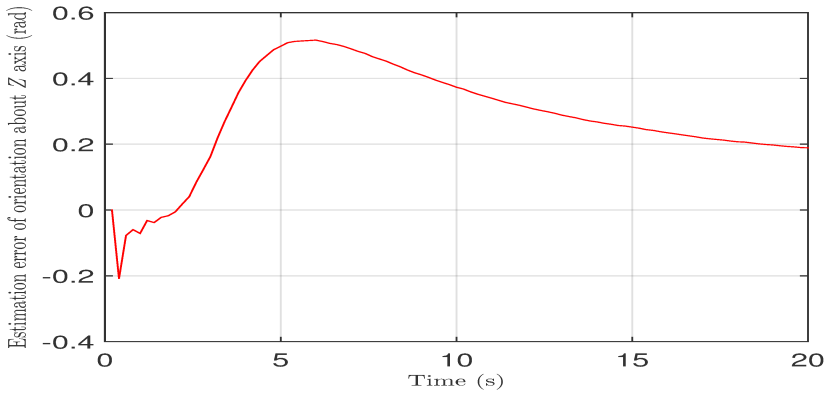


(b) Estimation error signals of position tracking along Y axis for nonlinear MPC scheme using EKF in the first experiment.

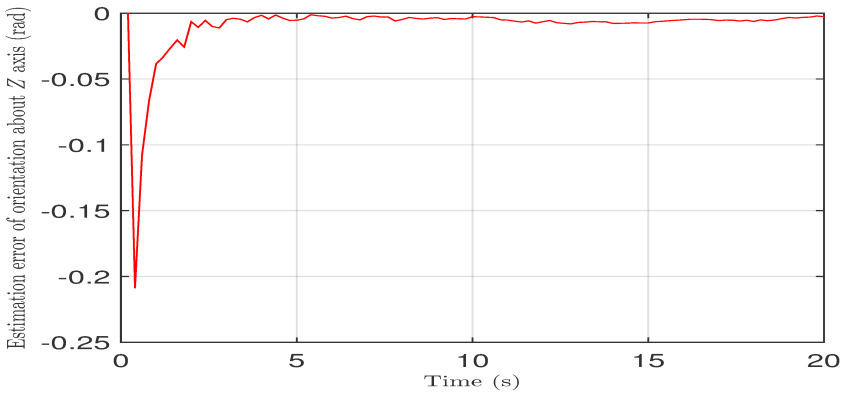


(c) Estimation error signals of position tracking along Y axis for our developed MPC-RNN scheme using EKF in the first experiment.

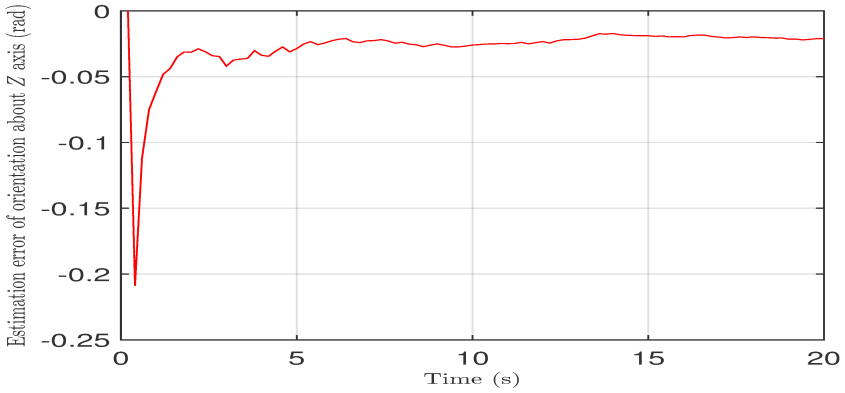
Figure 3.10: Comparison of the estimation error signals of position tracking along Y axis (sway).



(a) Estimation error signals of orientation tracking around Z axis for linear MPC scheme using KF in the first experiment.



(b) Estimation error signals of orientation tracking around Z axis for nonlinear MPC scheme using EKF in the first experiment.



(c) Estimation error signals of orientation tracking around Z axis for our developed MPC-RNN scheme using EKF in the first experiment.

Figure 3.11: Comparison of the estimation error signals of orientation tracking around Z axis (yaw).

Mean Square Error (MSE), Steady-state Tracking Error, Tracking Error Cost, Performance Runtime, and Average Control Cost:

Tables 3.2 to 3.4, and 3.6 illustrate the comparison results for control schemes in Section 3.3 during the trajectory tracking and path following problem stated in Section 3.4.1. The prediction horizon and control horizon are assumed to be $N = 9$ and $N_u = 1$, respectively.

Table 3.2: MSE for 100 sampling instants of each system state, using linear MPC with KF, nonlinear MPC with EKF and MPC-RNN with EKF.

System States	Linear MPC with KF	NMPC with EKF	MPC-RNN with EKF
Position along X axis	0.0074	4.9461×10^{-4}	3.5321×10^{-4}
Position along Y axis	0.0987	3.0781×10^{-4}	2.2462×10^{-4}
Orientation around Z axis	0.1049	0.0011	9.2110×10^{-4}

Table 3.2 shows that, linear MPC using KF control method has the highest values among three control schemes in case of MSE. On the other hand the proposed control scheme has lower values of MSE in case of positions and orientation states in comparison with NMPC using EKF.

Table 3.3: Steady-state Tracking Error of each system state, using linear MPC with KF, nonlinear MPC with EKF and MPC-RNN with EKF.

System States	Linear MPC with KF	NMPC with EKF	MPC-RNN with EKF
Position along X axis	0.0015	0.0005	0.0007
Position along Y axis	0.0580	0.0035	0.0003
Orientation around Z axis	0.0253	0.0008	0.0013

Table 3.3 presents that our proposed control method has a better performance than the linear MPC using KF in case of the steady-state tracking error. Overall, among all three methods, NMPC using EKF performed better than the other two control methods.

As can be seen in Table 3.3, our proposed control methodology performs very close to the NMPC using EKF control scheme in case of evaluating the steady-state tracking error. The steady-state error formulation is given by

$$e_{ss} = ||x_i(100) - x^{est}{}_i(100)||$$

where x is the system state.

Table 3.4: Tracking error cost for 100 sampling instants of each system state using linear MPC with KF, nonlinear MPC with EKF and MPC-RNN with EKF control schemes.

System States	Linear MPC with KF	NMPC with EKF	MPC-RNN with EKF
Position along X axis	0.5081	0.8395	0.5602
Position along Y axis	0.3666	0.4269	0.3524
Orientation around Z axis	0.0536	0.0406	0.0293

Table 3.4 presents that our developed control method has lower values of tracking error cost in comparison with the other two control schemes in tracking the desired position state along Y axis and orientation around Z axis. The tracking error cost formulation is given by

$$J_t = \int_0^{100} ||x_i(t) - x^{est}{}_i(t)||^2 dt$$

where x is the system state. In case of tracking error cost values of the positions state along X axis, the MPC-RNN using EKF scheme performs very close to linear MPC using KF control scheme. Table 3.4 also shows that the NMPC using EKF control scheme has higher tracking error cost among all in case of position state along X and orientation state around Z axes.

Table 3.5: A comparison between the performance runtime of each control system.

Control method	Performance runtime
Linear MPC with KF	$0.0002s$
NMPC with EKF	$0.0912s$
MPC-RNN with EKF	$0.0079s$

Table 3.5 shows that our proposed control method has improved the performance runtime of the AUV control system in missions when using nonlinear method and nonlinear optimization. Since linear MPC uses the linearized model and finds the optimal control input linearly, it has a lower performance runtime in comparison with the other two control methods. In our application of a nonlinear system with uncertainties, the linear MPC does not meet the objectives of problem statement in the trajectory following and path tracking mission.

Table 3.6: Average Control Cost of three designed control schemes for 100 sampling instants.

Control scheme	Linear MPC with KF	NMPC with EKF	MPC-RNN with EKF
Control cost of X	4.0088×10^2	3.4096×10^2	3.7056×10^2
Control cost of Y	1.2003×10^2	1.5669×10^2	2.4282×10^2
Control cost of N	1.2176×10^2	1.5291×10	1.0910×10^2
Total Control cost	6.4267×10^2	5.1294×10^2	7.2248×10^2

Table 3.6 presents the costs resulted from the parameters X, Y and N from Table 2.1 as well as the total cost of the designed trajectory tracking and path following problem stated in Section 3.4.1 for each control scheme from Section 3.3. Average control cost formulation is given by

$$J_c = \frac{1}{100} \int_0^{100} ||\Delta u_i(t)||^2 dt$$

where Δu is the system state. As can be seen in Table 3.6, the proposed scheme has higher average control cost among the three control schemes. It is worth mentioning that the NMPC method control cost in handling the path tracking goal around Z axis (the control cost regarding the yaw motion) is significantly lower than other motions in other control methods in Table 3.6. Table 3.6 also shows that the proposed control method has a better performance runtime (in comparison with NMPC method) and more accurate path tracking results (in comparison with linear MPC method), but with an increase in the control cost for the system.

3.4.2 Second Experiment

In the second experiment, the effect of changes in prediction horizon and the control horizon on the performance runtime of each control scheme given in Section 3.3 are investigated. The AUV is required to do the path tracking with its initial position, orientation and velocity set to $\eta_0 = [0, 0, \frac{\pi}{15}, 0, 0, 0]^T$ in which the desired tracking mission's surge, sway and yaw is $\eta_d = [2.4, 2, \frac{\pi}{4}, 0, 0, 0]^T$. Other constraints and specifications are found in Section 3.4.

The main goal of the second experiment is that when designing an MPC, a main concern is how large the prediction horizon N should be. If the pre-defined control plan goal is not achieved then an estimation of the remaining cost-to-go is essential and should be considered. In practice, variations in the parameter N have heavy costs. On the other hand, short control plans are not necessarily guaranteed to achieve all the plant goals [194–196]. Table 3.7 presents the results of second experiment The simulations are conducted in MATLAB. The simulation results is an average of 20 simulations held on a computer equipped with a quad-cored processor operating at 2.67 GHz, and is operated by a 64-bit operating system. This is worth mentioning that, linear MPC using KF control scheme faced with highly unstable circumstances when an increase happen in prediction and control horizons.

Table 3.7: The effect of changing N and N_u on the performance runtime of Nonlinear MPC and MPC-RNN control schemes using EKF for the total sampling times, $20s$.

Measured performance runtime in trajectory tracking and path following control					
N	Control scheme	$N = 8$	$N = 9$	$N = 10$	$N = 11$
$N_u = 1$	MPC-RNN	$0.0078s$	$0.0079s$	$0.0087s$	$0.0097s$
	NMPC	$0.0649s$	$0.0912s$	$0.1170s$	$0.1301s$
$N_u = 3$	MPC-RNN	$0.0082s$	$0.0084s$	$0.0090s$	$0.0110s$
	NMPC	$0.3168s$	$0.3550s$	$0.5432s$	$0.9932s$
$N_u = 5$	MPC-RNN	$0.0103s$	$0.0111s$	$0.0113s$	$0.0125s$
	NMPC	$2.0691s$	$2.1964s$	$2.2005s$	$2.3705s$

As can be seen in Table 3.7, the proposed control method has faster performance runtime. Since the AUV application is a real-time system which needs fast responses to the system inputs, lower performance runtime, while reaching the mission goals, makes it desirable to choose MPC-RNN. Nonlinear optimization of AUV it shown to be time-consuming according to the second experiment in Section 3.4.2. High time-consumption can result in an undesirable transient or unstable situations in the presence of fault. This issue is discussed more in the Chapter 4. Table 3.7 also shows that an increase in the number of the prediction horizon N of the optimization problem from 9 to 10 (with $N_u = 1$) results in an increase of 28% and 10% performance runtime in the NMPC and our developed MPC-RNN control schemes, respectively. In this case neurons in RNN are increased from 27 to 29 in this case. Also, an increase in the number of the prediction horizon N of the optimization problem from 9 to 11 (with $N_u = 1$) results in an increase up to 42% and 22% performance runtime in the NMPC and our developed MPC-RNN control schemes, respectively. In this case, neurons in RNN are increased from 27 to 31.

Increasing the number of control horizon N_u from 1 to 3 (with $N = 9$) results in an

increase of 289% and 6% performance runtime in NMPC and MPC-RNN control schems, respectively. In this case, neurons in RNN are increased from 27 to 45. Also, an increase in the number of control horizon N_u from 1 to 5 (with $N = 9$) results in increasing 2300% and 40% performance runtime in NMPC and MPC-RNN control schems, respectively. In this case, neurons in RNN are increased from 27 to 63.

Similar experiments in Table 3.7 show that increasing the number of the prediction horizon N and the control horizon N_u of the optimization problem has mild effects on the system performance runtime when the AUV is controlled with the proposed control method in comparison with the NMPC method. The reason behind this added tune-ability feature in the system comes from the adaptability of RNN in solving the solution of finding the optimal control input to the system.

3.5 Discussion

In this chapter a hybrid MPC and RNN control method is developed and integrated with EKF. The proposed control scheme in this chapter allows for designing a control solution that takes into account the system kinematics and meets uniform asymptotic convergence requirements using RNN. Comparative methods are destined and several simulations are given in order to evaluate the developed control method of this chapter. Based on the graphs and tables in each experiment the following points can be stated:

- The suboptimal algorithm in equation (3.4a) performs an online execution of the QP problem. It is shown that the solution is found within a bounded time frame (details in Section 2.8.1 and 3.2) with the developed control method of this chapter, where the nonlinear optimization may result in a local minimum.

- The proposed method of this chapter reached to the pre-defined path tracking goal faster than NMPC and Linear MPC methods according to Figures 3.4, 3.5, and 3.6. Also, the developed MPC-RNN method performed with least values of MSE for positions and orientation states among three designed control methods according to Table 3.2. Moreover, MPC-RNN developed method shown to have lower Tracking Error Cost values regarding the position and orientation states, overall, while NMPC method showed the highest values among 3 designed control methods.

- According to Figures 3.4, 3.5, and 3.6, Linear MPC using KF has the highest estimation errors and the developed control method of this chapter performs close to NMPC control method in this regards. Both NMPC and MPC-RNN control methods, benefited from EKF algorithm. Also, according to the results from Table 3.3, the proposed control method of this chapter performs very close to NMPC using EKF control scheme, in case of evaluating the steady-state tracking error values. Both

NMPC and MPC-RNN performed better than Linear MPC method, if steady-state error is considered.

- The developed method of this chapter has more oscillatory behavior in its control input results than the other 2 control methods, according to Figure 3.7. Also, Table 3.6 presents that the improvement in the performance runtime (in comparison with NMPC method) as well as the improvement in accurate trajectory following and path tracking (in comparison with linear MPC method) of the proposed control method of this chapter, comes along with an increase in the control cost for the system.

- The reduction in performance runtime complexity gained from our suboptimal algorithm in comparison with the NMPC is very significant when it comes to setting various prediction and control horizon numbers. This is illustrated in the Table 3.7.

- The number of the control horizon N_u of the optimization problem has larger effect on the performance runtime than the number of prediction horizon N, specially when NMPC control method is used, as per Table 3.7. The proposed control method in this chapter, overall has performed better when faced with MPC design tunning circumstances. According to Section 3.4.2, since MPC-RNN performance runtime would not be affected by tunning variables N and N_u significantly, this makes it a feasible option to be applied on industrial applications.

Therefore , considering our real-time application and its objective that is trajectory following and path tracking in horizontal plan, the choice of the developed method of this chapter to control is feasible.

3.6 Conclusion

In this chapter, a nonlinear MPC algorithm is formulated to ingrated with a Recurrent Neural Network to solve the QP problem, resulted from MPC cost function minimization. The developed MPC-RNN control scheme applied to force a nonlinear AUV model with uncertainties to follow a predefined trajectory of positions and orientations. Also, comparative methods are developed and series of simulations and analysis results are provided. Finally, the discussion about the results is presented. Next chapter equips our proposed control method of this chapter with a Dual Extended Kalman Filter in order to develop an active fault tolerant control system.

Chapter 4

Active Fault-Tolerant Nonlinear Predictive Control Using Recurrent Neural Networks

4.1 Introduction

In the previous chapter, the MPC method integrated with RNN was employed to design the controller for an AUV nonlinear model with uncertainties during a trajectory following and path tracking mission. In Chapter 3, EKF provided the AUV system with an efficient method for generating approximate maximum-likelihood estimate of the states of a discrete-time nonlinear dynamic system. EKF optimally combines noisy observations with predictions from the pre-defined dynamic model via a recursive procedure. Also, EKF can estimate the parameters of a model (e.g. neural network) given clear training data of input and output data. This way, EKF presents a modified-Newton type of algorithm for an online system identification [127].

The developed control method of the previous chapter has benefited from MPC properties that are using the system model, a prediction horizon and explicitly handling constraints. Also, The developed control method of the previous chapter allows for designing a control solution that takes into account the system dynamics and meets uniform asymptotic convergence requirements using RNN. Moreover, MPC-based control algorithms need well tuned parameters that the proposed method in Chapter 3 has addressed, when it comes to choosing prediction and control horizon numbers. MPC does an optimization problem setup and its development time is much shorter than for many competing advanced control methods and it is easier to maintain. Therefore, changing model or specifications does not require complete redesign of the model predictive controller. Therefore, in this chapter the proposed control method in Section 3.2 is modified with replacing the Extended Kalman Filter (EKF) by a dual-Extended Kalman Filter (dual-EKF) to address the stated problem of this chapter that is controlling a nonlinear system with faulty inputs.

The proposed active system control strategy of this chapter accounts for the online fault estimation as well as recovery, yields an Active Fault-Tolerant Control (AFTC) method. The developed AFTC method of this chapter uses the proposed control method from Chapter 3 to address the problem of trajectory tracking and path following of an AUV faced with Loss of Effectiveness (LOE) in its actuators during its pre-defined mission. The actuator LOE is depicted in the effectiveness coefficient matrix Γ given in equation (2.23), which affects on control input matrix u in the model.

In Section 3.4.2, the second experiment presented the conventional nonlinear MPC (NMPC) using EKF control method as a time consuming and risky optimization method. From fault-tolerant control view, using NMPC when a fault occurs, a non-convex problem may not converge or converge to a local minima, according to the works [87] and [99]. Table 3.7 presented NMPC as a method that its parameters are not easily tunable when AUV is following a predefined trajectory of positions and orientations in its normal status. This

can make Fault-Tolerant Nonlinear Model Predictive Control (FT-NMPC) algorithms get unstable after the occurrence of the fault.

Finally, performance efficiency of the developed AFTC method of this chapter is evaluated during several trajectory tracking and path following mission scenarios. Simulation results of an AUV nonlinear model with uncertainties during designed missions, faced with LOE actuator faults, are given in Section 4.3. The results are analyzed and compared with conventional FT-NMPC method in Section 4.4.

4.2 Active Fault-Tolerant Model Predictive Control (AFTC) Framework

Shen *et al.* [197] presents a unified receding horizon optimization scheme to solve the combined path planning and tracking control problem for an AUV. In [197], the NMPC technique was employed to the AUV and simulation results using a Falcon dynamic model with realistic experimentally identified parameters revealed the effectiveness of the proposed control algorithm. Shen *et al.* [197] claim that it remains to analyze the robustness of the closed-loop system and the disturbance rejection performance of the control system. Also, a fast NMPC algorithm should be developed for AUV real-time implementations [197] .

Motivated by the literature given, the problem of adaptive AFTC for a class of nonlinear systems with actuator fault is investigated in this chapter. In fact, the stated problem of this chapter arises when the input is not accurate and requires coupling of both the state estimation and the parameter estimation. We consider the problem of learning both the hidden states $x(k)$ and parameters w of a discrete-time nonlinear dynamical system in the

following equations,

$$\dot{x} = f(x, u, w) + v,$$
$$y = h(x) + n,$$

where

$$f(x, u, w) = \begin{bmatrix} M\nu^{-1}[w\tau - C(\nu)\nu - D(\nu) - g(\eta)] \\ J(\eta)\nu \end{bmatrix},$$

where f and h are known nonlinear functions, v and n denote the white noise sequences of uncorrelated Gaussian random vectors with zero means and covariance matrices Q_c and R_c. In fact, $w\tau = \tau_f$ is the faulty input. Then, a discrete-time nonlinear system is given as follows:

$$x(k+1) = f(x(k), u(k), w(k)) + v(k), \tag{4.3a}$$
$$y(k) = hx(k) + n(k), \tag{4.3b}$$

The nonlinear system in equations (4.3) is subject to the following constraints:

$$u_{min} \leq u(k) \leq u_{max}, \tag{4.4a}$$
$$\Delta u_{min} \leq \Delta u(k) \leq \Delta u_{max}, \tag{4.4b}$$
$$y_{min} \leq y(k) \leq y_{max}, \tag{4.4c}$$

where the system states $x(k) \in \mathbb{R}^n$ and the set of model parameters $w(k)$ for the dynamical system must be simultaneously estimated from only the observed noisy signal $y(k) \in \mathbb{R}^p$, $f(.)$ and $h(.)$ are the model nonlinear structure, $v(k)$ is the process noise, $n(k)$ is the measurement noise, and $u(k) \in \mathbb{R}^m$ corresponds to the observed exogenous inputs, u_{min}, u_{max}, $\Delta u_{min}, \Delta u_{max}$, and y_{min}, y_{max} are the lower and upper bounds.

As it is mentioned before, the task of dual-EKF is to estimate both the state and model parameters from only noisy observations. Therefore, at every time interval, a state-EKF estimates the states using the current model estimate $\hat{w}(k)$, while the weight-EKF estimates the weights using the current state estimate $\hat{x}(k)$. Since the observation $y(k)$ is set to be one of the states, we just need to consider estimating the parameters associated with a single nonlinear function f.

Definition

The developed AFTC method of this chapter adopts the actuator fault model in the nonlinear AUV model that are defined as the system parameters w obtained from the dual-EKF, therefore, w has an appearance close to how the system state variables and control input appear in the model.

In fact, the LOE faults are estimated by the dual-EKF and this estimation will modify the model used by MPC. So, whenever a fault occurs in the actuators, the amplitude of the LOE fault is estimated by the parameter estimator of dual-EKF, which in turn modify the state estimator. Then, the modified state estimator predicts the future states in the prediction horizon. Therefore, the predictive controller accommodates the fault with this configuration.

The actuator fault w is assumed to have effects on the effectiveness coefficient matrix Γ as follows:

$$w_{m\times m} = \Gamma_{m\times m} - I_{m\times m},$$

where m is the number of system inputs, the matrix Γ is represented by equation (2.23) and

is a multiplicative matrix as follows:

$$\Gamma = \begin{bmatrix} \gamma^1 & 0 & 0 \\ 0 & \gamma^2 & 0 \\ 0 & 0 & \gamma^3 \end{bmatrix},$$

where $0 < \gamma^k < 1$, $k = 1, \ldots, 3$ denotes the effectiveness factor of the forces and torque in surge, sway and yaw motions.

Figure 4.1 presents the dual-EKF algorithm that consists of two EKFs running concurrently. The top EKF generates state estimates, and requires $\hat{w}(k-1)$ for the time update. The bottom EKF generates weight estimates, and requires $\hat{x}(k-1)$ for the measurement update.

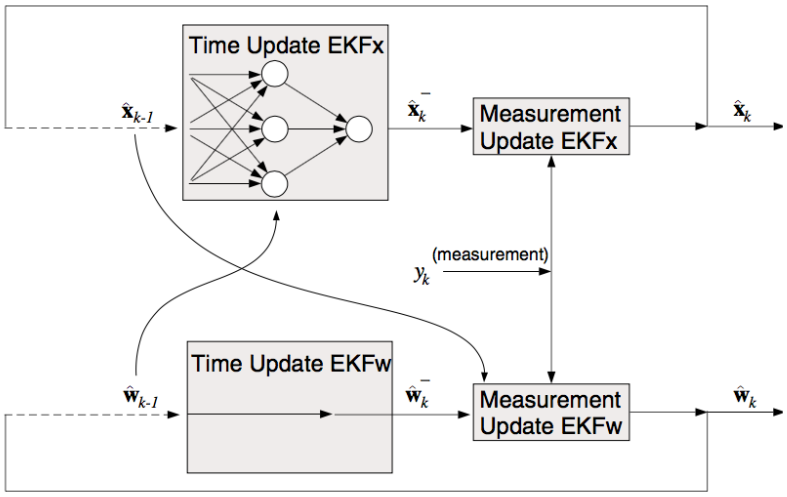


Figure 4.1: The dual Extended Kalman Filter. The top EKF generates state estimates, and requires $\hat{w}(k-1)$ for the time update. The bottom EKF generates weight estimates, and requires $\hat{x}(k-1)$ for the measurement update [12].

The dual-EKF algorithm [12] is presented in the Algorithm 2 in which the state estimation should be is used in the parameter estimation process. Therefore, at each time step, one EKF state filter estimates the current model estimate $\hat{w}(k)$. At the same time, the

other EKF parameter filter estimates the parameters using the current states estimate $\hat{x}(k)$.

Algorithm 2: Dual Extended Kalman Filter (dual-EKF) [12].

Input: Noisy y and faulty input u_f

Output: System states x and parameters w

Initialization

$\hat{w}_0 = E[w], P_{w_0} = E[(w_0 - \hat{w}_0) - (w_0 - \hat{w}_0)^T];$

$\hat{x}_0 = E[x], P_{x_0} = E[(x_0 - \hat{x}_0) - (x_0 - \hat{x}_0)^T];$

For $k \in \{1, \ldots, \infty\}$ the time-update equations for the weight filter are ;
$\hat{w^-}_k = \hat{w_{k-1}}^-;$

$\hat{P^-}_{w_k} = P_{w_{k-1}} + R^r{}_{k-1} = \lambda^{-1} P_{w_k}$;

and the time-update equations for the state filter are ;
$\hat{x^-}_k = \hat{x_{k-1}}^- + T_s f(\hat{x_{k-1}}^-);$

$\phi = I + T_s \bar{A}_{k-1}$;

$Q_d = (\phi R^v \phi^T + R^v)^{\frac{T_s}{2}};$

$\hat{P^-}_{x_k} = (\phi P^-{}_{x_{k-1}} \phi^T + Q_d)$

foreach *dual-EKF* **do**

the measurement updates equations for the state filter are

$K_k{}^x = P^-{}_{x_k} C^T (C \hat{P^-}_{x_k} C^T + R^n)^{-1}$

$\hat{x}_k = \hat{x^-}_k + K_k{}^x (y_k - C \hat{x^-}_k)$

$P_{x_k} = (I - K_k{}^x C) \hat{P^-}_{x_k}$

and the measurement updates equations for the weight filter are

$K_k{}^w = P^-{}_{w_k} (C_k{}^w)^T (C_k{}^w \hat{P^-}_{x_k} (C_k{}^w)^T + R^e)^{-1}$

$\hat{w}_k = \hat{w^-}_k + K_k{}^w . \epsilon_k$

$P_{w_k} = (I - K_k{}^w C_k{}^w) \hat{P^-}_{w_k}$

where

$\epsilon_k = (y_k - C \hat{x^-}_k)$

$\bar{A}_{k-1} = \frac{\partial f(x)}{\partial x}|_{x = \hat{x}_{k-1}}$

$C_k{}^w = -\frac{\partial \epsilon_k}{\partial w} = C \frac{\partial \hat{x_k}^-}{\partial w}|_{w = \hat{w}_k}$

end

P_{w_k} and P_{x_k} are the parameter part and the state part of the covariance matrix P_k, respectively. There are several choices on how to select matrix ${R^r}_k$ [12]. We set ${R^r}_k = ({\lambda_{ff}}^{-1} - 1)P_{w_k}$, where $\lambda_{ff} \in (0, 1]$ is often referred to as the forgetting factor. This provides an approximate of an exponentially decaying weighting on past data (More detail is available in [198]). Therefore, P_{x_k} gets updated by the forgetting factor, at each time interval, as follows:

$$P_{x_k} = {\lambda_x}^{-1} P_{x_k}.$$

R^n is the measurement noise and is set to be Gaussian, R^v is the measurement noise covariance matrix, Q_d and ϕ are auxiliary parameters, ${K_k}^x$ and ${K_k}^w$ are the Kalman gains related to state and parameter filters, respectively, ϵ is the error term, and ${C_k}^w$ is the parameter filter.

Algorithm 2 illustrates that $\hat{x}(k)$ is a function of $\hat{x}(k-1)$, and both are functions of system parameters w. Therefore, there is a recurrent architecture with a method similar to the real-time recurrent learning, in the linearization process of the parameter filter ${C_k}^w$. Thus, the following system of recursive equations are given [12]

$$\frac{\partial \hat{x}_{k+1}^-}{\partial \hat{w}} = \frac{\partial f(\hat{x}, \hat{w})}{\partial \hat{x}_k} \frac{\partial \hat{x}_k}{\partial \hat{w}} + \frac{\partial f(\hat{x}, \hat{w})}{\partial \hat{w}_k}, \tag{4.5}$$

$$\frac{\partial \hat{x}_k}{\partial \hat{w}} = (I - {K_k}^x C) \frac{\partial \hat{x}_k^-}{\partial \hat{w}} + \frac{\partial {K_k}^x}{\partial \hat{w}} (y_k - C\hat{x}_k^-), \tag{4.6}$$

where $\frac{\partial f(\hat{x}, \hat{w})}{\partial \hat{x}_k}$ and $\frac{\partial f(\hat{x}, \hat{w})}{\partial \hat{w}_k}$ contain static linearization of the nonlinear function at $\hat{w}_k$. The derivative of ${K_k}^x$ with respect to the i-th element of $\hat{w}$ by $\frac{\partial {K_k}^x}{\partial \hat{w}}$ (the i-th column of $\frac{\partial {K_k}^x}{\partial \hat{w}(i)}$) yields

$$\frac{\partial {K_k}^x}{\partial \hat{w}(i)} = \frac{(I - {K_k}^x C)}{C {P_{xk}}^- C^T + R^n} \frac{\partial {P_{xk}}^-}{\partial \hat{w}(i)} C^T \tag{4.7}$$

When the dual-EKF has estimated the states and parameters of the system that are

affected by the faulty input control $w\tau = \tau_f$, and $u_f = [w_1\tau_1,\ w_2\tau_2,\ w_3\tau_3]$, the problem formulation of the nonlinear prediction (see Section 2.7) with the conventional NMPC formulation a set of future control increments is calculated at each consecutive sampling instant k as follows:

$$\Delta u_f(k) = [\Delta u_f(k|k)\ \ldots\ \Delta u_f(k+N_u-1|k)]^T, \tag{4.8}$$

and the quadratic cost function (used from Chapter 3 equation (3.2)) given by

$$J(k) = \sum_{p=1}^{N} Q_p(y^{ref}(k+p|k) - \hat{y}(k+p|k))^2 + \sum_{p=1}^{N_u-1} R_p(\Delta u_f(k+p|k))^2. \tag{4.9}$$

It is assumed that $\Delta u_f(k+p|k) = 0$ for $p \geq N_u$, where N and N_u are prediction horizon $(1 \leq N)$ and control horizon $(0 < N_u < N)$, respectively, $y^{ref}(k+p|k)$ is the reference trajectory of output signal and $\hat{y}(k+p|k)$ is the predicted values of the output over the prediction horizon $N > N_u$, $\Delta u_f(k+j|k)$ denotes the input increment, and $\Delta u_f(k+pk) = u_f(k+pk) - u_f(k+p-1k)$, $Q_p > 0$ and $R_p > 0$ are weighting factors.

Objective Minimize the differences between the reference trajectory of the output signal $y^{ref}(k+p|k)$ and the predicted values of the output $\hat{y}(k+p|k)$ over the prediction horizon $N > N_u$, as well as considering the fact that the excessive control increments should be penalized. Also, w at each time instant is a vector of dimension $m \times 1$ that defines the system measurement of the effectiveness of each actuator, m is the number of actuators.

The first element of the determined sequence in equation (4.8) is applied to the system and the contol decosion in faulty condition $u_f(k)$ is given by

$$u_f(k) = \Delta u_f(k|k) + u_f(k-1).$$

At the next sampling instant $k+1$ the prediction is shifted one step forward, the

output measurement is updated, and finally the whole procedure is repeated. Since problem constraints have to be usually taken into account, future control increments are determined from the following optimization problem:

$$\min_{\Delta u_f(k|k) \ldots \Delta u_f(k+N_u-1|k)} J(k)$$

subject to the following constraints:

$$\begin{aligned} u_{min} \leq u_f(k+p|k) \leq u_{max}, p = 0, \ldots, N_u - 1 \\ -\Delta u_{max} \leq \Delta u_f(k+p|k) \leq \Delta u_{max}, p = 0, \ldots, N_u - 1 \\ y_{min} \leq \hat{y(k+p|k)} \leq y_{max}, p = 0, \ldots, N_u - 1 \end{aligned}$$

where u_f is the updated control input with regards to the LOE fault happened on the actuators. The general prediction equation for $p = 1, \ldots, N$ is

$$\hat{y}(k+p|k) = y(k+p|k) + d(k)$$

$y(k+p|k)$ is calculated from a dynamic model of the system and $d(k) = y(k) - y(k|k-1)$.

Prediction vectors $\hat{y}(k+p|k)$ are nonlinear functions of future control moves [188]. Again, the output prediction $\hat{y}(k)$ is expressed as the sum of a forced trajectory, depending only on the future and the free trajectory $y^0(k)$ that is based on the past only.

$$\hat{y}(k) = M_d{}'(k)\Delta u_f(k) + y^0(k),$$

where $\hat{y}(k)$ are vectors of length rN and presents output prediction, r is the number of outputs and N is the prediction horizon, $\Delta u_f(k)$ is the future input moves and it is in the form of equation (4.8), $\Delta u_f(k)$ is a vector of length mN_u while m is the number of inputs

and N_u is the control horizon. The free trajectory $y^0(k)$ is as follows:

$$y^0(k) = [y^0(k+1|k) \ \dots \ y^0(k+N|k)].$$

The dynamic matrix $M_d{}'(k)$ of dimensionality $rN \times mN_u$ is comprised of step-response coefficients of the linearized model (obtained from dual-EKF) as follows:

$$M_d{}'(k) = \begin{bmatrix} B_k\Gamma & 0 & \dots & 0 \\ A_kB_k\Gamma & B_k\Gamma & \dots & 0 \\ A_k^2B_k\Gamma & A_kB_k\Gamma & B_k\Gamma & \vdots \\ A_k^{N-1}B_k\Gamma & A_k^{N-2}B_k\Gamma & \dots & A_k^{N-N_u}B_k\Gamma \end{bmatrix}.$$

Similar to our approach in the previous chapter, both the free trajectory $y^0(k)$ and the dynamic matrix $M_d{}'(k)$ are calculated online from the current states of system by dual-EKF. Therefore, the controller calculates the matrix $B_k\Gamma$ and updates the system. Consequently, the dynamic matrix will be changed and the LOE fault get accommodated in an online manner. The Jacobian matrix A_k is comprised of recurrent derivatives $\frac{\partial f(\hat{x},\hat{w})}{\partial \hat{x}_k}$ and $\frac{\partial f(\hat{x},\hat{w})}{\partial \hat{w}_k}$. According to [12], $\bar{A}(k-1)$ in Algorithm 2 depends not only on the parameters $\hat{w}$, but also on the operating point of linearization $\hat{x}(k-1)$. $\frac{\partial \bar{A}(k-1)}{\partial \hat{w}(i)}$ contains a 3-D tensor given by

$$\frac{\partial \bar{A}(k-1)}{\partial \hat{w}(i)} = \frac{\partial^2 F}{\partial \hat{x}_{k-1}\partial \hat{w}(i)} + \frac{\partial^2 F}{(\partial \hat{x}_{k-1})^2}\frac{\partial \hat{x}_{k-1}}{\partial \hat{w}(i)}, \tag{4.11}$$

where $\frac{\partial \hat{x}_{k-1}}{\partial \hat{w}(i)}$ is ${C_{k-1}}^w$.

The matrices A_k, $B_k\Gamma$, and the rest of derivative terms for implementing the dual-EKF

are defined as follows:

$$\frac{\partial f}{\partial \hat{x}}|_{(\hat{x}(k),\hat{w}(k))} = \begin{bmatrix} 0 & 0 & -\hat{v}\cos(\hat{\psi}) - \hat{u}\sin(\hat{\psi}) & \cos(\hat{\psi}) & -\sin(\hat{\psi}) & 0 \\ 0 & 0 & \hat{u}\cos(\hat{\psi}) - \hat{v}\sin(\hat{\psi}) & \sin(\hat{\psi}) & \cos(\hat{\psi}) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{-d_u}{m_u} & \frac{m_v}{m_u}\hat{r} & \frac{m_v}{m_u}\hat{v} \\ 0 & 0 & 0 & \frac{-m_u}{m_v}\hat{r} & \frac{d_v}{m_v}\hat{r} & \frac{-m_u}{m_u}\hat{u} \\ 0 & 0 & 0 & \frac{m_u - m_v}{m_r}\hat{v} & \frac{m_u - m_v}{m_r}\hat{u} & \frac{-d_r}{m_r} \end{bmatrix},$$

$$\frac{\partial f}{\partial \hat{w}(k)}|_{(\hat{x}(k),\hat{w}(k))} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \tau_1\frac{1}{m_u} & 0 & 0 \\ 0 & \tau_2\frac{1}{m_v} & 0 \\ 0 & 0 & \tau_3\frac{1}{m_r} \end{bmatrix},$$

$$\bar{B}\Gamma = \frac{\partial f}{\partial u_f}|_{(\hat{x}(k),\hat{u}_f(k))} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ w_1\frac{1}{m_u} & 0 & 0 \\ 0 & w_2\frac{1}{m_v} & 0 \\ 0 & 0 & w_3\frac{1}{m_r} \end{bmatrix},$$

$$\frac{\partial^2 f}{\partial \hat{x}(k-1)\partial \hat{w}(i)}|_{(\hat{x}(k),\hat{w}(k))} = 0.$$

As mentioned earlier, $\frac{\partial^2 F}{(\partial \hat{x}_{k-1})^2}$ is a three-dimensional tensor given by 6 matrices of

dimension 6×6 which are given as:

$$\frac{\partial^2 F}{\partial \hat{x}^2}_1 = [0]_{6\times 6},$$

$$\frac{\partial^2 F}{\partial \hat{x}^2}_2 = [0]_{6\times 6},$$

$$\frac{\partial^2 F}{\partial \hat{x}^2}_3 = \begin{bmatrix} 0 & 0 & \hat{v}\sin(\hat{\psi}) - \hat{u}\cos(\hat{\psi}) & -\sin(\hat{\psi}) & -\cos(\hat{\psi}) & 0 \\ 0 & 0 & -\hat{u}\sin(\hat{\psi}) - \hat{v}\cos(\hat{\psi}) & \cos(\hat{\psi}) & -\sin(\hat{\psi}) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\frac{\partial^2 F}{\partial \hat{x}^2}_4 = \begin{bmatrix} 0 & 0 & -\sin(\hat{\psi}) & 0 & 0 & 0 \\ 0 & 0 & \cos(\hat{\psi}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{m_u}{m_v} \\ 0 & 0 & 0 & 0 & \frac{m_u - m_v}{m_r} & 0 \end{bmatrix},$$

$$\frac{\partial^2 F}{\partial \hat{x}^2}_5 = \begin{bmatrix} 0 & 0 & -\cos(\hat{\psi}) & 0 & 0 & 0 \\ 0 & 0 & -\sin(\hat{\psi}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{m_v}{m_u} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{m_u - m_v}{m_r} & 0 \end{bmatrix},$$

$$\frac{\partial^2 F}{\partial \hat{x}^2}_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{m_v}{m_u} & 0 \\ 0 & 0 & 0 & -\frac{m_u}{m_v} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

Therefore, the dual-EKF has estimated both states and parameters of the system, and similar to the Section 3.2, the optimization problem in the cost function (equation (4.9)) converts to a QP problem task as follows:

$$\min_{\Delta u_f(k)} \parallel y_{ref}(k) - M_d{}'(k)\Delta u(k) - y^0(k) \parallel_Q^2 + \parallel \Delta u_f(k) \parallel_R^2$$

subject to

$$u_{min} \leq J^{NPL}\Delta u_f(k) + u^{NPL}(k) \leq u_{max}$$

$$-\Delta u_{max} \leq \Delta u_f \leq \Delta u_{max}$$

$$y_{min} \leq M_d{}'(k)\Delta u_f(k) + y^0(k) \leq y_{max}$$

where y_{ref}, y_{min} and y_{max} are vectors of length N, and present the reference trajectory, minimum constraint, and maximum constraint output vectors, respectively, vectors of minimum and maximum control inputs, and input increments, respectively, u_{min}, u_{max}, and Δu_{max} are vectors of length N_u, Q and R are matrices of the sizes $N \times N$ and $N_u \times N_u$, respectively, R is the matrix of weights in MPC given by $R = diag(\lambda_0, \ldots, \lambda_{N_u-1})$. $u^{NPL}(k)$ is an auxiliary vector length N_u and J^{NPL} is an auxiliary matrix of dimensionality $N_u \times N_u$

as follows:

$$J^{NPL}(i,j) = \begin{cases} 1, \ i = j \ or \ i = j+3 \\ 0, \text{others} \end{cases}$$

Similar to the previous chapter, $x'_{QP} = \Delta u_f^T(k)$ is considered to be a vector containing all decision variables of the MPC algorithm, and correspondingly the optimization problem (equation (4.12)) is rewritten in a standard QP form as follows:

$$\begin{aligned} \min \frac{1}{2} {x'_{QP}}^T H'_{QP} x_{QP} + {f'_{QP}}^T x'_{QP} \\ \text{s.t.} \quad {M_d'}^T(k)\Delta u(k) = \hat{y}(k) - y^0(k) \\ A'_{QP} x_{QP} \leq b'_{QP} \\ -\Delta u_{max}^T \leq x'_{QP} \leq \Delta u_{max}^T \\ \text{where} \quad \Delta u^T(k) = x'_{QP}, \end{aligned}$$

The cost function in the QP problem (equation (4.12)) has a unique optimal solution and the objective function is strictly convex. H'_{QP} and f'_{QP} in equation ( ) are given by

$$H'_{QP} = 2({M_d'}^T(k) Q M_d'(k) + R), \tag{4.13}$$

$$f'_{QP} = -2({M_d'}^T(k) y^{ref}(k) - y^0(k)), \tag{4.14}$$

where Q and R are matrices of the sizes $N \times N$ and $N_u \times N_u$, respectively. R is the matrix of weights in MPC given by $R = diag(\lambda_0, \ldots, \lambda_{N_u-1})$, and y_{ref} presents the reference

trajectory. A'_{QP} and b'_{QP} in equation ( ) are defined as

$$A'_{QP} = \begin{bmatrix} -J^{NPL} \\ J^{NPL} \\ -CM_d{}'(k) \\ CM_d{}'(k) \\ -I_{N_u \times N_u} \\ I_{N_u \times N_u} \end{bmatrix}, \tag{4.15}$$

$$b'_{QP} = \begin{bmatrix} -u_{min} + u_{f_{k-1}}(k) \\ u_{max} - u_{f_{k-1}}(k) \\ -y_{min} + y^0(k) \\ y_{max} - y^0(k) \\ \Delta u_{max} \\ \Delta u_{max} \end{bmatrix}, \tag{4.16}$$

Therefore, a recurrent neural network is developed with the dynamics detailed in equation (3.15). The reader is referred to the Section 3.2 (equations (3.12) to (3.15)) for rest of the proof of neural networks convergence motivated from [191].

The implementation of the developed hybrid of MPC and RNN active fault-tolerant control method is described in Figure 4.2 .

4.2.1 The Overall Developed AFTC Scheme

The following steps should be repeated at each sampling instant for the AFTMPC-RNN algorithm to work:

- Let $k = 1$ and set the control time terminal T, prediction horizon N, control horizon N_u, control time terminal T, sampling interval T_s, weight matrices Q and R.

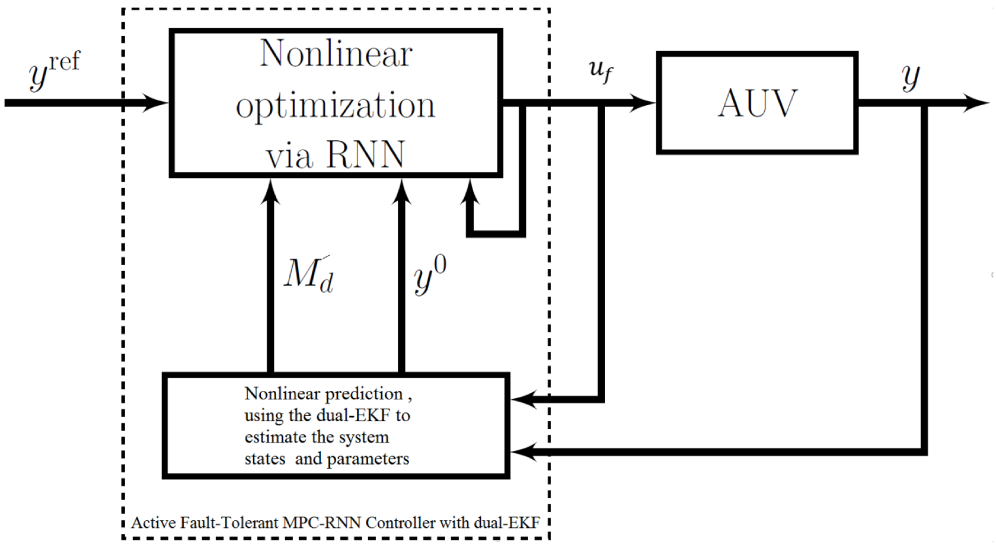


Figure 4.2: The developed AFTC methodology.

- Calculate the nonlinear free trajectory $y^0(k)$ given in equation (2.41). Update the system model and Calculate the nonlinear free trajectory $y^0(k)$ given in equation (2.41) using dual-EKF

- Calculate the dynamic matrix $M_d{}'(k)$ using the updated model. Set NN parameters, H'_{QP}, f'_{QP}, A'_{QP}, R_y, q and W (given in equations (3.7), (3.8), (4.15), and (3.14)).

- Solve the convex quadratic minimization problem given in equation (3.6) to obtain the optimal control action Δu_k by using an RNN, with $3N_u m + 2N$ neurons, where m denotes the number of system inputs and n denotes the number of system states.

- Apply the optimal input vector $u(k)$ given in equation (2.35).

- If $k < T$, set $k = k + 1$ and go to the second step; otherwise end.

The proposed method in this chapter achieves the stability by proper tuning of the prediction horizon and weighting coefficients λ_{RNN}.

4.3 Application to AUV Mission and Simulation Results

Our proposed AFTC integrates the dual-EKF estimations as well as the control method proposed in Chapter 3. In this chapter, the goal is to design a robust MPC-RNN controller that inherits the stability properties of a model-based controller. The proposed solution isolates and identifies the severity of faults in the system within a single integrated framework. For evaluation purposes, the proposed AFTC method is applied on an AUV.

4.3.1 Mission Objective

The control objective is to force the AUV to follow a pre-defined trajectory of positions and orientations while keeping its physical constraints considered. AUV is faced with various LOE fault scenarios in its actuators while controling surge, sway and yaw motions. Therefore, the performance runtime, average control cost and the estimation of system from the actuator faults are the important parameters that are taken into account for further improvements.

- First, we aim to show how applying FTC changes the performance of a system faced with mild and severe actuator faults. Therefore, in Section 4.3.2 a series of scenarios are designed to make a comparison between our proposed method with and without FTC feature. In the first experiment, it is assumed that the initial position, orientation and velocity of the AUV are set to $\eta_0 = [0, 0, \frac{\pi}{15}, 0, 0, 0]^T$, and the desired tracking mission's surge, sway and yaw are $\eta_d = [0.5\cos(t/3/\pi) + 1.5, 2.5, \pi/3, 0, 0, 0]^T$. In the first experiment, the objective is improving the average control cost while reducing the steady-state errors in path tracking, fault detection and control recovery of actuator faults.

- Second, demonstrating a comparison between our developed AFTMPC-RNN and FT-NMPC methods, several scenarios are considered in Section 4.3.3. In the second experiment, it is assumed that the initial position, orientation and velocity of the AUV are set to $\eta_0 = [0, 0, \frac{\pi}{15}, 0, 0, 0]^T$ and the desired tracking mission's surge, sway and yaw are $\eta_d = [0.5\cos(t/3/\pi) + 1.5, 2.5, \pi/3, 0, 0, 0]^T$.

Simulation results are presented in both set of experiments to demonstrate the efficiency of the developed scheme.

The physical constraints of the AUV dynamic model in both experiments is summarized in Table 3.1. For initializations of MPC algorithm, the output penalty matrix is set to be $Q = 5 \times 10^4 \times I_{6\times6}$ and the control penalty matrix $R = I_{3\times3}$. The prediction horizon assumed to be $N = 20$ and the control horizon is $N_u = 5$. The maximum thruster force along X and Y axes are assumed to be $400N$ and the maximum thruster torque of yaw axis is set to be $100Nm$. The inertia matrix M, the Coriolis and centripetal matrix $C(\nu)$, and the damping matrix $D(\nu)$ in equation (2.3) are provided as follows:

$$M = \begin{bmatrix} 25.8 & 0 & 0 \\ 0 & 33.8 & 1.0115 \\ 0 & 1.0115 & 2.76 \end{bmatrix},$$

$$C(\nu) = \begin{bmatrix} 0 & 0 & -(33.8v + 1.0115r) \\ 0 & 0 & 25.8u \\ 33.8v + 1.0115r & -25.8u & 0 \end{bmatrix}$$

$$D(\nu) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 7 & 0.1 \\ 0 & 0.1 & 0.5 \end{bmatrix}$$

The simplified dynamic model consists of six state variables and three input variables. AUV system states are the positions on X and Y axes and its orientation about Z axis. The effect of environmental disturbances due to irrational ocean currents are gained from equation (2.19) and are described as slowly varying drift forces acting on the input channels of AUV along X and Y axes.

The implementation of the EKF algorithm requires the noise variance ${\sigma^2}_v$ and ${\sigma^2}_n$ that are commonly determined from physical knowledge of the problem (e.g. sensor accuracy or ambient noise measurements) [12]. In the initializations of the dual-EKF it is assumed that $\lambda_w = \lambda_x = 0.999$. The covariance matrices are initialized with larger values of diagonal matrix entities comparing to what we have previously assumed in Section 3.4. $P_x = I_{6\times6}$,$P_w = I_{3\times3}$, $R_n = diag[5\times10^{-2}, 5\times10^{-2}, 5\times10^{-2}, 1\times10^{-2}, 1\times10^{-2}, 1\times10^{-2}]^2$, $R_e = R_n$ and $Q_{dEKF} = diag[5\times10^{-2}, 5\times10^{-2}, 5\times10^{-2}, 1\times10^{-2}, 1\times10^{-2}, 1\times10^{-2}]^2$.

The matrix of unknown nonlinear uncertainty from equation (2.3) is considered as the diagonal matrix of square roots of matrix entities from the measurement covariance matrix multiplied by $rand(1, p)$, where p is the number of system output states.

$$g(\eta) = diag[sqrt(R_c) * rand(1, p)]$$

The initialization of matrix ${C_k}^w$ is as follows:

$$
{C_k}^w = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ X\frac{1}{m_u} & 0 & 0 \\ 0 & Y\frac{1}{m_u} & 0 \\ 0 & 0 & N\frac{1}{m_u} \end{bmatrix}
$$

where X, Y, ans N are defined in Table 2.1.

For RNN initialization, $\lambda_{RNN} = 100$, $\Delta u_0 = 0 \times ones(N, 1)$, $\mu_0 = 0$, $N = 20$, $N_u = 5$, and 85 neurons in the RNN architecture.

4.3.2 First Experiment: The Effect of Making the Developed MPC-RNN Control System Fault-Tolerant

Consider the equation (2.1a), the initial position, orientation and velocity of the AUV are set to $\eta_0 = [0, 0, \frac{\pi}{15}, 0, 0, 0]^T$ and the desired tracking mission's surge, sway and yaw are $\eta_d = [0.5\cos(t/3/\pi) + 1.5, 2.5, \pi/3, 0, 0, 0]^T$. During this mission, the AUV is faced with various LOE scenarios in its actuators. The coresponding effectiveness coefficient matrices are as follows:

- In the first scenario of Section 4.3.2, the AUV's actuator along X and about Z axes work with 10% of their efficiency and the actuator along Y axis works with 50% of its

efficiency,

$$\Gamma_1 = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}.$$

- In the second scenario of Section 4.3.2, the AUV's actuator along X axis is fully working and the actuators along Y and around Z axes are working with 10% of their efficiency,

$$\Gamma_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix},$$

- In the third scenario, the AUV's actuators along X and Y axes are fully working and the actuator around Z axis is working with 10% of its efficiency,

$$\Gamma_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix},$$

where t_d represents the the time takes for the FDI module to detect and identify the fault severity and it is considered as the performance runtime of the system. The simulation results are obtained for a duration of $T = 200s$ with $T_s = 0.2s$. As discussed in Section 4.3.1, in the first experiment , the objective is improving the average control cost while reducing the steady-state errors in path tracking, fault detection and control recovery of actuator faults.

First Scenario

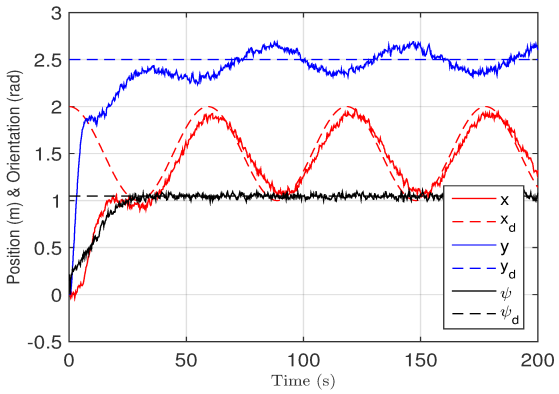
In this scenario, the AUV is forced to do the mission discussed in Section 4.3.2 while its actuators along X and around Z axes are working with 10% of their efficiency and the actuator along Y axis is working with 50% of its efficiency. The effectiveness coefficient matrix Γ_1 of this scenario is given by

$$\Gamma_1 = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}.$$

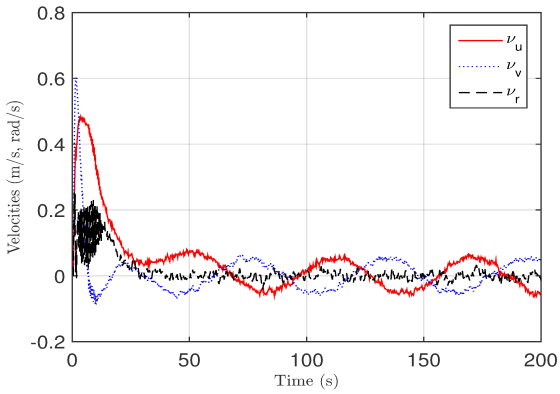
The main objectives of this mission are discussed in Section 4.3.1. Figures 4.3 and 4.4 illustrate the slight improvements of the results in case of using our AFTC method with regards to the objectives of this mission. As shown, the system is not capable of measuring the actuator effectiveness when no FTC applied. The goal of our developed methodology is to fault detection and control recovery of the system when faced with LOE faults. Figure 4.4 illustrates that the developed AFTC method accomplished this goal. Note that in this path following mission, the aim is to just control the position and orientation states. the states regarding linear and rotational velocities are just given as information.

Table 4.1: The LOE fault in actuators under different scenarios. A comparison between the performance runtime t_d.

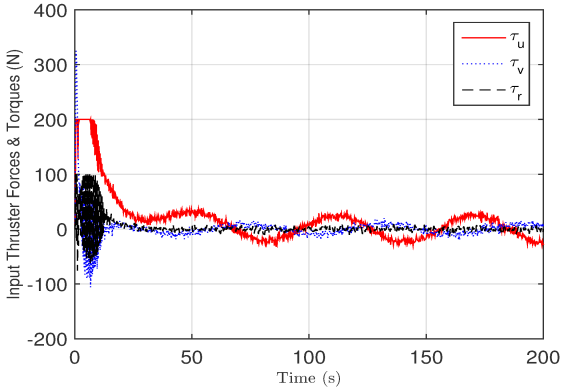
t_d performance runtime	no FTC	proposed FTC
Γ_1	$0.0233s$	$0.0164s$
Γ_2	$0.0167s$	$0.0164s$
Γ_3	$0.0216s$	$0.0165s$



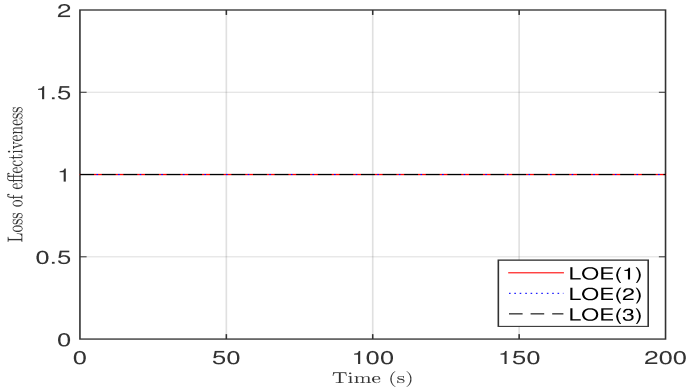
(a) Position and orientation state signals.



(b) Velocity state signals.

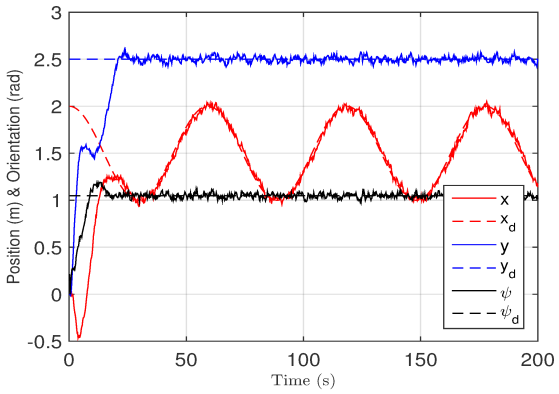


(c) Control input signals.

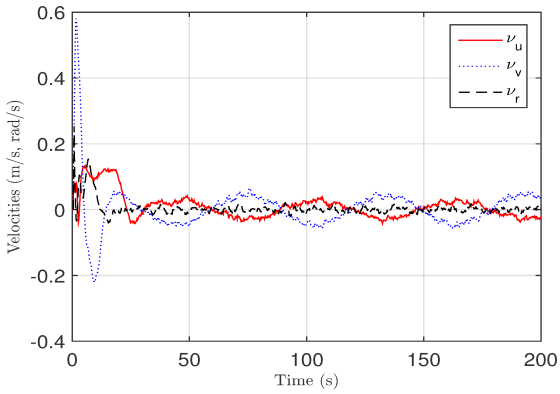


(d) Actuators' effectivness of an AUV that is not equipped with dual-EKF algorithm during the fault scenario Γ_1.

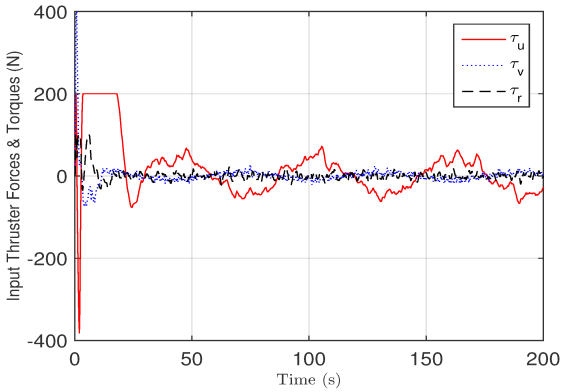
Figure 4.3: States and parameter signals of an AUV that is not equipped with dual-EKF algorithm with 90% LOE in its actuators along X and around Z axes, and 50% LOE in its actuator along Y axis.



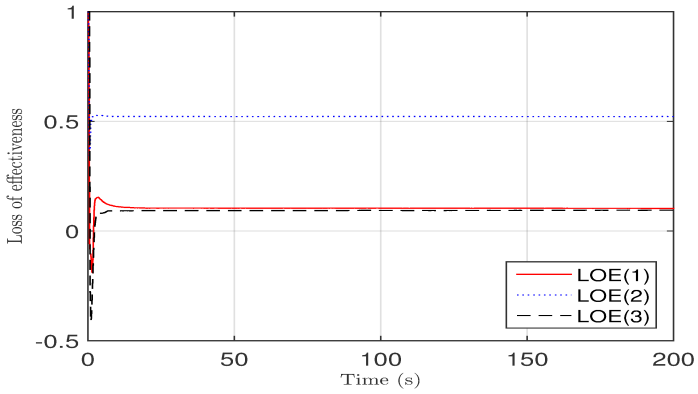
(a) Position and orientation state signals.



(b) Velocity state signals.



(c) Control input signals.



(d) Actuators' effectivness of proposed AFTC method during the fault scenario Γ_1.

Figure 4.4: States and parameter signals of the proposed AFTC scheme applied on an AUV with 90% LOE in its actuators along X and around Z axes, and 50% LOE in its actuator along Y axis.

Table 4.1 shows that our proposed AFTC method has slightly improved the performance runtime of the AUV control system which affected the performance runtime corresponding to the control, fault detection and recovery procedures. Note that in this set of experiment, our objective is to show that the added FTC features to the system do not improve the performance runtime of the system.

Table 4.2: MSE for 1000 sampling instants of position and orientation system state during Γ_1 fault scenario.

System States	no FTC	proposed FTC
Position along X axis	7.8820×10^{-4}	5.5105×10^{-4}
Position along Y axis	6.4589×10^{-4}	5.0792×10^{-4}
Orientation around Z axis	5.6147×10^{-4}	5.9886×10^{-4}

Table 4.2 illustrates the slight improvement that we achieved in decreasing the MSE in position system states by using the dual-EKF algorithm. In this scenario, there exist a 90% LOE in the AUV actuators along X and around Z axes, and 50% LOE in the AUV actuator along Y axis.

Table 4.3: Steady-state error of position and orientation system state during Γ_1 fault scenario.

System States	no FTC	proposed FTC
Position along X axis	0.2312	0.1161
Position along Y axis	0.1678	0.0101
Orientation about Z axis	0.0426	0.0197

Table 4.3 illustrates the improvement that we achieved in decreasing the steady-state error in position and orientation system states by using the dual-EKF algorithm. In this scenario, there exist a 90% LOE in the AUV actuators along X and around Z axes, and 50% LOE in the AUV actuator along Y axis.

Table 4.4: Average Control cost during 1000 sampling instants in scenario Γ_1.

Control Methods	no FTC	proposed FTC
Control cost of X	1.2306×10^2	1.4092×10^2
Control cost of Y	7.6793×10^2	1.4935×10^2
Control cost of N	1.0016×10^3	3.7247×10
Total Control cost	1.8926×10^3	3.2751×10^2

Consider the fault scenario depicted in the effectiveness coefficient matrix Γ_1, Table 4.4 shows that the proposed AFTC in this chapter improves the performance runtime as well as decreasing the MSE and steady-sate error while resulting in lower costs for the AUV controller in comparison with the AUV controller that does not equipped with FTC.

Second Scenario

In this scenario, the AUV is forced to perform the mission discussed in Section 4.3.2, while its actuator along X axis is fully working and AUV's actuators along Y and around Z axes are working with 10% of their efficiency,

$$\Gamma_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix},$$

The main objectives of this mission are discussed in Section 4.3.1. Figures 4.5 and 4.6 illustrate the improvement of the results in case of using our AFTC method with regards to the objectives of this mission. As shown, the system is not capable of measuring the actuator effectiveness when no FTC applied. The goal of our developed methodology is to perform fault detection and control recovery of the system when faced with LOE faults. Figure 4.6 illustrates that the developed AFTC method accomplished this goal. Note that

in this path following mission, the aim is to just control the position and orientation states. the states regarding linear and rotational velocities are just given as information.

Table 4.5: MSE for 1000 sampling instants of position and orientation system state during Γ_2 fault scenario.

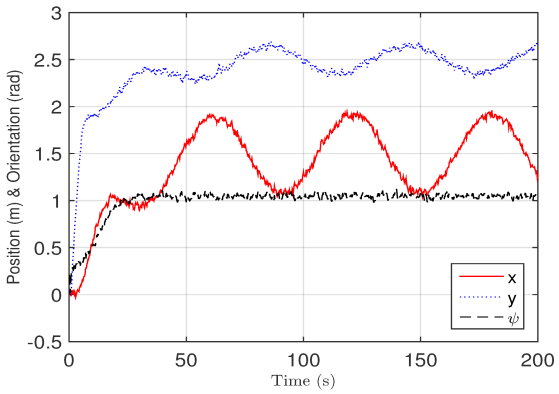
System States	no FTC	proposed FTC
Position along X axis	5.6113×10^{-4}	4.9173×10^{-4}
Position along Y axis	6.1010×10^{-4}	4.8442×10^{-4}
Orientation around Z axis	6.5109×10^{-4}	6.0294×10^{-4}

Table 4.5 illustrates the slight improvement that we achieved in decreasing the MSE in position and orientations system states by using the dual-EKF algorithm. In this scenario, the AUV actuator along X axis is fully working and actuators along Y and around Z axes have 90% LOE.

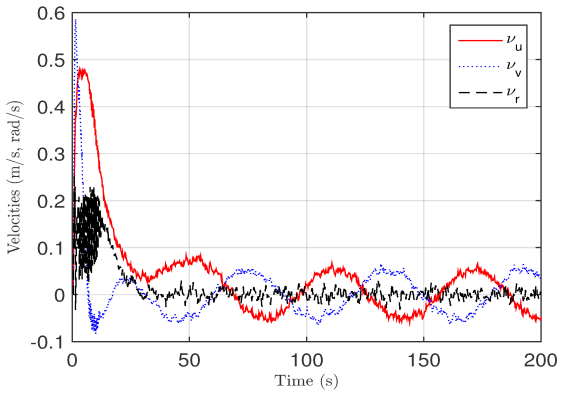
Table 4.6: Steady-state error of position and orientation system state during Γ_2 fault scenario.

System States	no FTC	proposed FTC
Position along X axis	0.3948	0.1446
Position along Y axis	0.1684	0.0111
Orientation around Z axis	0.0303	0.0211

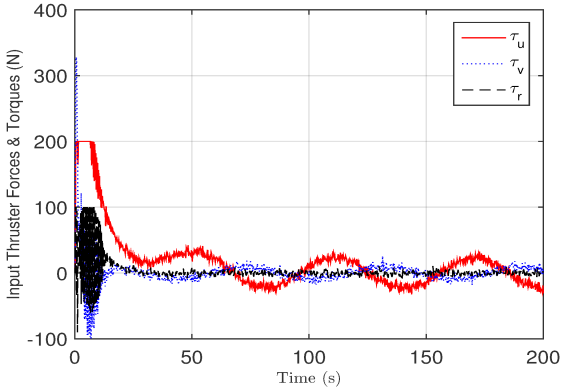
Table 4.6 illustrates the improvement that we achieved in decreasing the steady-state error in position and orientation system states by using the dual-EKF algorithm. In this scenario, the AUV actuator along X axis is fully working and actuators along Y and around Z axes have 90% LOE.



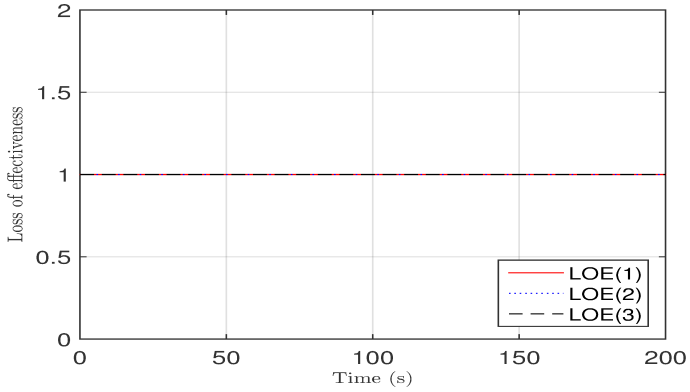
(a) Position and orientation state signals.



(b) Velocity state signals.

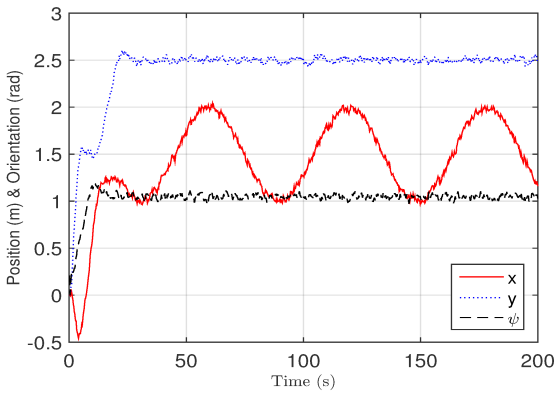


(c) Control input signals.

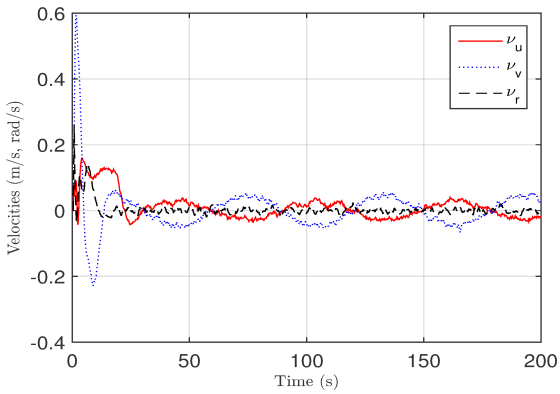


(d) Actuators' effectivness of an AUV not equipped with dual-EKF algorithm during the fault scenario Γ_2.

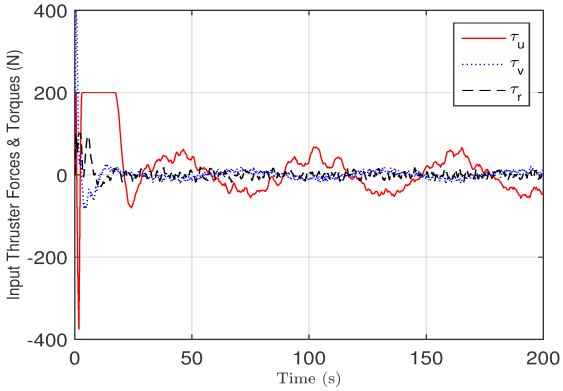
Figure 4.5: States and parameter signals of an AUV not equipped with dual-EKF algorithm when its actuator along X axis is fully working and actuators along Y and around Z axes have 90% LOE.



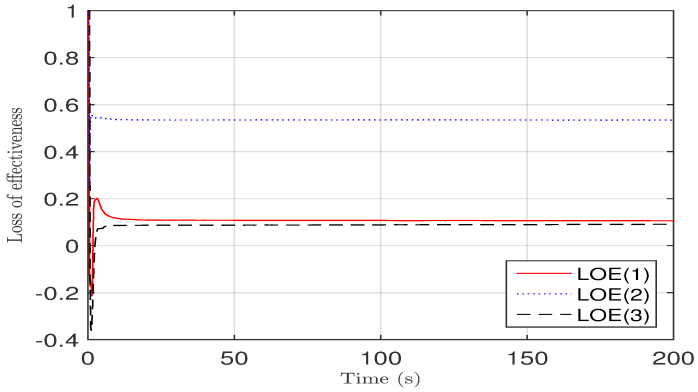
(a) Position and orientation state signals.



(b) Velocity state signals.



(c) Control input signals.



(d) Actuators' effectivness of proposed AFTC method during the fault scenario Γ_1.

Figure 4.6: States and parameter signals of the proposed AFTC scheme applied on an AUV equipped with our proposed AFTC algorithm when its actuator along X axis is fully working and actuators along Y and around Z axes have 90% LOE.

Table 4.7: Average Control cost during 1000 sampling instants in scenario Γ_2.

Control Methods	no FTC	proposed FTC
Control cost of X	5.3013×10^2	7.3396×10
Control cost of Y	1.4069×10^3	2.1712×10^2
Control cost of N	5.6914×10^2	7.7058×10
Total Control cost	2.4984×10^3	3.6757×10^2

Consider the fault scenario depicted in the effectiveness coefficient matrix Γ_2, Table 4.7 shows that the proposed AFTC in this chapter improves the performance runtime as well as decreases the MSE and steady-sate error while resulting in lower costs for the AUV controller in comparison with the AUV controller that is not equipped with FTC.

Third Scenario

In this scenario, the AUV is forced to perform the mission discussed in Section 4.3.2, while the AUV's actuators along X and Y axes are fully working and the actuator around Z axis is working with 10% of its efficiency,

$$\Gamma_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}.$$

The main objectives of this mission are discussed in Section 4.3.1. Figures 4.7 and 4.8 illustrate the improvement of the results in case of using our AFTC method with regards to the objectives of this mission. As shown, the system is not capable of measuring the actuator effectiveness when no FTC applied. The goal of our developed methodology is to perform fault detection and control recovery of the system when faced with LOE faults, as one of its objectives. Figure 4.8 illustrates that the developed AFTC method accomplished

this goal. Note that in this path following mission, the aim is to just control the position and orientation states. the states regarding linear and rotational velocities are just given as information.

Table 4.8: MSE for 1000 sampling instants of positions and orientation system states during Γ_3 fault scenario.

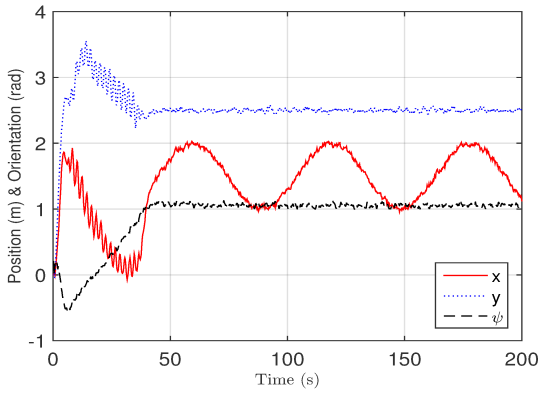
System States	no FTC	proposed FTC
Position along X axis	5.6306×10^{-4}	5.0859×10^{-4}
Position along Y axis	5.3400×10^{-4}	4.9815×10^{-4}
Orientation around Z axis	0.0017	6.2688×10^{-4}

Table 4.8 illustrates the slight improvement that we achieved in decreasing the MSE in position and orientations system states by using the dual-EKF algorithm. In this scenario, the AUV actuator along X and Y axesare fully working and its actuator around Z axis has 90% LOE.

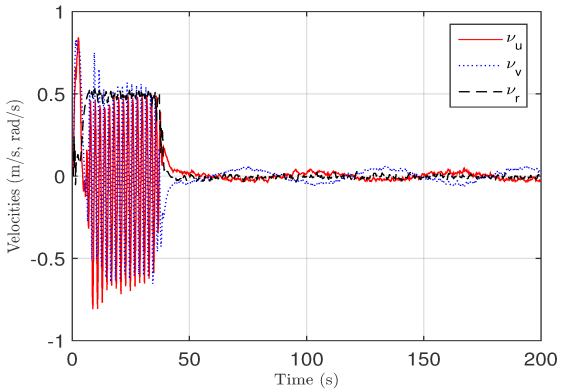
Table 4.9: Steady-state error of position and orientation system states during Γ_3 fault scenario.

System States	no FTC	proposed FTC
Position along X axis	0.0921	0.1242
Position along Y axis	0.0260	0.0103
Orientation around Z axis	0.0195	0.0088

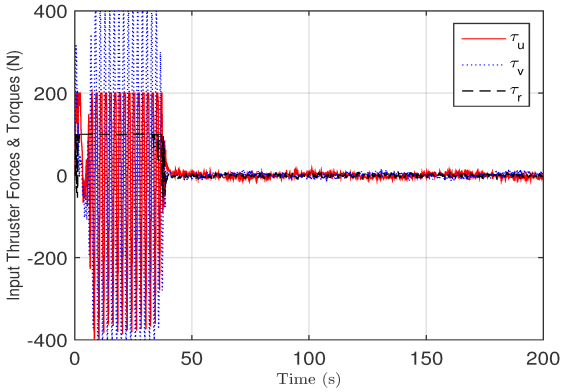
Table 4.9 illustrates the improvement that we achieved in decreasing the steady-state error in position and orientation system states by using the dual-EKF algorithm. In this scenario the AUV actuator along X and Y axes are fully working and its actuator around Z axis has 90% LOE.



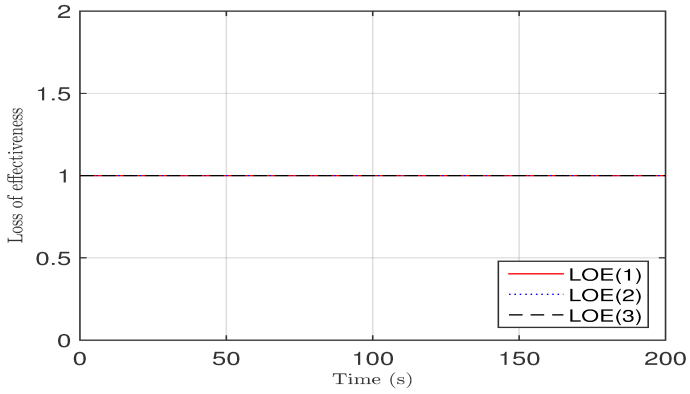
(a) Position and orientation state signals.



(b) Velocity state signals.

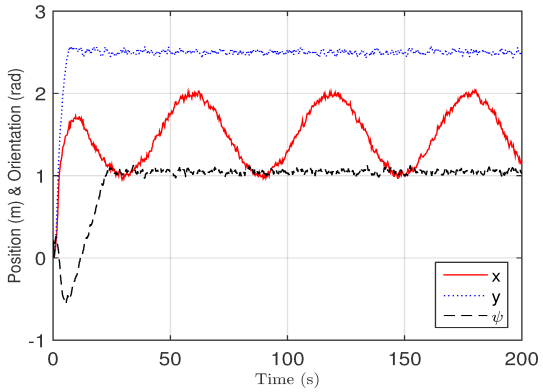


(c) Control input signals.

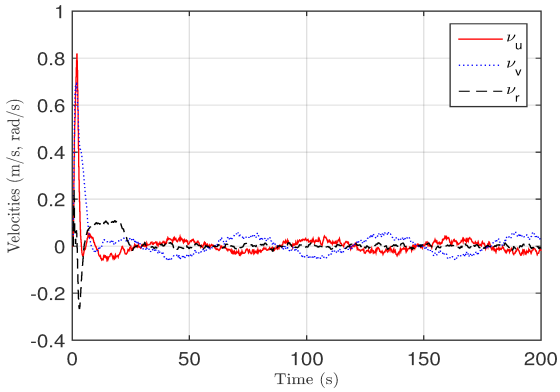


(d) Actuators' effectivness of an AUV not equipped with dual-EKF algorithm during the fault scenario Γ_2.

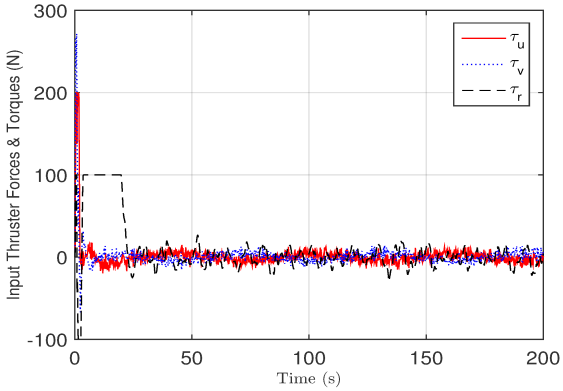
Figure 4.7: States and parameter signals of an AUV not equipped with dual-EKF algorithm when its actuators along X and Y axes are fully working and its actuator around Z axis has 90% LOE.



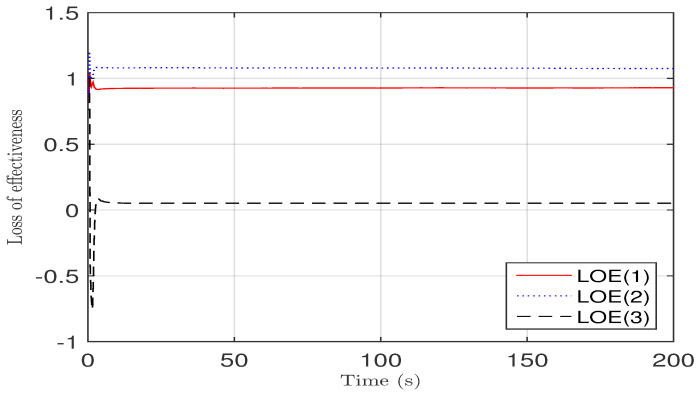
(a) Position and orientation state signals.



(b) Velocity state signals.



(c) Control input signals.



(d) Actuators' effectivness of proposed AFTC method during the fault scenario Γ_3.

Figure 4.8: States and parameter signals of an AUV equipped with our proposed AFTC algorithm when its actuators along X and Y axes are fully working and its actuator around Z axis has 90% LOE.

Table 4.10: Average Control cost of proposed AFTC schemes for 1000 sampling instants in scenario Γ_3.

Control Methods	no FTC	proposed FTC
Control cost of X	3.4230×10^3	9.1849×10
Control cost of Y	5.7103×10^3	1.4065×10^2
Control cost of N	2.1550×10^2	5.2835×10
Total Control cost	9.3488×10^3	2.8533×10^2

Consider the fault scenario depicted in the effectiveness coefficient matrix Γ_3, Table 4.10 shows that the proposed AFTC in this chapter improves the performance runtime as well as decreasing the MSE and steady-sate error while resulting in lower costs for the AUV controller in comparison with the AUV controller that does not equipped with FTC.

In this section, the first sets of experiment is conducted to demonstrate the effects of FTC on our developed control method. During the first experiment, the objective is improving the average control cost, while reducing the steady-state errors in path tracking, fault detection and control recovery of actuator faults. The objectives sought were all reached by the developed AFTC scheme of this chapter. Next set of experiment illustrate a comparison between the developed AFTC of this chapter and nonlinear MPC AFTC algorithm.

4.3.3 Second Experiment: A Comparison Between the Proposed AFTC Method and Nonlinear MPC AFTC Method

This section aims to study the comparisons between the proposed AFTC approach in this chapter with the existing FT-NMPC approach. Considering the equation (2.1a), the initial position, orientation and velocity of the AUV are set to $\eta_0 = [0, 0, \frac{\pi}{15}, 0, 0, 0]^T$ and the desired tracking mission's surge, sway and yaw are $\eta_d = [0.5\cos(t/3/\pi) + 1.5, 2.5, \pi/3, 0, 0, 0]^T$. During this mission, the AUV is faced with LOE scenarios in its actuators.

The coresponding effectiveness coefficient matrices are as follows:

- In the fourth scenario, the AUV is in its path tracking mission faced with an actuator fault. In this scenario, actuators are working with 10% of their real efficiency,

$$\Gamma_4 = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1. \end{bmatrix},$$

- In the fifth scenario, the AUV performs its path tracking mission with 75% LOE in its actuator along X axis, 50% LOE in its actuator along Y axis, and 25% LOE in its actuator around Z axis. This scenario is depicted in matrix Γ_5 with diagonal effectiveness coefficients,

$$\Gamma_5 = \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.50 & 0 \\ 0 & 0 & 0.75 \end{bmatrix}.$$

- In the sixth scenario, the AUV works with a fully effective actuator along X axis (0% LOE), 50% LOE in its actuator along Y axis, and 25% LOE in its actuator around Z

axis. This scenario is depicted in matrix Γ_6 with diagonal effectiveness coefficients,

$$\Gamma_6 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.50 & 0 \\ 0 & 0 & 0.75 \end{bmatrix}.$$

In the implementation of NMPC method, the Levenberg-Marquardt algorithm is chosen among the nonlinear least squared methods to optimize equation (2.34). The simulation results are obtained using $T = 200s$, $T_s = 0.2s$. The fault is assumed to occur sometime after $k = 50$ while the total number of signal sampling is $k = 1000$.

The performance runtime takes for FDI module to detect and isolate the fault, and performance runtime t_d are given in Table 4.11.

Table 4.11: The LOE fault in actuators in different scenarios. A comparison between the performance runtime t_d FT-NMPC and the developed MPC-RNN AFTC using dual-EKF.

Effectiveness Coefficient Matrix	FT-NMPC	Developed AFT MPC-RNN
Γ_4	$4.4629s$	$0.0152s$
Γ_5	$4.8544s$	$0.0153s$
Γ_6	$4.2118s$	$0.0167s$

Table 4.11 shows that the proposed AFTC method in this chapter improved the performance runtime of the fault detection and recovery in comparison with the FT-NMPC scheme, significantly.

Fourth Scenario

Fourth scenario studies the performance of the AUV in its path tracking mission, discussed in Section 4.3.3, faced with actuator fault while equipped with FT-NMPC and the developed AFTC scheme of this chapter. In this fault scenario actuators are working with 10% of their

real efficiency,

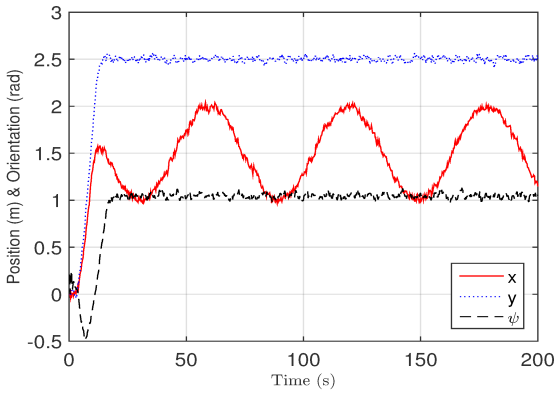
$$\Gamma_4 = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}.$$

The main objectives of this mission are discussed in Section 4.3.1. Figures 4.9 to 4.12 illustrate the results and estimation errors of simulations during fault scenarios which their effectiveness coefficient matrices are Γ_4. Note that in this path following mission, the aim is to just control the position and orientation states. the states regarding linear and rotational velocities are given just as information.

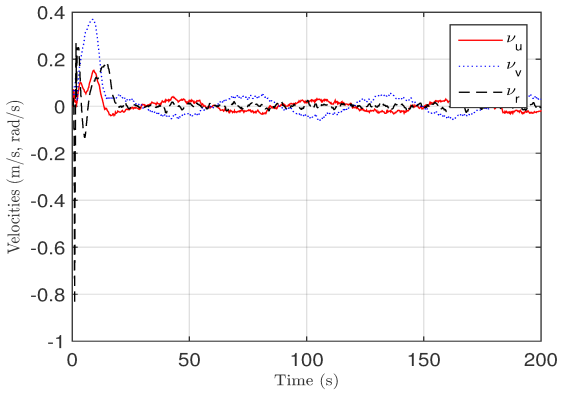
Table 4.12: MSE for 1000 sampling instants of positions and orientation system state during Γ_4 fault scenario. A comparison between the MSE of the FT-NMPC and the proposed MPC-RNN AFTC using dual-EKF.

System States	FT-NMPC	Developed AFT MPC-RNN
Position along X axis	5.1556×10^{-4}	5.5197×10^{-4}
Position along Y axis	5.2340×10^{-4}	5.4294×10^{-4}
Orientation around Z axis	6.8721×10^{-4}	6.3048×10^{-4}

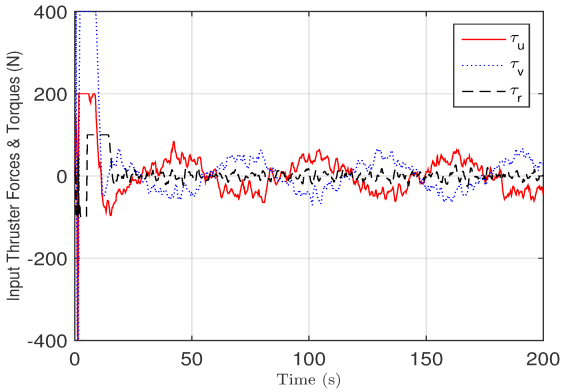
Table 4.12 shows that during the fourth scenario which its effectiveness coefficients are illustrated in the matrix Γ_4, our proposed AFTC method has, approximately, the same MSE using both AFTC methods.



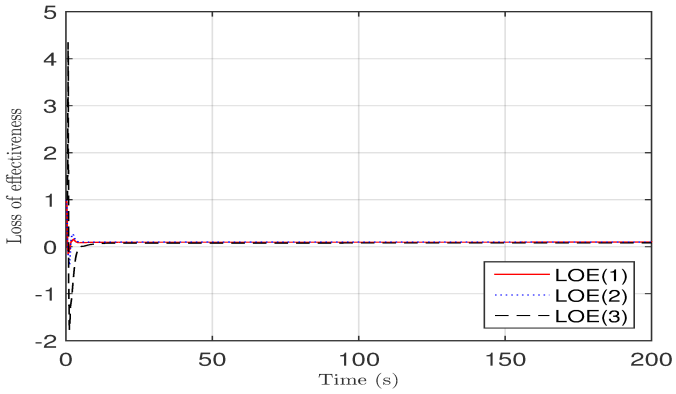
(a) Position and orientation state signals.



(b) Velocity state signals.

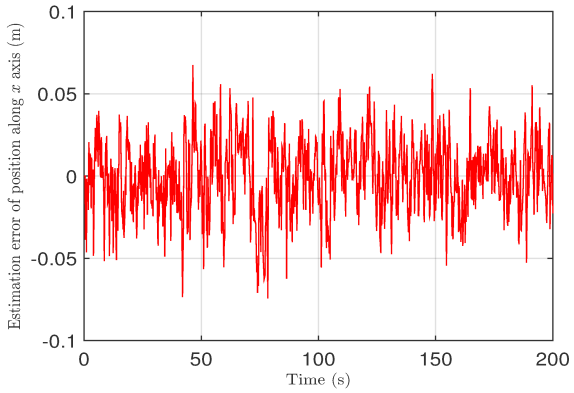


(c) Control input signals.

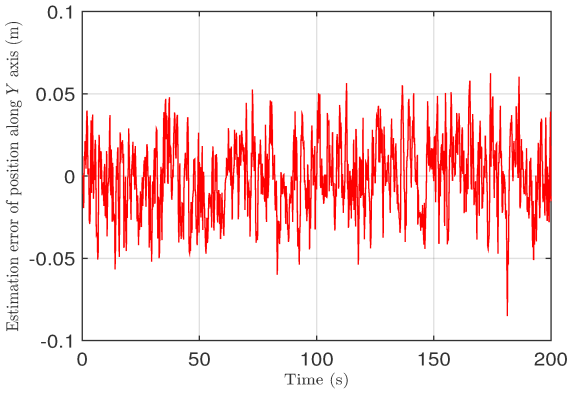


(d) Actuators' effectivness of FT-NMPC method during the fault scenario Γ_4.

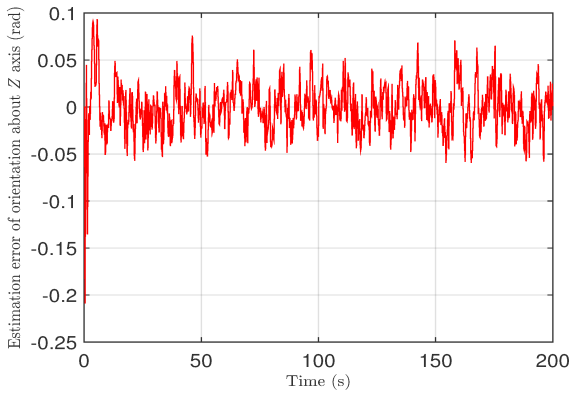
Figure 4.9: States and parameter signals of the FT-NMPC scheme applied on an AUV with 90% LOE in each of its actuators.



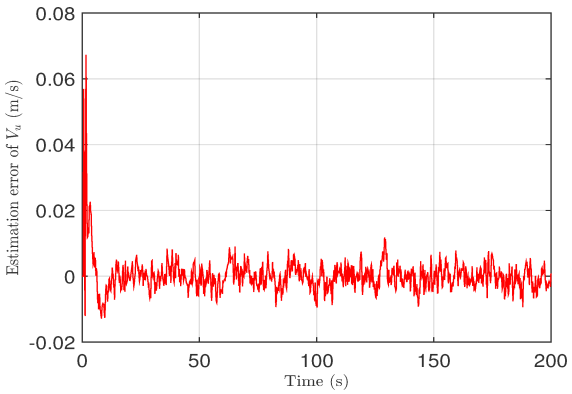
(a) Position state estimation error signal along X axis.



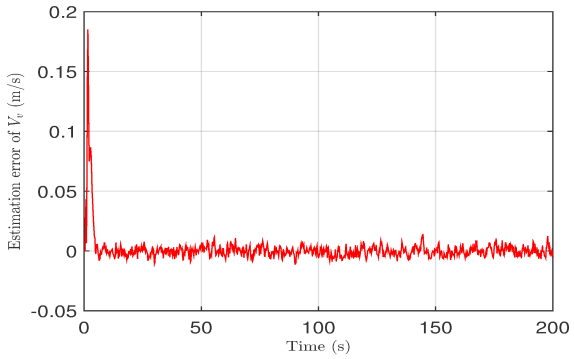
(b) Position state estimation error signal along Y axis.



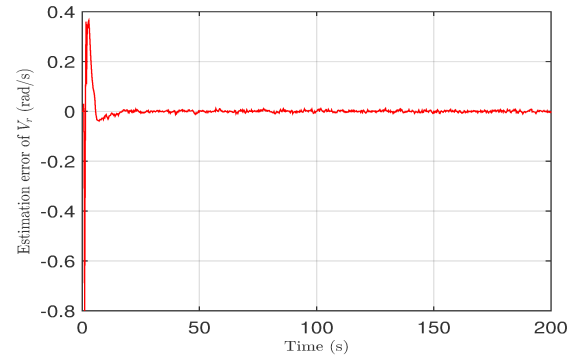
(c) Orientation state estimation error signal around Z axis.



(d) Velocity state estimation error signal along X axis.

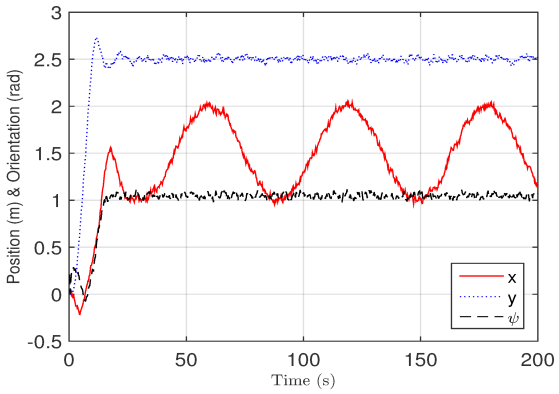


(e) Velocity state estimation error signal along Y axis.

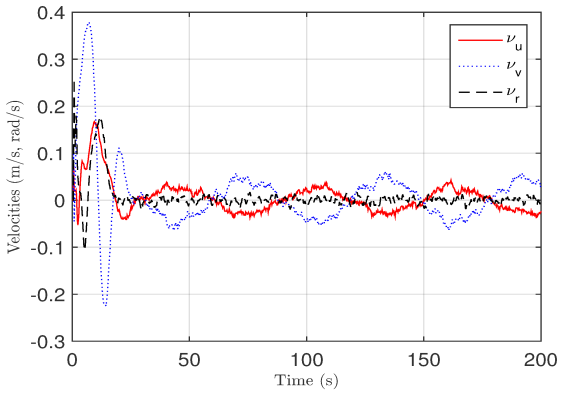


(f) Rotational state estimation error signal arond Z axis.

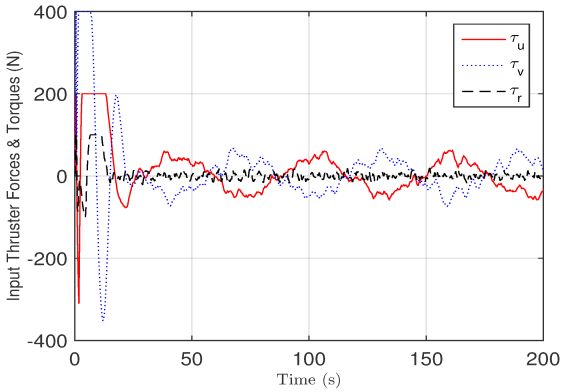
Figure 4.10: States and parameter estimation error signals of the FT-NMPC scheme applied on an AUV with 90% LOE in each of its actuators.



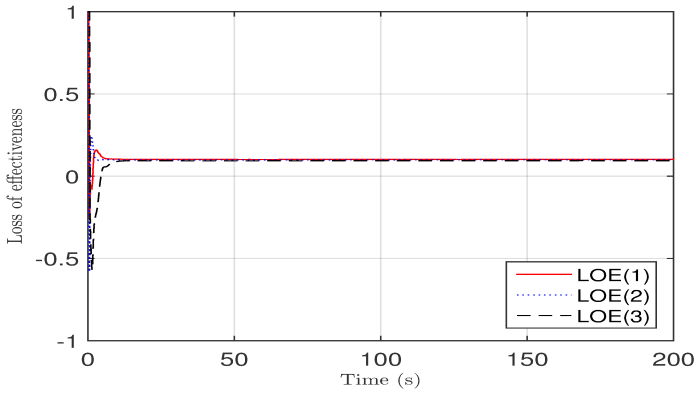
(a) Position and orientation state signals.



(b) Velocity state signals.

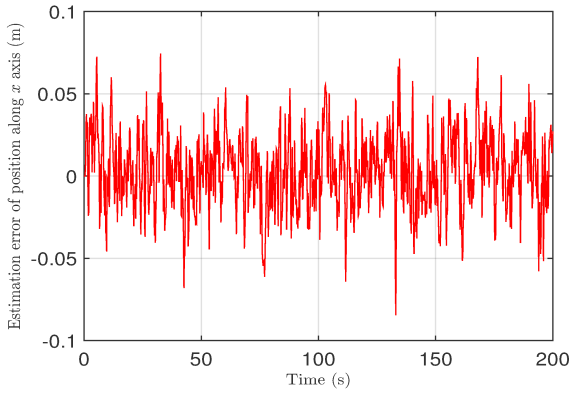


(c) Control input signals.

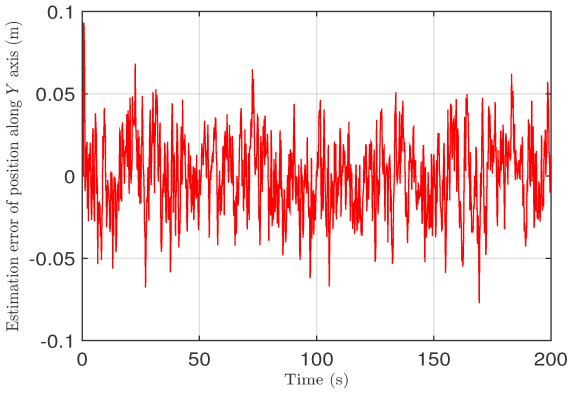


(d) Actuators' effectivness of our proposed AFTC method during the fault scenario Γ_4.

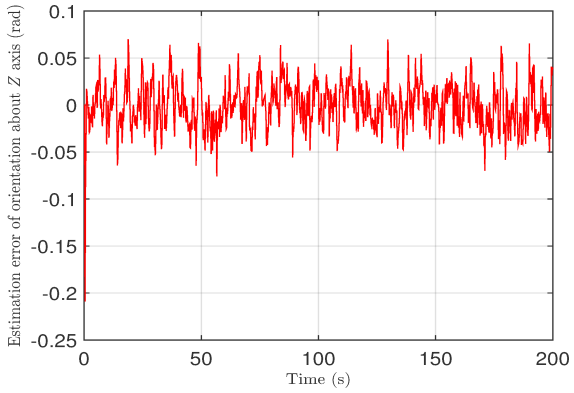
Figure 4.11: States and parameter signals of the proposed AFTC scheme applied on an AUV with 90% LOE in each of its actuators.



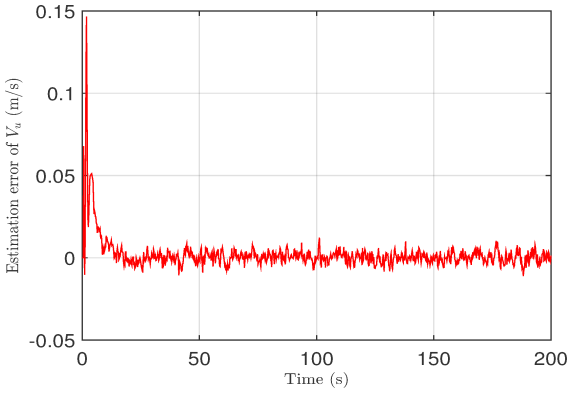
(a) Position state estimation error signal along X axis.



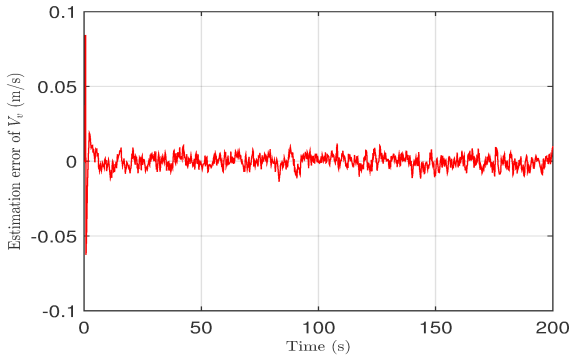
(b) Position state estimation error signal along Y axis.



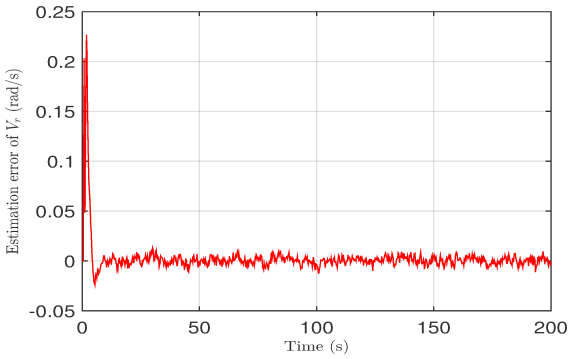
(c) Orientation state estimation error signal around Z axis.



(d) Velocity state estimation error signal along X axis.



(e) Velocity state estimation error signal along Y axis.



(f) Rotational state estimation error signal arond Z axis.

Figure 4.12: States and parameter estimation error signals of our proposed AFTC scheme applied on an AUV with 90% LOE in each of its actuators.

Table 4.13: Steady-state error for each system state for the AUV faced with the Γ_4 fault scenario. A comparison between the steady-state error of the FT-NMPC and the proposed MPC-RNN AFTC using dual-EKF.

System States	FT-NMPC	Developed AFT MPC-RNN
Position along X axis	0.1181	0.0973
Position along Y axis	0.0108	0.0054
Orientation around Z axis	0.0285	0.0019
Parameter w_{loe1}	0.0016	0.0046
Parameter w_{loe2}	0.0005	0.0005
Parameter w_{loe3}	0.0199	0.0077

Table 4.13 shows that during the fourth scenario which its effectiveness coefficients are illustrated in the matrix Γ_4, our proposed AFTC method has slightly lower steady-states error, except the states of system regarding the orientation around Z axis and the first parameter of system w_{loe1}. Note that in this path following mission, the goal is to just control the position and orientation states.

Table 4.14: Average Control cost during 1000 sampling instants in scenario Γ_4.

LOE Scenarios	FT-NMPC	Developed AFT MPC-RNN
Control cost of X	7.1142×10^3	1.4177×10^2
Control cost of Y	4.9571×10^2	2.7730×10^2
Control cost of N	1.9809×10^3	5.4051×10
Total Control cost	9.5908×10^3	4.73121×10^2

Table 4.14 shows that during the fourth scenario which its effectiveness coefficients are illustrated in the matrix Γ_4, the average control cost of our proposed AFTC method is appreciably lower than the average control cost during the experiment using FT-NMPC

method.

Fifth Scenario

Fifth scenario studies the performance of the AUV in its path tracking mission, discussed in Section 4.3.3, faced with actuator fault while equipped with FT-NMPC and the developed AFTC scheme of this chapter. In this fault scenario, the actuator along X and Y axes work with 25% and 50% of their efficiency, respectively. Also, The AUV's actuator around Z axis works with 75% of its efficiency,

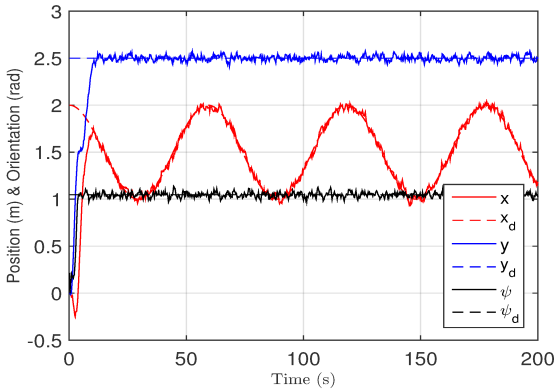
$$\Gamma_5 = \begin{bmatrix} 0.250 & 0 & 0 \\ 0 & 0.50 & 0 \\ 0 & 0 & 0.75 \end{bmatrix}.$$

The main objectives of this mission are discussed in Section 4.3.1. Figures 4.13 to 4.16 illustrates the results and estimation errors of simulations during fault scenarios which their effectiveness coefficient matrices are Γ_5. Note that in this path following mission, the goal is to just control the position and orientation states. the states regarding linear and rotational velocities are given just for information purposes.

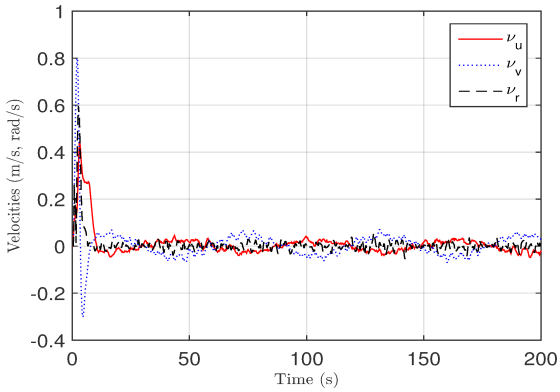
Table 4.15: MSE for 1000 sampling instants of each system state during Γ_5 fault scenario. A comparison between the MSE of the FT-NMPC and our proposed MPC-RNN AFTC using dual-EKF.

System States	FT-NMPC	Developed AFT MPC-RNN
Position along X axis	6.2475×10^{-4}	5.5315×10^{-4}
Position along Y axis	5.2572×10^{-4}	4.9474×10^{-4}
Orientation around Z axis	6.0255×10^{-4}	6.3894×10^{-4}

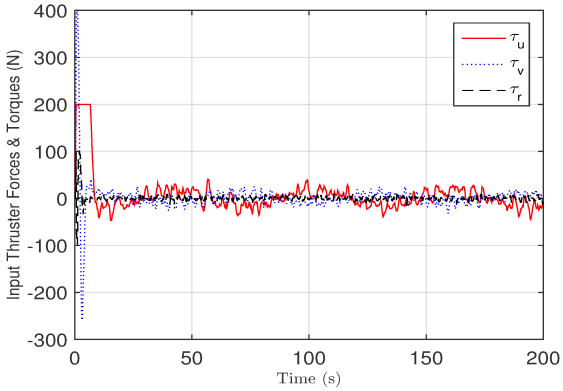
Table 4.15 shows that during the fifth scenario which its effectiveness coefficients are illustrated in the matrix Γ_5, our proposed AFTC method has lower MSE except for the



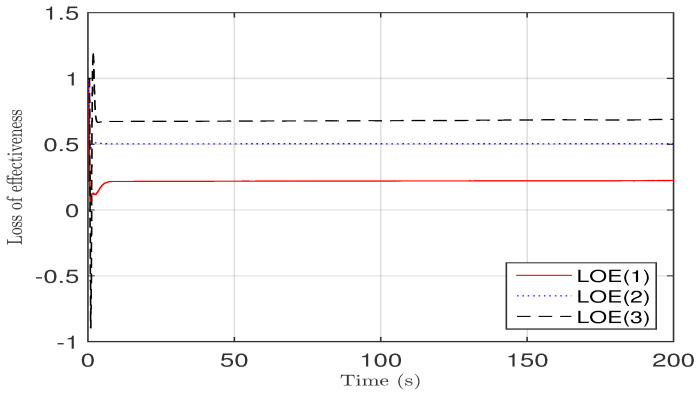
(a) Position and orientation state signals.



(b) Velocity state signals.

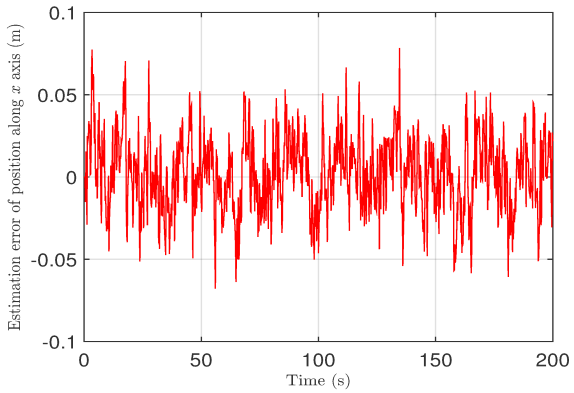


(c) Control input signals.

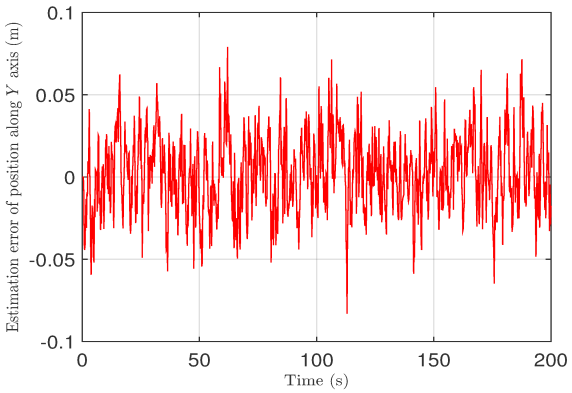


(d) Actuators' effectivness of FT-NMPC method during the fault scenario Γ_5.

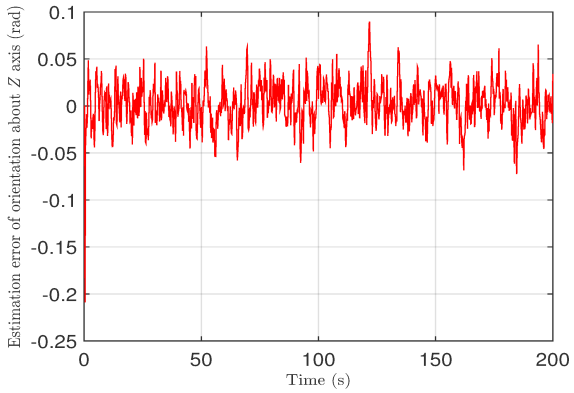
Figure 4.13: States and parameter signals of the FT-NMPC scheme applied on an AUV with 75% LOE in its actuator along X axis, 50% LOE in its actuator along Y axis, and 25% LOE in its actuator around Z axis.



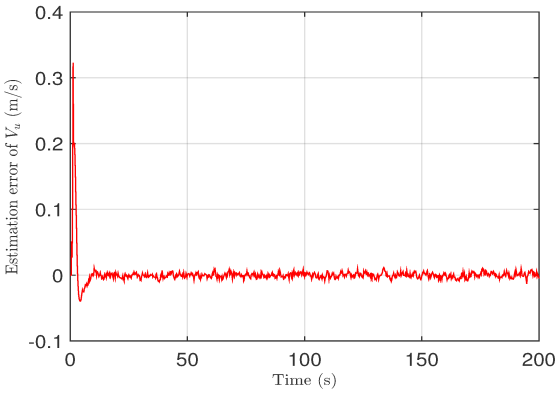
(a) Position state estimation error signal along X axis.



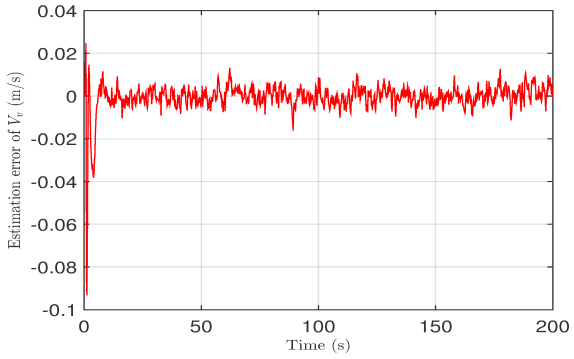
(b) Position state estimation error signal along Y axis.



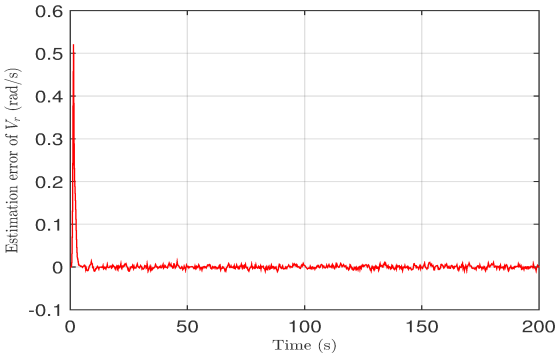
(c) Orientation state estimation error signal around Z axis.



(d) Velocity state estimation error signal along X axis.

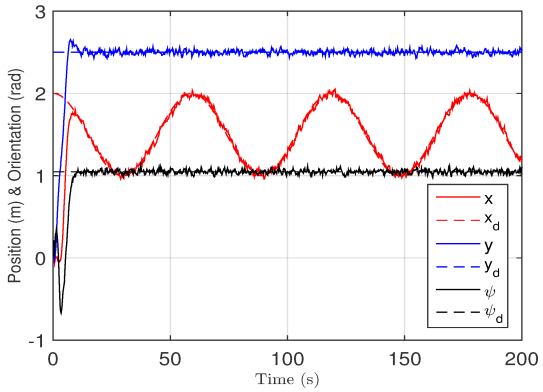


(e) Velocity state estimation error signal along Y axis.

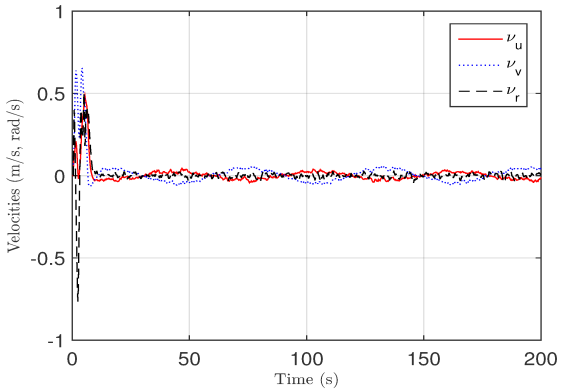


(f) Rotational state estimation error signal arond Z axis.

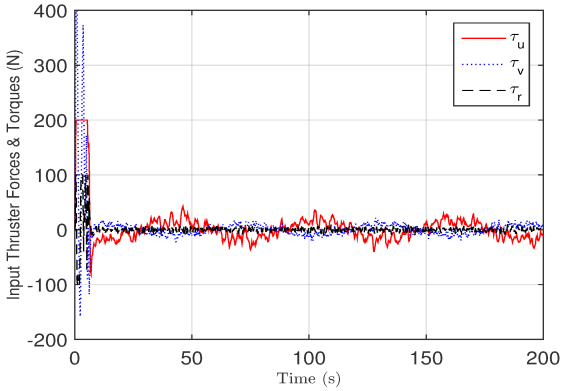
Figure 4.14: States and parameter estimation error signals of the FT-NMPC scheme applied on an AUV with 75% LOE in its actuator along X axis, 50% LOE in its actuator along Y axis, and 25% LOE in its actuator about Z axis.



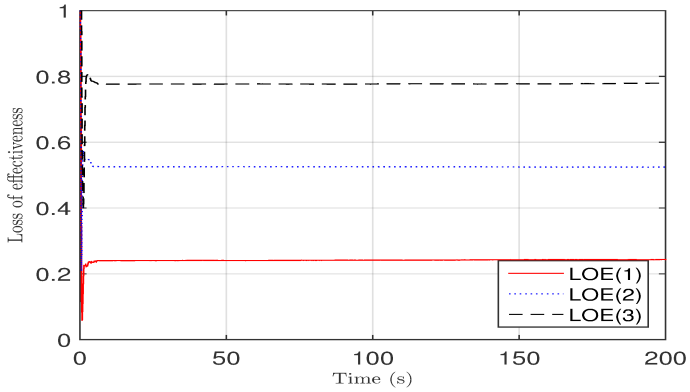
(a) Position and orientation state signals.



(b) Velocity state signals.

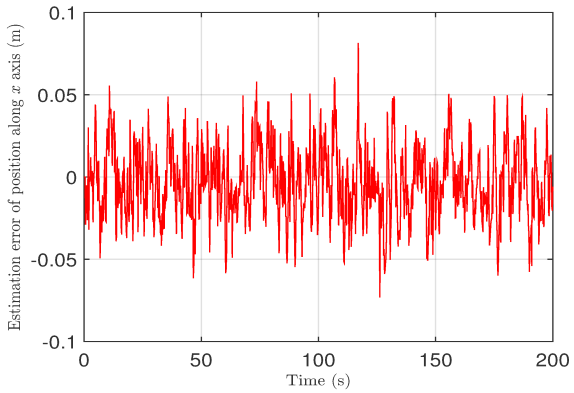


(c) Control input signals.

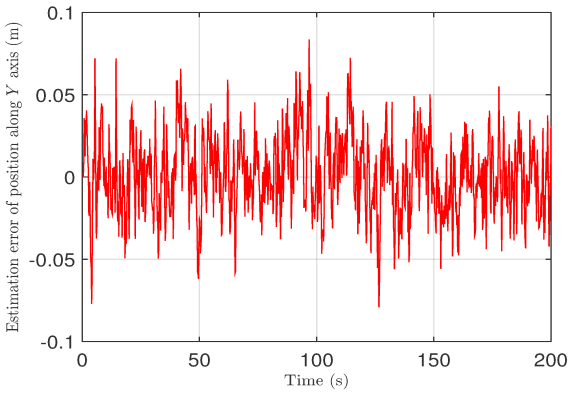


(d) Actuators' effectivness of our proposed AFTC method during the fault scenario Γ_5.

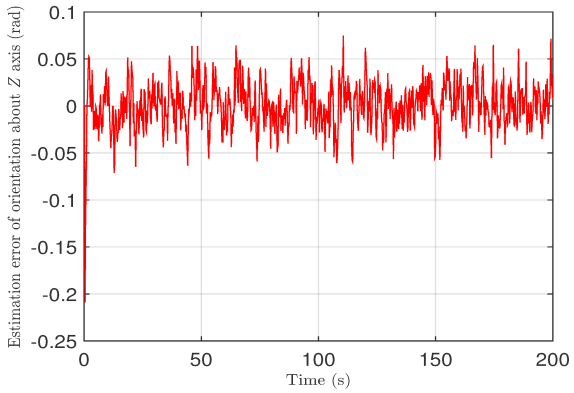
Figure 4.15: States and parameter signals of the proposed AFTC scheme applied on an AUV with 75% LOE in its actuator along X axis, 50% LOE in its actuator along Y axis, and 25% LOE in its actuator around Z axis.



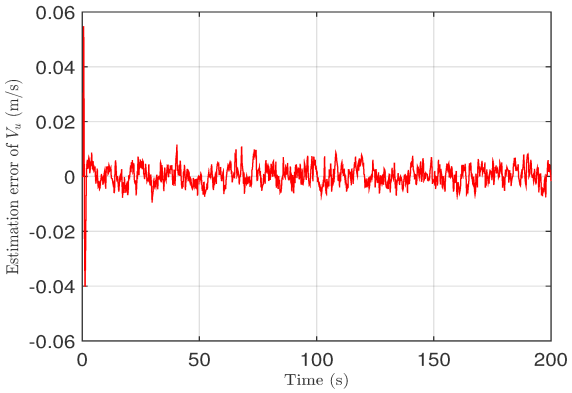
(a) Position state estimation error signal along X axis.



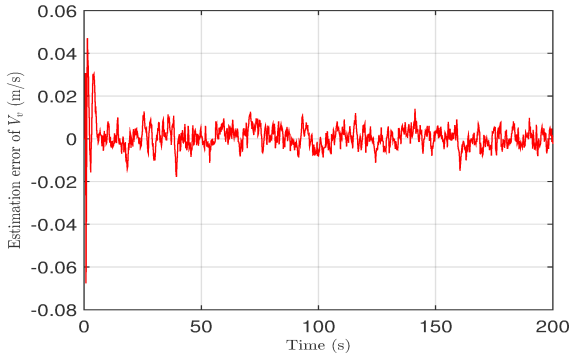
(b) Position state estimation error signal along Y axis.



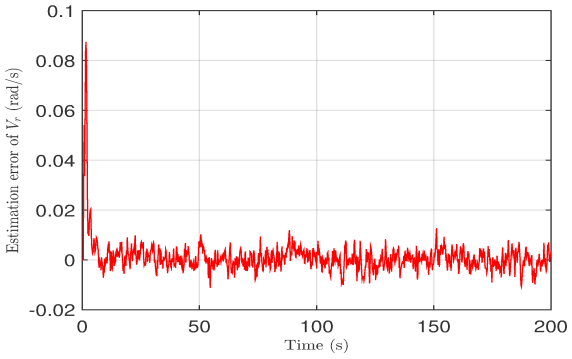
(c) Orientation state estimation error signal around Z axis.



(d) Velocity state estimation error signal along X axis.



(e) Velocity state estimation error signal along Y axis.



(f) Rotational state estimation error signal arond Z axis.

Figure 4.16: States and parameter estimation error signals of our proposed AFTC scheme applied on an AUV with 75% LOE in its actuator along X axis, 50% LOE in its actuator along Y axis, and 25% LOE in its actuator around Z axis.

states of system regarding the orientation around Z axis.

Table 4.16: Steady-state error for each system state during Γ_5 fault scenario. A comparison between the steady-state error of the FT-NMPC and the proposed MPC-RNN AFTC using dual-EKF.

System States	FT-NMPC	Developed AFT MPC-RNN
Position along X axis	0.0606	0.0902
Position along Y axis	0.0148	0.0149
Orientation around Z axis	0.0281	0.0108
Parameter w_{loe1}	0.6236	0.0047
Parameter w_{loe2}	0.1476	0.0406
Parameter w_{loe3}	0.0723	0.0186

Table 4.16 shows that during the fifth scenario which its effectiveness coefficients are illustrated in the matrix Γ_5, our proposed AFTC method has lower steady-states error except for the states of system regarding the position along X axis, position along Y axis. Since accuracy in fault severity detection is one of our objectives, this is worth mentioning that the measured values for parameters are shown to be estimated more accurate with our proposed AFTC method than using the FT-NMPC method.

Table 4.17: Average Control cost during 1000 sampling instants in scenario Γ_5.

LOE Scenarios	FT-NMPC	Developed AFT MPC-RNN
Control cost of X	2.6494×10^3	8.6731×10
Control cost of Y	2.1348×10^2	3.2184×10^2
Control cost of N	8.4657×10	1.0878×10^2
Total Control cost	2.9475×10^3	5.1735×10^2

Table 4.17 shows that during the fifth scenario which its effectiveness coefficients are

illustrated in the matrix Γ_5, the average control cost of our proposed AFTC method is lower than the average control cost during the experiment using FT-NMPC method.

Sixth Scenario

Sixth scenario studies the performance of the AUV in its path tracking mission, discussed in Section 4.3.3, faced with actuator fault while equipped with FT-NMPC and the developed AFTC scheme of this chapter. In this fault scenario the AUV's actuator along X axis is fully effective (0% LOE), the actuator along Y and around Z axes work with 50% and 25% of their efficiency, respectively,

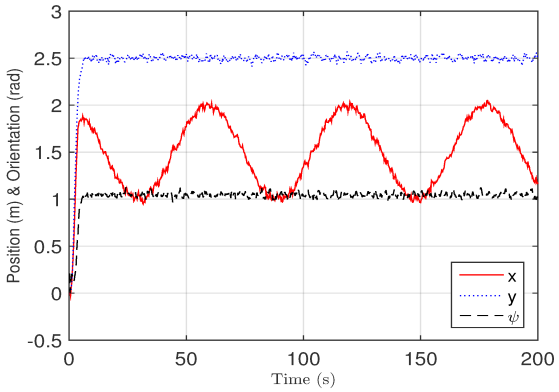
$$\Gamma_6 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.50 & 0 \\ 0 & 0 & 0.75 \end{bmatrix}$$

The main objectives of this mission are discussed in Section 4.3.1. Figures 4.17 to 4.20 illustrate the results and estimation errors of simulations during fault scenarios which their effectiveness coefficient matrices are Γ_6. Note that in this path following mission, the goal is to just control the position and orientation states. the states regarding linear and rotational velocities are given just for information purposes.

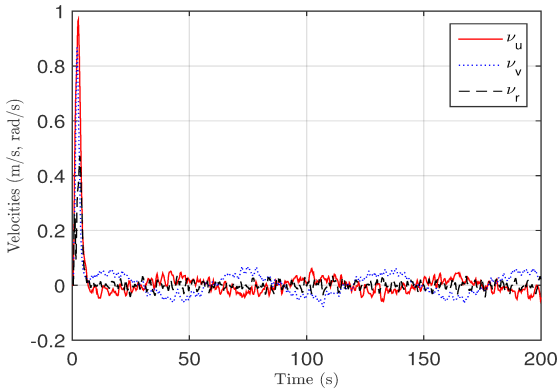
Table 4.18: MSE for 1000 sampling instants of each system state during Γ_6 fault scenario. A comparison between the MSE of the FT-NMPC and the proposed MPC-RNN AFTC using dual-EKF.

System States	FT-NMPC	Developed AFT MPC-RNN
Position along X axis	5.1326×10^{-4}	5.5298×10^{-4}
Position along Y axis	5.0431×10^{-4}	5.6239×10^{-4}
Orientation around Z axis	7.0810×10^{-4}	6.0469×10^{-4}

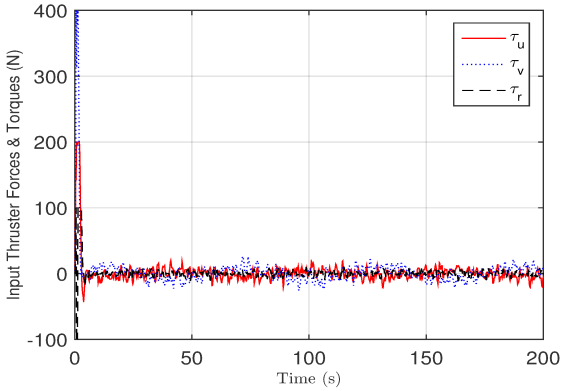
Table 4.18 shows that during the sixth scenario which its effectiveness coefficients are



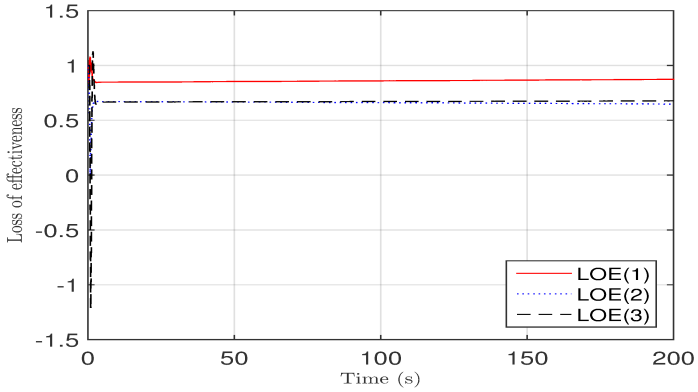
(a) Position and orientation state signals.



(b) Velocity state signals.

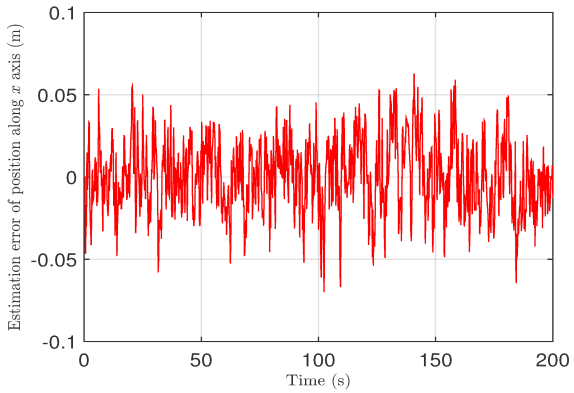


(c) Control input signals.

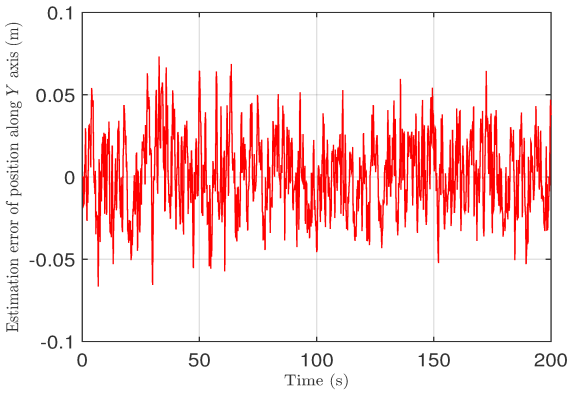


(d) Actuators' effectivness of FT-NMPC method during the fault scenario Γ_6.

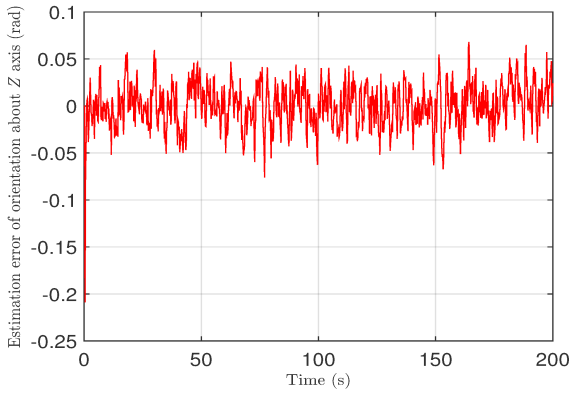
Figure 4.17: States and parameter signals of FT-NMPC scheme applied on an AUV with a fully effective actuator along X axis, 50% LOE in its actuator along Y axis, and 25% LOE in its actuator around Z axis.



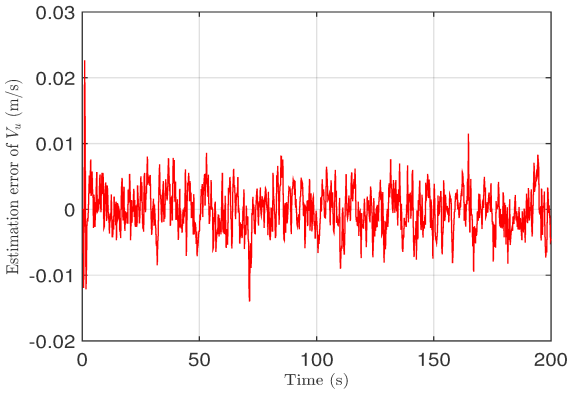
(a) Position state estimation error signal along X axis.



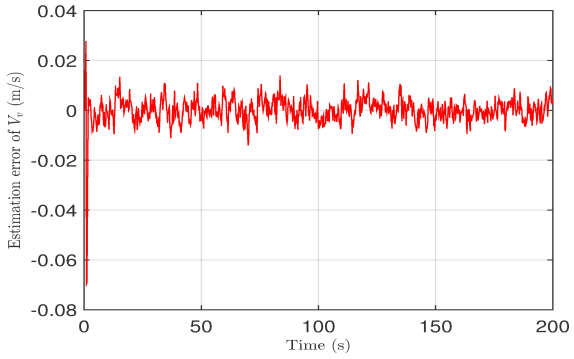
(b) Position state estimation error signal along Y axis.



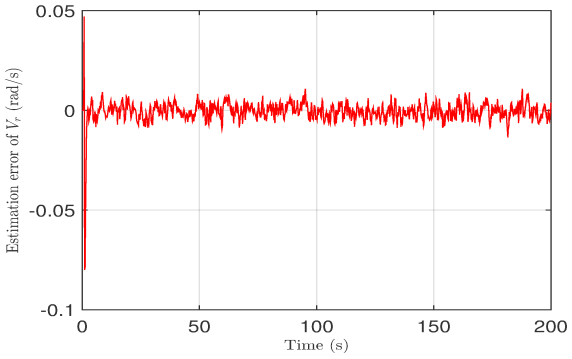
(c) Orientation state estimation error signal around Z axis.



(d) Velocity state estimation error signal along X axis.

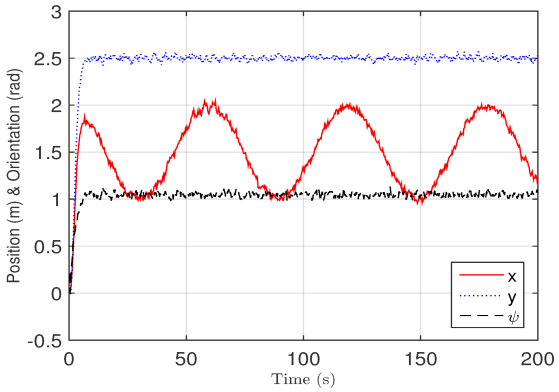


(e) Velocity state estimation error signal along Y axis.

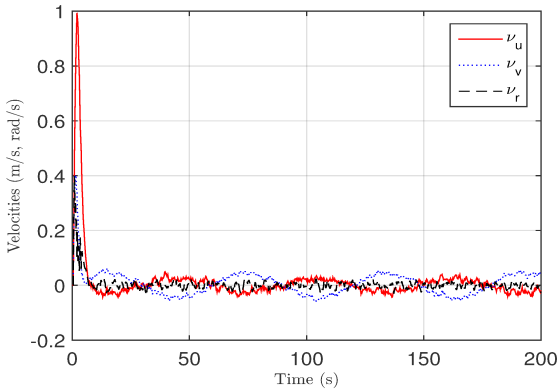


(f) Rotational state estimation error signal arond Z axis.

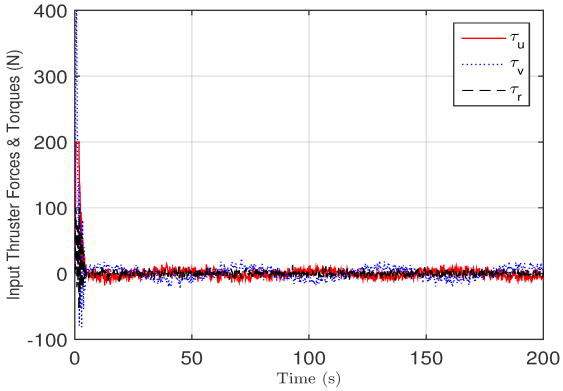
Figure 4.18: States and parameter estimation error signals of FT-NMPC scheme applied on an AUV with a fully effective actuator along X axis, 50% LOE in its actuator along Y axis, and 25% LOE in its actuator about Z axis.



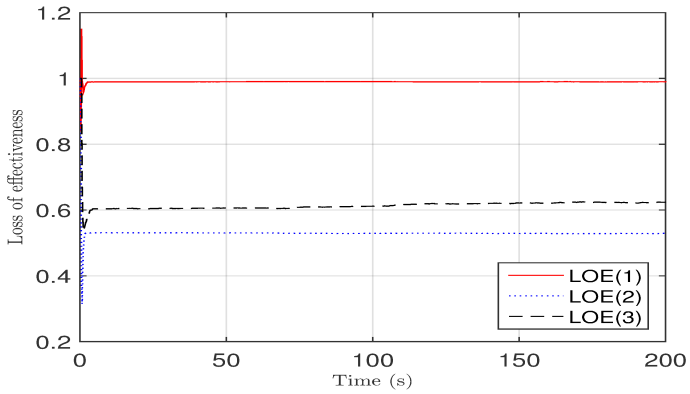
(a) Position and orientation state signals.



(b) Velocity state signals.

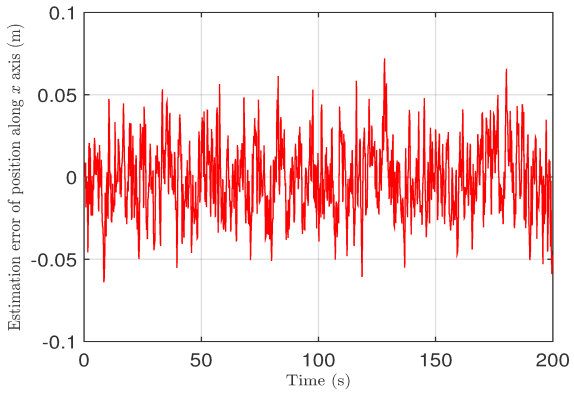


(c) Control input signals.

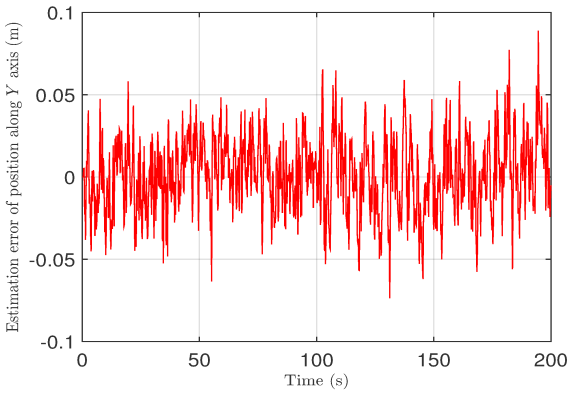


(d) Actuators' effectivness of our proposed AFTC method during the fault scenario Γ_6.

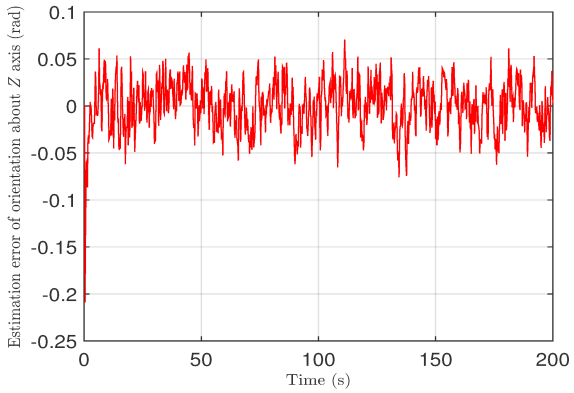
Figure 4.19: States and parameter signals of the proposed AFTC scheme applied on an AUV with a fully effective actuator along X axis, 50% LOE in its actuator along Y axis, and 25% LOE in its actuator about Z axis.



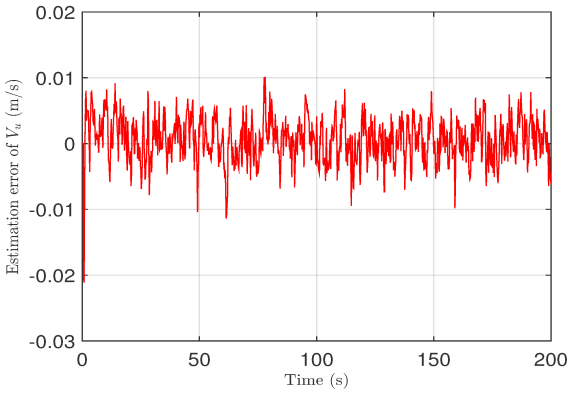
(a) Position state estimation error signal along X axis.



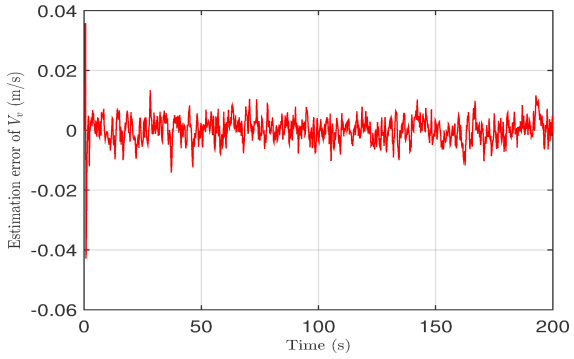
(b) Position state estimation error signal along Y axis.



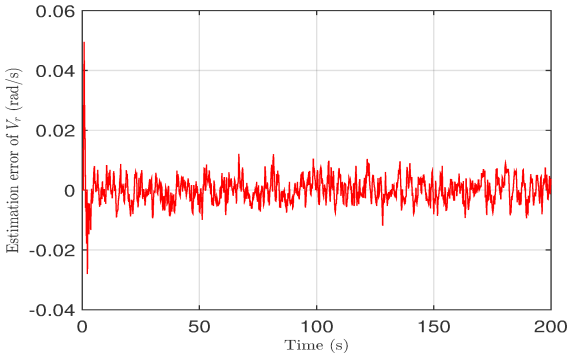
(c) Orientation state estimation error signal around Z axis.



(d) Velocity state estimation error signal along X axis.



(e) Velocity state estimation error signal along Y axis.



(f) Rotational state estimation error signal arond Z axis.

Figure 4.20: States and parameter estimation error signals of our proposed AFTC scheme applied on an AUV with a fully effective actuator along X axis, 50% LOE in its actuator along Y axis, and 25% LOE in its actuator around Z axis.

illustrated in the matrix Γ_6, our proposed AFTC method has approximately the same MSE as the nonlinear MPC AFTC method.

Table 4.19: Steady-state error for each system state during Γ_6 fault scenario. A comparison between the steady-state error of the FT-NMPC and our proposed MPC-RNN AFTC using dual-EKF.

System States	FT-NMPC	Developed AFT MPC-RNN
Position along X axis	0.1368	0.1167
Position along Y axis	0.0137	0.0151
Orientation around Z axis	0.0031	0.0072
Parameter w_{loe1}	0.1351	0.0065
Parameter w_{loe2}	0.2023	0.0887
Parameter w_{loe3}	0.0604	0.018

Table 4.19 shows that during the sixth scenario which its effectiveness coefficients are illustrated in the matrix Γ_6, our proposed AFTC method has better estimations of the parameters of the system, which are the actuator effectivenesses.

Table 4.20: Average Control cost during 1000 sampling instants in scenario Γ_6.

LOE Scenarios	FT-NMPC	Developed AFT MPC-RNN
Control cost of X	3.7008×10^2	7.6973×10
Control cost of Y	6.2001×10^2	2.4056×10^2
Control cost of N	4.7591×10	1.4476×10^2
Total Control cost	1.0376×10^3	4.6229×10^2

Table 4.20 shows that during the sixth scenario which its effectiveness coefficients are illustrated in the matrix Γ_6, the average control cost of our proposed AFTC method is lower than the average control cost during the experiment using FT-NMPC method.

The second experiment series illustrates reaching to the objectives of developing the AFTC method of this chapter. Our proposed AFTC method in this thesis shown to have approximately the same accuracy of FT-NMPC method while it reduced the average control cost and performance runtime of the controller. Also, our proposed method is more accurate in estimating the actuator faults which are the parameters of the system, in comparison with FT-NMPC algorithm. Next section brings more discussions on the results of all sixth series of simulations illustrated in this chapter.

4.4 Discussion

In this chapter using the proposed control method from Chapter 3, a dual-EKF algorithm was integrated to the controller. The proposed AFTC scheme in this chapter allows for designing a control solution that takes into account the system kinematics and meets uniform asymptotic convergence requirements using RNN. Several simulations are given in order to evaluate the developed AFTC method of this chapter. First the effects of applying FTC is illustrated in 3 different scenarios. Then, a comparison between the developed method of this chapter and FT-NMPC method is done through 3 more additional scenarios. The Results from the graphs and tables in each experiment are discussed below:

- First experiment, given in Section 4.3.2, presented the privileges of integration of the dual-EKF algorithm with the proposed control method from Chapter 3 to illustrate the satisfactory performance of the developed controller when severe faults occur in the actuators.

- As shown in the first experiment, several severe actuator fault scenarios are designed and shows that the proposed AFTC in this chapter is slightly enhancing the performance runtime as well as decreasing the MSE and steady-sate error while resulting in, appreciably, lower costs for the AUV controller in comparison with the AUV controller that does not equipped with FTC.

- As a comparison, the second experiment, given in Section 4.3.3, presented the privileges of the developed AFTC method of this chapter over the FT-NMPC method with regards to the objectives of this chapter.

- The second experiment illustrates the accomplishment of the objectives of developing the AFTC method of this chapter. The proposed AFTC method in this thesis shown to have approximately the same accuracy of FT-NMPC method while it reduces the

average control cost and performance runtime of the controller. Also, our proposed method is more accurate in estimating the actuator faults which are defined the parameters of the system, in comparison with FT-NMPC algorithm. The estimation of the fault severity is another main objective of this chapter.

4.5 Conclusion

In this chapter, the problem of adaptive AFTC for a class of nonlinear systems with actuator fault is investigated. It is stated that the problem statement of this chapter arises when the input is not accurate and requires the coupling both state estimation and parameter estimation. Therefore, using the proposed control method from Chapter 3, a dual-EKF algorithm was integrated to the controller. Through a number of scenarios the effects of FTC on the system have been studied. Moreover, a series of analysis given as a comparison between our developed AFTC method and a conventional FT-NMPC method. Finally, considering model nonlinearities and our designed mission objectives, the advantages of our proposed AFTC approach are discussed. Next chapter gives the conclusions and probable future works of this thesis.

Chapter 5

Conclusion and Future Works

In this thesis we have explored the problem of trajectory tracking and path following control of autonomous underwater vehicle (AUV) systems. The purpose of this work is to improve the model-based control of highly nonlinear systems with uncertainties while incorporating system constraints, and to reduce the high performance runtime cost of solving QP problem in Nonlinear Model Predictive Control (MPC) -based algorithms. This was to be achieved by integrating the control scheme with a Recurrent Neural Network (RNN). The second goal of this work is to develop a Fault-Tolerant Control (FTC) algorithm of an AUV system with uncertainties, to include some alternative techniques and methods based on hybrid of MPC and RNN, so that the AUV follows the desired trajectory while meeting a set of requirements and bounds on position, orientation, linear and rotational velocity, and actuator efforts. The requirements which are set to be achieved by the developed control scheme in the AUV's mission, aim to minimize the control cost along with the performance runtime of the system while the AUV's performance in path tracking remains satisfactory. Also, as another objective, this research seeks a fault detection-recovery task to overcome the loss-of-effectiveness (LOE) faults as well as to improve the system's recognition of fault severity in the actuators of the AUV system.

Toward aforementioned goals, the performance-runtime efficient approach proposed in this thesis falls into a hybrid of MPC and RNN control method to benefit from both nonlinear mathematical model information of the system and the adaptation capability of RNN. Due to the advantages of Extended Kalman filter (EKF) and the objectives of our problem statement, EKF is selected for state estimation of the AUV nonlinear system to provide MPC formulation as well as the nonlinear prediction. Since, the algorithm requires solving an online quadratic programming problem, an RNN is employed to guarantee obtaining the optimal solution of the model predictive control in each sampling time. The main feature of the overall developed controller is that due to the use of MPC, it can explicitly consider constraints on control inputs, and achieve acceptable path tracking performance. To evaluate the performance of the developed control method, 3 comparative control methods are given, namely linear MPC using Kalman filter (KF), NMPC using EKF and the developed MPC-RNN using EKF. The developed method of this work has reached to the pre-defined path tracking goal faster with lower Tracking Error Cost values in the position and orientation states, than the NMPC and Linear MPC methods. Also, the developed MPC-RNN control method has performed with least values of Mean Square Error (MSE) for tracking positions and orientation states among the above mentioned comparative methods. In comparison with conventional NMPC algorithm results on steady-state tracking error, our proposed control method performs very close to NMPC.

Although, it is shown that improvement in the performance runtime (in comparison with NMPC method) as well as the improvement in accurate trajectory following and path tracking (in comparison with linear MPC method) of our developed control method, comes along with an increase in the control cost for the system, but the developed MPC-RNN, overall has performed better when faced with MPC design tunning circumstances. Hence, considering our real-time application and its objective that is trajectory following and path tracking in horizontal plan, and confirmed from the simulations and analysis, choosing the

developed method of this chapter to control is a feasible option to be applied to industrial applications.

To accomplish the second goal of this work, an Active Fault-Tolerant Control (AFTC) scheme is developed by integrating the developed hybrid MPC and RNN controller with dual Extended Kalman Filter (dual-EKF). The developed AFTC method of this chapter adopted the actuator fault model in the nonlinear AUV model defined as the system parameters that are obtained form the dual-EKF. Hence, in the mentions active fault-tolerant control system faults are detected and identified by a fault detection identification scheme, and the controllers are reconfigured accordingly, online in a single frame. Several simulations are given in order to evaluate the developed AFTC method of this chapter. Conclusions and analysis of the simulation results show the advantages of integration of the dual-EKF algorithm with the proposed control method of this research by demonstrating the satisfactory performance of the developed fault-tolerant controller when severe faults occur in the AUV's actuators. The proposed AFTC is slightly enhancing the performance runtime as well as decreasing the MSE and steady-sate errors while resulting in, appreciably, lower costs for the AUV controller in comparison with the AUV controller that is not equipped with FT scheme.

The developed AFTC method based on integrating MPC and RNN, demonstrated approximately the same accuracy of Fault-Tolerant Nonlinear Model predictive Control (FT-NMPC) scheme while the developed method reduces the average control cost and performance runtime of the AUV controller during a trajectory following mission faced with LOE actuator faults. The proposed AFTC method is more accurate in estimating the actuator faults which are defined as the parameters of the system, in comparison with FT-NMPC algorithm. Therefore, the proposed AFTC scheme in this work allows for designing a control solution that reaches our stated problem goals as well as taking into account the system dynamics and meets convergence requirements using RNN while providing an automated system recovery scheme when subjected to common actuator faults

Based on the conclusions obtained in this study, the recommended future works can be listed as

- To extend the functionality of the AUV for wider range of underwater missions, considering the 6-DOF nonlinear model of AUV can enhance the trajectory tracking and path following precision to a great deal.

- The proposed fault tolerant scheme can be developed to address sensor faults as well as other types of actuator faults.

- Fault-Tolerant Control of multi-agent systems are more challenging as compared with the single agent system studied in this thesis. Therefore, as an extension to this work, one can develop the proposed fault-tolerant control of this thesis for a team of dynamically identical or heterogeneous agents.

- Proactive fault-tolerant control defined as another approach to FTC in the communication systems and aerospace control system communities. At this point, no work has been done on proactive fault-tolerant control within the context of AUV control. If a big data history is available for the AUV application performance, this approach to proactive fault-tolerant control is feasible in AUV single-agent and multi-agent systems as a future work.

Appendix A

MATLAB Implementation Codes

A.1 MATLAB Codes for Chapter 3

```
clc
clear all
close all
global Ts
global Q R N u x p C m Nu agent k r_bar
method=2; %1: Linear 2:nonlinear 3: MPC-RNN
stop_time=20;
computation_time=0;
Ts=0.2;
NN=stop_time/Ts;
na=1;
n=6;
m=3;
p=6;
% EKF initialization
P=eye(n);Rn=diag([5e-2;5e-2;5e-2;1e-2;1e-2;1e-2]).^2;
Qkf=diag([5e-5;5e-5;5e-5;1e-5;1e-5;1e-5]).^2;
moderr=zeros(p,NN);
%MPC initialization
N=9;Nu=1;
Q=10^4*eye(p);%
% Q(4,4)=0;
% Q(5,5)=0;
% Q(6,6)=0;
Q=kron(diag(1.^(1:N)),Q);
```

```
R=1*eye(m);
R=kron(diag(1.^(1:Nu)),R);
%Neural network initialization
epsilon=0.000001;
z_0=0.2*randn(3*Nu*m+2*N*p,1);
%AUV initialization
mu=200;mv=250;mr=80;
du=170;dv=100;dr=50;
C=eye(n);
C_tilde=kron(eye(N),C);
x_ac=zeros(na,n,NN);
x=zeros(na,n,NN); % (E)KF estimated state
y_ac=zeros(na,p,NN);% Actual output
y_m=zeros(na,p,NN); % Measured noisy output

u=zeros(na,m,NN);
for i=1:na
    x_ac(i,:,1)=[0;0;pi/15;0;0;0];
    x(i,:,1)=[0;0;pi/15;0;0;0];
    %u(i,:,1)=[2.8;-pi/6;        -pi/6;  -pi/6];
    u(i,:,1)=[4;4;0];
end
x_ac(:,:,2)=x_ac(:,:,1);
x(:,:,2)=x(:,:,1);
dU=zeros(na,m,NN);
u_bar_min=kron(ones(Nu,1),[-400;     -400;        -100]);
u_bar_max=kron(ones(Nu,1),[400;      400;    100]);
y_bar_min=kron(ones(N,1),[-inf;     -inf;   -inf;   -inf;    -inf;
    -inf]);
y_bar_max=kron(ones(N,1),[inf;       inf;    inf;    inf;      inf;
     inf]);
du_bar_min=-inf*ones(Nu*m,1);
du_bar_max=inf*ones(Nu*m,1);
I_tilde=kron(ones(Nu,Nu),eye(m));I_tilde=tril(I_tilde);
du_bar=zeros(na,m*Nu);
KMPL=zeros(m,N*n,na);
options = optimoptions(@quadprog,'Algorithm','active-set');

if method==1 %linear
    for agent=1:na
        A=[0 0 -x(agent,5,1)*cos(x(agent,3,1))-x(agent,4,1)*sin(x(
            agent,3,1)) cos(x(agent,3,1)) -sin(x(agent,3,1)) 0;
            0 0 x(agent,4,1)*cos(x(agent,3,1))-x(agent,5,1)*sin(x(
                agent,3,1)) sin(x(agent,3,1)) cos(x(agent,3,1)) 0;
            0 0  0 0 0 1;
            0 0 0 -du/mu mv*x(agent,6,1)/mu mv*x(agent,5,1)/mu;
```

```
68                  0  0  0 -mu*x(agent,6,1)/mv -dv/mv -mu*x(agent,4,1)/mv;
69                  0  0  0 (mu-mv)*x(agent,5,1)/mr (mu-mv)*x(agent,4,1)/mr
                        -dr/mr];
70              B=[0  0  0;
71                  0  0  0;
72                  0  0  0;
73                  1/mu 0  0;
74                  0  1/mv  0;
75                  0  0  1/mr];
76              sys=c2d(ss(A,B,C,0),Ts);Ad=sys.A;Bd=sys.B;
77              for q=1:N
78                  for w=1:Nu
79                      if q>=w
80                          G(n*q-n+1:n*q,m*w-m+1:m*w)=Ad^(q-w)*Bd;%
                                calculation of dynamic matrix
81                      end
82                  end
83              end
84              KMPL(:,:,agent)=inv(G'*Q*G+R)*G'*Q;
85          end
86      end
87      %Main control Loop
88
89      for agent=1:na
90          for k=2:NN
91            tic;
92                  % EKF or KF estimation
93           if method~=1 %nonlinear estimation (EKF)
94                  A=[0  0 -x(agent,5,k)*cos(x(agent,3,k))-x(agent,4,k)*
                        sin(x(agent,3,k)) cos(x(agent,3,k)) -sin(x(agent,3,
                        k)) 0;
95                    0  0 x(agent,4,k)*cos(x(agent,3,k))-x(agent,5,k)*
                          sin(x(agent,3,k)) sin(x(agent,3,k)) cos(x(agent
                          ,3,k)) 0;
96                    0  0   0  0  0  1;
97                    0  0  0 -du/mu mv*x(agent,6,k)/mu mv*x(agent,5,k)/mu
                          ;
98                    0  0  0 -mu*x(agent,6,k)/mv -dv/mv -mu*x(agent,4,k)/
                          mv;
99                    0  0  0 (mu-mv)*x(agent,5,k)/mr (mu-mv)*x(agent,4,k)
                          /mr -dr/mr];
100                 B=[0  0  0;
101                   0  0  0;
102                   0  0  0;
103                   1/mu 0  0;
104                   0  1/mv  0;
```

```
105                     0  0  1/mr];
106                 sys=c2d(ss(A,B,C,0),Ts);Ad=sys.A;Bd=sys.B;
107                 phi=expm(Ts*A);
108                 Qd=(phi*Qkf*phi'+Qkf)*Ts/2;
109                 P=phi*P*phi'+Qd;
110                 K=P*inv(P+Rn);
111                 x(agent,:,k)=AUV(x(agent,:,k-1)',u(agent,:,k-1)');
112                 moderr(:,k)=y_m(agent,:,k)-x(agent,:,k);
113                 x(agent,:,k)=x(agent,:,k)+moderr(:,k)'*K';
114                 P=P-K*P;
115         else % Linear Estimation (KF)
116                 x(agent,:,k)=Ad*(x(agent,:,k-1)'-x(agent,:,1)')+Bd*(u(
                        agent,:,k-1)'-u(agent,:,1)')+x(agent,:,1)';
117                 P=Ad*P*Ad'+Qkf;
118                 K=P*inv(P+Rn);
119                 moderr(:,k)=y_m(agent,:,k)-x(agent,:,k);
120                 x(agent,:,k)=x(agent,:,k)+moderr(:,k)'*K';
121                 P=P-K*P;
122         end
123             % MPC
124             r_bar=fun_r_decen(agent,k*Ts);
125             for i=1:N-1
126                 r_bar=[r_bar;fun_r_decen(agent,(k+i)*Ts)];
127             end
128             du_bar(agent,:)=[du_bar(agent,m+1:end) du_bar(agent,end-m
                    +1:end)];
129             if method==2 %nonlinear du_bar(agent,:)
130 %                 [du_bar(agent,:),fval,exitflag]=fminunc(@costAUV,
    du_bar(agent,:));
131 %                 [du_bar(agent,:),fval,exitflag]=lsqnonlin(
    @costAUVlsq,du_bar(agent,:));
132                 [du_bar(agent,:),fval(k),exitflag]=fminsearch(@costAUV
                        ,du_bar(agent,:));
133             elseif method==1%linear
134                 Xfree(:,1)=Ad*(x(agent,:,k)'-x(agent,:,1)')+Bd*(u(
                        agent,:,k-1)'-u(agent,:,1)');
135                 for i=1:N-1
136                     Xfree(:,i+1)=Ad*Xfree(:,i)+Bd*(u(agent,:,k-1)'-u(
                            agent,:,1)');
137                 end
138                 Xfree=Xfree+kron(ones(1,N),x(agent,:,1)');
139                 du_bar(agent,:)=KMPL(:,:,agent)*(r_bar-reshape(Xfree,[
                        n*N,1]));
140             else %NPL
141                 for q=1:N
142                     for w=1:Nu
```

```
143                         if q>=w
144                             G(n*q-n+1:n*q,m*w-m+1:m*w)=Ad^(q-w)*Bd;% calculation of dynamic matrix
145                         end
146                     end
147                 end
148                 Xfree(:,1)=AUV(x(agent,:,k)',u(agent,:,k-1)');
149                 for i=1:N-1
150                     Xfree(:,i+1)=AUV( Xfree(:,i),u(agent,:,k-1)');
151                 end
152                 W=2*(G'*Q*G+R);%W=(W+W')/2;
153                 c= -2*G'*Q*(r_bar-reshape(Xfree,[n*N,1]));
154                 E=[-I_tilde; I_tilde; -C_tilde*G; C_tilde*G; eye(Nu*m)];
155                 b=[-u_bar_min+kron(ones(Nu,1),u(agent,:,k-1)');
156                     u_bar_max-kron(ones(Nu,1),u(agent,:,k-1)');
157                     -y_bar_min+C_tilde*reshape(Xfree,[n*N,1]);
158                     y_bar_max-C_tilde*reshape(Xfree,[n*N,1])];
159                 l=[-inf*ones(2*m*Nu+2*p*N,1); du_bar_min];
160                 h=[b;du_bar_max];
161                 % Recurrent Neural network simulation
162                 if 0
163                     sim('SimulinkFile')
164                     z_0=z(end,:)';
165                     du_bar(agent,:)=delta_u(end,:)';
166                 else
167                     du_bar(agent,:) = quadprog(W,c,E,h,[],[],[],[],[],options);
168 %                     du_bar(agent,:) = quadprog(W,c,[],[],[],[],[],[],[],options);
169                     %           [du_bar,fval,exitflag] = fmincon(@costf,0.001*randn(m*Nu,1)+du_bar,E,h);
170 
171                 end
172 
173             end
174             computation_time=toc;
175             dU(agent,:,k)=du_bar(agent,1:m);
176             u(agent,:,k)=u(agent,:,k-1)+dU(agent,:,k);
177             %Auv Simulation
178             x_ac(agent,:,k+1)=AUV(x_ac(agent,:,k)',u(agent,:,k)'); % actual state
179             y_ac(agent,:,k+1)=C*x_ac(agent,:,k+1)'; % actual output
180 
181             y_m(agent,:,k+1)=y_ac(agent,:,k+1)+diag(sqrt(Rn))'.*randn(1,p); % Measured noisy output
```

```
182
183       end
184 end
```

A.2 MATLAB Codes for Chapter 4

```
1  clc
2  clear all
3  close all
4  global Ts
5  global Q R N u x p C m Nu agent k r_bar W_loe
6  method=2; %1: Linear 2:nonlinear 3: MPC-RNN-DEKF
7  stop_time=200;
8  Ts=0.2;
9  NN=stop_time/Ts;
10 na=1;
11 n=6;
12 m=3;
13 p=6;
14 % DEKF initialization
15 Px=eye(n);Rn=diag([5e-2;5e-2;5e-2;1e-2;1e-2;1e-2]).^2;Re=Rn;
16 W_loe=zeros(m,NN);Pw=eye(m);ff=.999;
17 Qkf=diag([5e-2;5e-2;5e-2;1e-2;1e-2;1e-2]).^2;
18 moderr=zeros(p,NN);
19 rKx_rW(1:n,1:n,1:m)=0;rP_rW(1:n,1:n,1:m)=0;
20 %MPC initialization
21 N=20;Nu=5;
22 Q=50000*eye(p);%
23 % Q(4,4)=0;
24 % Q(5,5)=0;
25 % Q(6,6)=0;
26 Q=kron(diag(1.^(1:N)),Q);
27 R=1*eye(m);
28 R=kron(diag(1.^(1:Nu)),R);
29 %Neural network initialization
30 epsilon=0.000001;
31 z_0=0.2*randn(3*Nu*m+2*N*p,1);
32 %AUV initialization
33 mu=200;mv=250;mr=80;
34 du=170;dv=100;dr=50;
35 C=eye(n);
36 % C=[1 0 0 0 0 0;
37 %    0 1 0 0 0 0;
38 %    0 0 1 0 0 0];
```

```
C_tilde=kron(eye(N),C);
x_ac=zeros(na,n,NN);
x=zeros(na,n,NN); % (E)KF estimated state
W_loe(:,1)=ones(m,1);
y_ac=zeros(na,p,NN);% Actual output
y_m=zeros(na,p,NN); % Measured noisy output

u=zeros(na,m,NN);
for i=1:na
    x_ac(i,:,1)=[0;0;pi/15;0;0;0];
    x(i,:,1)=[0;0;pi/15;0;0;0];
    %u(i,:,1)=[2.8;-pi/6;        -pi/6;  -pi/6];
    u(i,:,1)=[4;4;0];
end
x_ac(:,:,2)=x_ac(:,:,1);
x(:,:,2)=x(:,:,1);
dU=zeros(na,m,NN);
u_bar_min=kron(ones(Nu,1),[-400;     -400;        -100]);
u_bar_max=kron(ones(Nu,1),[200;      400;    100]);
y_bar_min=kron(ones(N,1),[-inf;     -inf;   -inf;   -inf;    -inf;
    -inf]);
y_bar_max=kron(ones(N,1),[inf;       inf;    inf;    inf;      inf;
    inf]);
du_bar_min=-inf*ones(Nu*m,1);
du_bar_max=inf*ones(Nu*m,1);
I_tilde=kron(ones(Nu,Nu),eye(m));I_tilde=tril(I_tilde);
du_bar=zeros(na,m*Nu);
KMPL=zeros(m,N*n,na);
options = optimoptions(@quadprog,'Algorithm','active-set');
if method==1 %linear
    for agent=1:na
        A=[0 0 -x(agent,5,1)*cos(x(agent,3,1))-x(agent,4,1)*sin(x(
            agent,3,1)) cos(x(agent,3,1)) -sin(x(agent,3,1)) 0;
            0 0 x(agent,4,1)*cos(x(agent,3,1))-x(agent,5,1)*sin(x(
                agent,3,1)) sin(x(agent,3,1)) cos(x(agent,3,1)) 0;
            0 0  0 0 0 1;
            0 0 0 -du/mu mv*x(agent,6,1)/mu mv*x(agent,5,1)/mu;
            0 0 0 -mu*x(agent,6,1)/mv -dv/mv -mu*x(agent,4,1)/mv;
            0 0 0 (mu-mv)*x(agent,5,1)/mr (mu-mv)*x(agent,4,1)/mr
                -dr/mr];
        B=[0 0 0;
            0 0 0;
            0 0 0;
            1/mu 0 0;
            0 1/mv 0;
            0 0 1/mr];
```

```
80            sys=c2d(ss(A,B,C,0),Ts);Ad=sys.A;Bd=sys.B;
81            for q=1:N
82                for w=1:Nu
83                    if q>=w
84                        G(n*q-n+1:n*q,m*w-m+1:m*w)=Ad^(q-w)*Bd;%
                              calculation of dynamic matrix
85                    end
86                end
87            end
88            KMPL(:,:,agent)=inv(G'*Q*G+R)*G'*Q;
89        end
90    end
91    %Main control Loop
92    for agent=1:na
93        Cw=[zeros(m,n-m);diag(u(agent,:,1).*[1/mu 1/mv 1/mr])];
94        for k=2:NN
95            tic;
96            % DEKF or KF estimation
97            if method~=1 %nonlinear estimation (DEKF)
98                %%%%%%%%%%%%%%%%%%%%%% Dual Extended Kalman Filter
                      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
99                %//////////////////////// weight time update.....
100               W_loe(:,k)=W_loe(:,k-1);
101               Pw=Pw/ff;
102               %\\\\\\\\\\\\\\\\\\\\\\ weight time update \\\\\\\\\
103               %//////////////////////// state time update.....
104               A=[0 0 -x(agent,5,k-1)*cos(x(agent,3,k-1))-x(agent,4,k
                      -1)*sin(x(agent,3,k-1)) cos(x(agent,3,k-1)) -sin(x(
                      agent,3,k-1)) 0;
105                   0 0 x(agent,4,k-1)*cos(x(agent,3,k-1))-x(agent,5,k
                          -1)*sin(x(agent,3,k-1)) sin(x(agent,3,k-1)) cos
                          (x(agent,3,k-1)) 0;
106                   0 0 0 0 0 1;
107                   0 0 0 -du/mu mv*x(agent,6,k-1)/mu mv*x(agent,5,k
                          -1)/mu;
108                   0 0 0 -mu*x(agent,6,k-1)/mv -dv/mv -mu*x(agent,4,k
                          -1)/mv;
109                   0 0 0 (mu-mv)*x(agent,5,k-1)/mr (mu-mv)*x(agent,4,
                          k-1)/mr -dr/mr];
110               B=[0 0 0;
111                   0 0 0;
112                   0 0 0;
113                   W_loe(1,k)/mu 0 0;
114                   0 W_loe(2,k)/mv 0;
115                   0 0 W_loe(3,k)/mr];
116               sys=c2d(ss(A,B,C,0),Ts);Ad=sys.A;Bd=sys.B;
```

```
117             phi=expm(Ts*A);
118             Qd=(phi*Qkf*phi'+Qkf)*Ts/2;
119             P_x=phi*Px*phi'+Qd;
120             %\\\\\\\\\\\\\\\\\\\\\\ state time update
                    \\\\\\\\\\\\\\\
121             %////////////////////// state filter measurement
                    update...
122             Kx=P_x/(P_x+Rn);
123             x(agent,:,k)=AUV(x(agent,:,k-1)',u(agent,:,k-1)',W_loe
                    (:,k));
124             moderr(:,k)=y_m(agent,:,k)-x(agent,:,k);
125             x(agent,:,k)=x(agent,:,k)+[Kx*moderr(:,k)]';
126             Px=P_x-Kx*P_x;
127             %\\\\\\\\\\\\\\\\\\\\\\ state filter measurement
                    update\\\\
128             %////////////////////// weight filter measurement
                    update...
129             Kw=Pw*Cw'/(Cw*Pw*Cw'+Re);
130             W_loe(:,k)=W_loe(:,k)+Kw*moderr(:,k);%([Ym(7,i);Ym(1,i
                    );Ym(5,i);Ym(3,i)]-X(:,i));
131             Pw=Pw-Kw*Cw*Pw;
132             A=[0 0 -x(agent,5,k)*cos(x(agent,3,k))-x(agent,4,k)*
                    sin(x(agent,3,k)) cos(x(agent,3,k)) -sin(x(agent,3,
                    k)) 0;
133                0 0 x(agent,4,k)*cos(x(agent,3,k))-x(agent,5,k)*
                       sin(x(agent,3,k)) sin(x(agent,3,k)) cos(x(agent
                       ,3,k)) 0;
134                0 0 0 0 0 0 1;
135                0 0 0 -du/mu mv*x(agent,6,k)/mu mv*x(agent,5,k)/mu
                       ;
136                0 0 0 -mu*x(agent,6,k)/mv -dv/mv -mu*x(agent,4,k)/
                       mv;
137                0 0 0 (mu-mv)*x(agent,5,k)/mr (mu-mv)*x(agent,4,k)
                       /mr -dr/mr];
138             phi=expm(Ts*A);
139             if 1 && k>=3%recurrent derivative of the Kalman Gain
140                r2F_rX2(:,:,1)=zeros(6,6);
141                r2F_rX2(:,:,2)=zeros(6,6);
142                r2F_rX2(:,:,3)=[0 0 x(agent,5,k-1)*sin(x(agent,3,k
                       -1))-x(agent,4,k-1)*cos(x(agent,3,k-1)) -sin(x(
                       agent,3,k-1)) -cos(x(agent,3,k-1)) 0;
143                      0 0 -x(agent,4,k-1)*sin(x(agent,3,k-1))-x(
                             agent,5,k-1)*cos(x(agent,3,k-1)) cos(x(
                             agent,3,k-1)) -sin(x(agent,3,k-1)) 0;
144                       zeros(4,6)];
145                r2F_rX2(:,:,4)=[0 0 -sin(x(agent,3,k-1)) 0 0 0;
```

```
146         0  0  cos(x(agent,3,k-1))  0  0  0;
147         0  0   0  0  0  0;
148         0  0  0  0  0  0;
149         0  0  0  0  0 -mu/mv;
150         0  0  0  0  (mu-mv)/mr  0];
151     r2F_rX2(:,:,5)=[0  0 -cos(x(agent,3,k-1))  0  0  0;
152         0  0 -sin(x(agent,3,k-1))  0  0  0;
153         0  0  0  0  0  0;
154         0  0  0  0  0  mv/mu;
155         0  0  0  0  0  0;
156         0  0  0  (mu-mv)/mr  0  0];
157     r2F_rX2(:,:,6)=[zeros(3,6);
158         0  0  0  0  mv/mu  0;
159         0  0  0 -mu/mv  0  0;
160         0  0  0  0  0  0];
161     rA_rW(:,:,1)=[r2F_rX2(:,:,1)*Cwp(:,1)  r2F_rX2
        (:,:,2)*Cwp(:,1)  r2F_rX2(:,:,3)*Cwp(:,1)
        r2F_rX2(:,:,4)*Cwp(:,1)  r2F_rX2(:,:,5)*Cwp(:,1)
         r2F_rX2(:,:,6)*Cwp(:,1)];
162     rA_rW(:,:,2)=[r2F_rX2(:,:,1)*Cwp(:,2)  r2F_rX2
        (:,:,2)*Cwp(:,2)  r2F_rX2(:,:,3)*Cwp(:,2)
        r2F_rX2(:,:,4)*Cwp(:,2)  r2F_rX2(:,:,5)*Cwp(:,2)
         r2F_rX2(:,:,6)*Cwp(:,2)];
163     rA_rW(:,:,3)=[r2F_rX2(:,:,1)*Cwp(:,3)  r2F_rX2
        (:,:,2)*Cwp(:,3)  r2F_rX2(:,:,3)*Cwp(:,3)
        r2F_rX2(:,:,4)*Cwp(:,3)  r2F_rX2(:,:,5)*Cwp(:,3)
         r2F_rX2(:,:,6)*Cwp(:,3)];
164     rP_rW(:,:,1)=rA_rW(:,:,1)*Pxp*phip'+phip*(-rKx_rW
        (:,:,1)*P_xp+(eye(6)-Kxp)*rP_rW(:,:,1))*phip'+
        phip*Pxp*rA_rW(:,:,1)';
165     rP_rW(:,:,2)=rA_rW(:,:,2)*Pxp*phip'+phip*(-rKx_rW
        (:,:,2)*P_xp+(eye(6)-Kxp)*rP_rW(:,:,2))*phip'+
        phip*Pxp*rA_rW(:,:,2)';
166     rP_rW(:,:,3)=rA_rW(:,:,3)*Pxp*phip'+phip*(-rKx_rW
        (:,:,3)*P_xp+(eye(6)-Kxp)*rP_rW(:,:,3))*phip'+
        phip*Pxp*rA_rW(:,:,3)';
167     rKx_rW(:,:,1)=(eye(6)-Kx)*rP_rW(:,:,1)/(P_x+Rn);
168     rKx_rW(:,:,2)=(eye(6)-Kx)*rP_rW(:,:,2)/(P_x+Rn);
169     rKx_rW(:,:,3)=(eye(6)-Kx)*rP_rW(:,:,3)/(P_x+Rn);
170     Cwp=Cw;
171     Cw(:,1)=phi*((eye(6)-Kx)*Cw(:,1)+rKx_rW(:,:,1)*
        moderr(:,k))+Ts*[0;0;0;u(agent,1,k-1)/mu;0;0];
172     Cw(:,2)=phi*((eye(6)-Kx)*Cw(:,2)+rKx_rW(:,:,2)*
        moderr(:,k))+Ts*[0;0;0;0;u(agent,2,k-1)/mv;0];
173     Cw(:,3)=phi*((eye(6)-Kx)*Cw(:,3)+rKx_rW(:,:,3)*
        moderr(:,k))+Ts*[0;0;0;0;0;u(agent,3,k-1)/mr];
```

```
174             else
175                 Cwp=Cw;
176                 Cw=phi*((eye(6)-Kx)*Cw)+Ts*[zeros(m,n-m);diag(u(
                        agent,:,k-1).*[1/mu 1/mv 1/mr])];
177             end
178             phip=phi;Pxp=Px;P_xp=P_x;Kxp=Kx;
179             %\\\\\\\\\\\\\\\\\\\\\\\\ weight filter measurement
                    update\\\\
180         else % Linear Estimation (KF)
181             x(agent,:,k)=Ad*(x(agent,:,k-1)'-x(agent,:,1)')+Bd*(u(
                    agent,:,k-1)'-u(agent,:,1)')+x(agent,:,1)';
182             P=Ad*P*Ad'+Qkf;
183             K=P*inv(P+Rn);
184             moderr(:,k)=y_m(agent,:,k)-x(agent,:,k);
185             x(agent,:,k)=x(agent,:,k)+moderr(:,k)'*K';
186             P=P-K*P;
187         end
188         % MPC
189         r_bar=fun_r_decen(agent,k*Ts);
190         for i=1:N-1
191             r_bar=[r_bar;fun_r_decen(agent,(k+i)*Ts)];
192         end
193         du_bar(agent,:)=[du_bar(agent,m+1:end) du_bar(agent,end-m
                +1:end)];
194         if method==2 %nonlinear du_bar(agent,:)
195             %               [du_bar(agent,:),fval,exitflag]=fminunc(
                    @costAUV,du_bar(agent,:));
196             %               [du_bar(agent,:),fval,exitflag]=
                    lsqnonlin(@costAUVlsq,du_bar(agent,:));
197             [du_bar(agent,:),fval,exitflag]=fminsearch(@costAUV,
                    du_bar(agent,:));
198
199         elseif method==1%linear
200             Xfree(:,1)=Ad*(x(agent,:,k)'-x(agent,:,1)')+Bd*(u(
                    agent,:,k-1)'-u(agent,:,1)');
201             for i=1:N-1
202                 Xfree(:,i+1)=Ad*Xfree(:,i)+Bd*(u(agent,:,k-1)'-u(
                        agent,:,1)');
203             end
204             Xfree=Xfree+kron(ones(1,N),x(agent,:,1)');
205             du_bar(agent,:)=KMPL(:,:,agent)*(r_bar-reshape(Xfree,[
                    n*N,1]));
206         else %NPL
207             for q=1:N
208                 for w=1:Nu
209                     if q>=w
```

```
                    G(n*q-n+1:n*q,m*w-m+1:m*w)=Ad^(q-w)*Bd;% calculation of dynamic matrix
                end
            end
        end
        Xfree(:,1)=AUV(x(agent,:,k)',u(agent,:,k-1)',W_loe(:,k));
        for i=1:N-1
            Xfree(:,i+1)=AUV( Xfree(:,i),u(agent,:,k-1)',W_loe(:,k));
        end
        W=2*(G'*Q*G+R);W=(W+W')/2;
        c= -2*G'*Q*(r_bar-reshape(Xfree,[n*N,1]));
        E=[-I_tilde; I_tilde; -C_tilde*G; C_tilde*G; eye(Nu*m)];
        b=[-u_bar_min+kron(ones(Nu,1),u(agent,:,k-1)');
            u_bar_max-kron(ones(Nu,1),u(agent,:,k-1)');
            -y_bar_min+C_tilde*reshape(Xfree,[n*N,1]);
            y_bar_max-C_tilde*reshape(Xfree,[n*N,1])];
        l=[-inf*ones(2*m*Nu+2*p*N,1); du_bar_min];
        h=[b;du_bar_max];
        % Recurrent Neural network simulation
        if 0
            sim('SimulinkFile')
            z_0=z(end,:)';
            du_bar(agent,:)=delta_u(end,:)';
        else
            du_bar(agent,:) = quadprog(W,c,E,h,[],[],[],[],[],options);
            %                 du_bar(agent,:) = quadprog(W,c,[],[],[],[],[],[],[],options);
            %           [du_bar,fval,exitflag] = fmincon(@costf,0.001*randn(m*Nu,1)+du_bar,E,h);
        end
    end
    computation_time=toc;
    dU(agent,:,k)=du_bar(agent,1:m);
    u(agent,:,k)=u(agent,:,k-1)+dU(agent,:,k);%+randn(1,m);
    u(agent,u(agent,:,k)'>u_bar_max(1:m,1),k)=u_bar_max(u(agent,:,k)'>u_bar_max(1:m,1))';
    u(agent,u(agent,:,k)'<u_bar_min(1:m,1),k)=u_bar_min(u(agent,:,k)'<u_bar_min(1:m,1))';
    %Auv Simulation
    if 1 %&& k>NN/3
        loe=[0.25;0.5;0.75];
    else
```

```
247                 loe=[1;1;1];
248             end
249             x_ac(agent,:,k+1)=AUV(x_ac(agent,:,k)',u(agent,:,k)',loe);
                    %actual state
250             y_ac(agent,:,k+1)=C*x_ac(agent,:,k+1)'; % actual output
251             y_m(agent,:,k+1)=y_ac(agent,:,k+1)+diag(sqrt(Rn))'.*randn
                    (1,p); % Measured noisy output
252         end
253     end
```

Bibliography

[1] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[2] X. Wang, Z. Yan, and J. Wang, "Model predictive control of multi-robot formation based on the simplified dual neural network," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 3161–3166.

[3] Underwater Vehicles. http://www.whoi.edu/main/underwater-vehicles.

[4] C. Edwards, T. Lombaerts, and H. Smaili, "Fault tolerant flight control," *Lecture Notes in Control and Information Sciences*, vol. 399, pp. 1–560, 2010.

[5] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual reviews in control*, vol. 32, no. 2, pp. 229–252, 2008.

[6] "The Computational Intelligence Hierarchy," http://dailyvillain.com/hierarchy.htm.

[7] V. A. Akpan and G. D. Hassapis, "Nonlinear model identification and adaptive model predictive control using neural networks," *ISA transactions*, vol. 50, no. 2, pp. 177–194, 2011.

[8] D. P. Mandic, J. A. Chambers *et al.*, *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley Online Library, 2001.

[9] B. Allen, R. Stokey, T. Austin, N. Forrester, R. Goldsborough, M. Purcell, and C. Von Alt, "Remus: a small, low cost auv; system description, field trials and performance results," in *OCEANS'97. MTS/IEEE Conference Proceedings*, vol. 2. IEEE, 1997, pp. 994–1000.

[10] F. El-Hawary, *The ocean engineering handbook*. Crc Press, 2000.

[11] Y. Xia and J. Wang, "A one-layer recurrent neural network for support vector machine learning," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 2, pp. 1261–1269, 2004.

[12] S. S. Haykin *et al.*, *Kalman filtering and neural networks*. Wiley Online Library, 2001.

[13] M. R. Davoodi, N. Meskin, and K. Khorasani, "Simultaneous fault detection, isolation and control tracking design using a single observer-based module," in *American Control Conference (ACC), 2014*. IEEE, 2014, pp. 3047–3052.

[14] J. Yuh, "Design and control of autonomous underwater robots: A survey," *Autonomous Robots*, vol. 8, no. 1, pp. 7–24, 2000.

[15] B. N. Rath, "Adaptive depth control algorithms for an autonomous underwater vehicle," Ph.D. dissertation, NATIONAL INSTITUTE OF TECHNOLOGY ROURKELA, 2014.

[16] "WHAT IS AN ROV," http://www.rov.org/rov_overview.cfm.

[17] S. Byrne and V. Schmidt, "Uncertainty modeling for auv acquired bathymetry," 2015.

[18] E. Kobayashi, T. Aoki, T. Maeda, K. Hirokawa, T. Ichikawa, T. Saitou, S. Miyamoto, S. Iwasaki, and H. Kobayashi, "Development of an autonomous underwater vehicle maneuvering simulator," in *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, vol. 1. IEEE, 2001, pp. 361–368.

[19] J. Wu, J. Liu, and H. Xu, "A variable buoyancy system and a recovery system developed for a deep-sea auv qianlong i," in *Oceans 2014-Taipei*. IEEE, 2014, pp. 1–4.

[20] C. German, D. Connelly, R. Prien, D. Yoerger, M. Jakuba, A. Bradley, T. Shank, K. Nakamura, C. Langmuir, and L. Parsons, "New techniques for hydrothermal plume investigation by auv," in *Geophysical Research Abstracts*, vol. 7, 2005, p. 04361.

[21] R. Henthorn, D. W. Caress, H. Thomas, R. McEwen, W. Kirkwood, C. Paull, and R. Keaten, "High-resolution multibeam and subbottom surveys of submarine canyons, deep-sea fan channels, and gas seeps using the mbari mapping auv," in *OCEANS 2006*. IEEE, 2006, pp. 1–6.

[22] K. M. Sharp and R. H. White, *More tools in the toolbox: The naval oceanographic office's Remote Environmental Monitoring UnitS (REMUS) 6000 AUV*. IEEE, 2008.

[23] D. Yoerger and J. Slotine, "Robust trajectory control of underwater vehicles," *IEEE Journal of Oceanic Engineering*, vol. 10, no. 4, pp. 462–470, 1985.

[24] R. Hare, A. Godin, and L. Mayer, "Accuracy estimation of canadian swath (multibeam) and sweep (multitransducer) sounding systems," *Canadian Hydrographic Service and University of New Brunswick Publication, Fredericton*, 1995.

[25] Y. Nakamura and S. Savant, "Nonlinear tracking control of autonomous underwater vehicles," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 1992, pp. A4–A9.

[26] K. R. Goheen and E. R. Jefferys, "Multivariable self-tuning autopilots for autonomous and remotely operated underwater vehicles," *IEEE Journal of Oceanic Engineering*, vol. 15, no. 3, pp. 144–151, 1990.

[27] S. K. Choi and J. Yuh, "Experimental study on a learning control system with bound estimation for underwater robots," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 3. IEEE, 1996, pp. 2160–2165.

[28] S. S. Tabaii, F. El-Hawary, and M. El-Hawary, "Hybrid adaptive control of autonomous underwater vehicle," in *Autonomous Underwater Vehicle Technology, 1994. AUV'94., Proceedings of the 1994 Symposium on*. IEEE, 1994, pp. 275–282.

[29] K. Ishii, T. Fujii, and T. Ura, "Neural network system for online controller adaptation and its application to underwater robot," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 1. IEEE, 1998, pp. 756–761.

[30] C. Tsukamoto, J. Yuh, S. K. Choi, C. Lee, and J. Lorentz, "Experimental study of advanced controllers for an underwater robotic vehicle thruster system," *Intelligent Automation & Soft Computing*, vol. 5, no. 3, pp. 225–238, 1999.

[31] H. Mahesh, J. Yuh, and R. Lakshmi, "A coordinated control of an underwater vehicle and robotic manipulator," *Journal of Field Robotics*, vol. 8, no. 3, pp. 339–370, 1991.

[32] T. W. McLain, S. M. Rock, and M. J. Lee, "Experiments in the coordinated control of an underwater arm/vehicle system," *Autonomous robots*, vol. 3, no. 2-3, pp. 213–232, 1996.

[33] T. J. Tarn, G. Shoults, and S. Yang, "A dynamic model of an underwater vehicle with a robotic manipulator using kane's method," *Autonomous Robots*, vol. 3, no. 2, pp. 269–283, 1996.

[34] G. Antonelli and S. Chiaverini, "Task-priority redundancy resolution for underwater vehicle-manipulator systems," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 1. IEEE, 1998, pp. 768–773.

[35] C. C. De Wit, E. O. Diaz, and M. Perrier, "Robust nonlinear control of an underwater vehicle/manipulator system with composite dynamics," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 1. IEEE, 1998, pp. 452–457.

[36] C. Silvestre, "Multi-objective optimization theory with applications to the integrated design of controllers/plants for autonomous vehicles," *dissertation*, 2000.

[37] C. Silvestre and A. Pascoal, "Control of the infante auv using gain scheduled static output feedback," *Control Engineering Practice*, vol. 12, no. 12, pp. 1501–1509, 2004.

[38] C. Silvestre, A. Pascoal, and A. Healey, "Auv control under wave disturbances," in *International Symposium on Unmanned Untethered Submersible Technology*. UNIVERSITY OF NEW HAMPSHIRE-MARINE SYSTEMS, 1997, pp. 228–239.

[39] A. Pascoal, C. Silvestre, and P. Oliveira, "Vehicle and mission control of single and multiple autonomous marine robots," *IEE Control Engineering Series*, vol. 69, p. 353, 2006.

[40] A. P. Aguiar and A. M. Pascoal, "Regulation of a nonholonomic autonomous underwater vehicle with parametric modeling uncertainty using lyapunov functions," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 5. IEEE, 2001, pp. 4178–4183.

[41] R. W. Brockett *et al.*, "Asymptotic stability and feedback stabilization," *Differential geometric control theory*, vol. 27, no. 1, pp. 181–191, 1983.

[42] J. Hauser and R. Hindman, "Maneuver regulation from trajectory tracking: Feedback linearizable systems," *IFAC Proceedings Volumes*, vol. 28, no. 14, pp. 595–600, 1995.

[43] R. Skjetne, T. I. Fossen, and P. V. Kokotović, "Robust output maneuvering for a class of nonlinear systems," *Automatica*, vol. 40, no. 3, pp. 373–383, 2004.

[44] R. Skjetne, A. R. Teel, and P. V. Kokotovic, "Stabilization of sets parametrized by a single variable: Application to ship maneuvering," in *Proc. 15th Int. Symp. Mathematical Theory of Networks and Systems*, 2002.

[45] W. Dixon, D. M. Dawson, E. Zergeroglu, and A. Behal, *Nonlinear control of wheeled mobile robots*, 2001.

[46] W. Lin and C. I. Byrnes, "Design of discrete-time nonlinear control systems via smooth feedback," *IEEE Transactions on Automatic Control*, vol. 39, no. 11, pp. 2340–2346, 1994.

[47] A. Isidori, *Nonlinear control theory*. Springer-Verlag, New York, 1989.

[48] H. K. Khalil, "Adaptive output feedback control of nonlinear systems represented by input-output models," *IEEE Transactions on Automatic Control*, vol. 41, no. 2, pp. 177–188, 1996.

[49] P. Encarnacao, A. Pascoal, and M. Arcak, "Path following for autonomous marine craft," *IFAC Proceedings Volumes*, vol. 33, no. 21, pp. 117–122, 2000.

[50] G. Papadimitriou, Z. Saigol, and D. M. Lane, "Enabling fault recovery and adaptation in mine-countermeasures missions using ontologies," in *OCEANS 2015-Genova*. IEEE, 2015, pp. 1–7.

[51] W. Naeem, R. Sutton*, J. Chudley, F. Dalgleish, and S. Tetlow, "An online genetic algorithm based model predictive control autopilot design with experimental verification," *International Journal of Control*, vol. 78, no. 14, pp. 1076–1090, 2005.

[52] S. R. Ahmadzadeh, M. Leonetti, A. Carrera, M. Carreras, P. Kormushev, and D. G. Caldwell, "Online discovery of auv control policies to overcome thruster failures," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6522–6528.

[53] P. Babcock and J. Zinchuk, "Fault-tolerant design optimization: Application to an autonomous underwater vehicle navigation system," in *Autonomous Underwater Vehicle Technology, 1990. AUV'90., Proceedings of the (1990) Symposium on*. IEEE, 1990, pp. 34–43.

[54] S. Cannon and D. Dunn, "A high-level model for the development of fault-tolerant parallel and distributed systems," *University of Utah Computer Science TR A*, vol. 192, 1992.

[55] A. Orrick, M. McDermott, D. M. Barnett, E. L. Nelson, and G. N. Williams, "Failure detection in an autonomous underwater vehicle," in *Autonomous Underwater Vehicle Technology, 1994. AUV'94., Proceedings of the 1994 Symposium on*. IEEE, 1994, pp. 377–382.

[56] M. Takai and T. Ura, "Development of a system to diagnose autonomous underwater vehicles," *International Journal of Systems Science*, vol. 30, no. 9, pp. 981–988, 1999.

[57] K. Yang, J. Yuh, and S. K. Choi, "Fault-tolerant system design of an autonomous underwater vehicle odin: an experimental study," *International Journal of Systems Science*, vol. 30, no. 9, pp. 1011–1019, 1999.

[58] M. Leonetti, S. R. Ahmadzadeh, and P. Kormushev, "On-line learning to recover from thruster failures on autonomous underwater vehicles," in *Oceans-San Diego, 2013*. IEEE, 2013, pp. 1–6.

[59] M. Blanke, M. Staroswiecki, and N. E. Wu, "Concepts and methods in fault-tolerant control," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 4. IEEE, 2001, pp. 2606–2620.

[60] R. J. Patton, "Fault-tolerant control: the 1997 situation," *IFAC Proceedings Volumes*, vol. 30, no. 18, pp. 1029–1051, 1997.

[61] R. F. Stengel and L. Ryan, "Stochastic robustness of linear time-invariant control systems," *IEEE Transactions on Automatic Control*, vol. 36, no. 1, pp. 82–87, 1991.

[62] P. R. Chandler, "Self-repairing flight control system reliability and maintainability program executive overview," in *Proceedings of the IEEE national aerospace and electronics conference*, 1984, pp. 586–590.

[63] J. S. Eterno, J. L. Weiss, D. P. Looze, and A. Willsky, "Design issues for fault tolerant-restructurable aircraft control," in *Decision and Control, 1985 24th IEEE Conference on*, vol. 24. IEEE, 1985, pp. 900–905.

[64] D. D. Moerder, N. Halyo, J. R. Broussard, and A. K. Caglayan, "Application of precomputed control laws in a reconfigurable aircraft flight control system," *Journal of Guidance, Control, and Dynamics*, vol. 12, no. 3, pp. 325–333, 1989.

[65] D. Looze, J. Weiss, J. Eterno, and N. Barrett, "An automatic redesign approach for restructurable control systems," *IEEE Control systems magazine*, vol. 5, no. 2, pp. 16–22, 1985.

[66] R. Montoya, W. Howell, W. Bundick, A. Ostroff, R. Hueschen, and C. M. Belcastro, "Restructurable controls," 1983.

[67] J. Monaco, D. Ward, R. Barron, and R. Bird, "Implementation and flight test assessment of an adaptive, reconfigurable flight control system," in *Proceedings of the 1997 AIAA Guidance Navigation and Control Conference, AIAA Paper*, vol. 97, 1997, p. 3738.

[68] W. T. Vetterling, *Numerical Recipes Example Book (C++): The Art of Scientific Computing*. Cambridge University Press, 2002.

[69] Q. Shen, B. Jiang, and P. Shi, "Neural network-based fault tolerant control scheme against un-modeled fault," in *Fault Diagnosis and Fault-Tolerant Control Based on Adaptive Control Approach*. Springer, 2017, pp. 163–190.

[70] G. Vallee, K. Charoenpornwattana, C. Engelmann, A. Tikotekar, C. Leangsuksun, T. Naughton, and S. L. Scott, "A framework for proactive fault tolerance," in *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*. IEEE, 2008, pp. 659–664.

[71] R. A. Oldfield, "Investigating lightweight storage and overlay networks for fault tolerance," in *Proceedings of the High Availability and Performance Computing Workshop*, 2006.

[72] D. M. de la Pena and P. D. Christofides, "Lyapunov-based model predictive control of nonlinear systems subject to data losses," *IEEE Transactions on Automatic Control*, vol. 53, no. 9, pp. 2076–2089, 2008.

[73] J. Liu, D. Muñoz de la Peña, P. D. Christofides, and J. F. Davis, "Lyapunov-based model predictive control of nonlinear systems subject to time-varying measurement delays," *International Journal of Adaptive Control and Signal Processing*, vol. 23, no. 8, pp. 788–807, 2009.

[74] D. L. Poole, A. K. Mackworth, and R. Goebel, *Computational intelligence: a logical approach*. Oxford University Press New York, 1998, vol. 79.

[75] M. Ö. Efe, O. Kaynak, and X. Yu, "Variable structure systems theory in computational intelligence," in *Variable Structure Systems: Towards the 21st Century*. Springer, 2002, pp. 365–390.

[76] S. Tangirala, R. Kumar, S. Bhattacharyya, M. O'Connor, and L. Holloway, "Hybrid-model based hierarchical mission control architecture for autonomous underwater vehicles," in *American Control Conference, 2005. Proceedings of the 2005*. IEEE, 2005, pp. 668–673.

[77] A. Hajiloo, "Robust and multi-objective model predictive control design for nonlinear systems," Ph.D. dissertation, Concordia University, 2016.

[78] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part i: Quantitative model-based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 293–311, 2003.

[79] W. Wang, D. E. Rivera, and K. G. Kempf, "Model predictive control strategies for supply chain management in semiconductor manufacturing," *International Journal of Production Economics*, vol. 107, no. 1, pp. 56–77, 2007.

[80] S. Meyn, *Control techniques for complex networks*. Cambridge University Press, 2008.

[81] E. G. Cho, K. A. Thoney, T. J. Hodgson, and R. E. King, "Rolling horizon scheduling of multi-factory supply chains," in *Simulation Conference, 2003. Proceedings of the 2003 Winter*, vol. 2. IEEE, 2003, pp. 1409–1416.

[82] F. Herzog, *Strategic portfolio management for long-term investments*. Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 16137, 2005, 2005.

[83] J. A. Primbs, "Dynamic hedging of basket options under proportional transaction costs using receding horizon control," *International Journal of Control*, vol. 82, no. 10, pp. 1841–1855, 2009.

[84] K. T. Talluri and G. J. Van Ryzin, *The theory and practice of revenue management*. Springer Science & Business Media, 2006, vol. 68.

[85] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[86] D. Gorinevsky, "Model predictive control: Industrial mpc."

[87] D. Di Ruscio, "Model predictive control and optimization," *Telemark University College*, 2001.

[88] J. Dong, M. Verhaegen, and E. Holweg, "Closed-loop subspace predictive control for fault tolerant mpc design," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 3216–3221, 2008.

[89] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[90] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[91] P. Tatjewski, *Advanced control of industrial processes: structures and algorithms*. Springer Science & Business Media, 2007.

[92] M. A. Henson, "Nonlinear model predictive control: current status and future directions," *Computers & Chemical Engineering*, vol. 23, no. 2, pp. 187–202, 1998.

[93] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[94] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[95] D. Corona and B. D. Schutter, "Adaptive cruise control for a smart car: A comparison benchmark for mpc-pwa control methods," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 365–372, March 2008.

[96] R. Khan, P. Williams, P. Riseborough, A. Rao, and R. Hill, "Designing a nonlinear model predictive controller for fault tolerant flight control," *arXiv preprint arXiv:1609.01529*, 2016.

[97] M. Rau and D. Schroder, "Model predictive control with nonlinear state space models," in *Advanced Motion Control, 2002. 7th International Workshop on*. IEEE, 2002, pp. 136–141.

[98] S. Martinez, J. Cortes, and F. Bullo, "Motion coordination with distributed information," *Control Systems, IEEE*, vol. 27, no. 4, pp. 75–88, 2007.

[99] P. O. Scokaert, D. Q. Mayne, and J. B. Rawlings, "Suboptimal model predictive control (feasibility implies stability)," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 648–654, 1999.

[100] H. Michalska and D. Q. Mayne, "Robust receding horizon control of constrained nonlinear systems," *IEEE Transactions on automatic control*, vol. 38, no. 11, pp. 1623–1633, 1993.

[101] M. Ławryńczuk, "Modelling and nonlinear predictive control of a yeast fermentation biochemical reactor using neural networks," *Chemical Engineering Journal*, vol. 145, no. 2, pp. 290–307, 2008.

[102] C. Ocampo-Martinez and V. Puig, "Fault-tolerant model predictive control within the hybrid systems framework: Application to sewer networks," *international journal of adaptive control and signal processing*, vol. 23, no. 8, pp. 757–787, 2009.

[103] O. Kaynak, K. Erbatur, and M. Ertugnrl, "The fusion of computationally intelligent methodologies and sliding-mode control-a survey," *Industrial Electronics, IEEE Transactions on*, vol. 48, no. 1, pp. 4–17, 2001.

[104] W. Naeem, "Model predictive control of an autonomous underwater vehicle," in *Proceedings of UKACC 2002 Postgraduate Symposium, Sheffield, UK, September*, 2002, pp. 19–23.

[105] H. Al-Duwaish and W. Naeem, "Nonlinear model predictive control of hammerstein and wiener models using genetic algorithms," in *Control Applications, 2001.(CCA'01). Proceedings of the 2001 IEEE International Conference On*. IEEE, 2001, pp. 465–469.

[106] R. Cui, S. S. Ge, B. V. E. How, and Y. S. Choo, "Leader–follower formation control of underactuated autonomous underwater vehicles," *Ocean Engineering*, vol. 37, no. 17, pp. 1491–1502, 2010.

[107] H. Mehrjerdi, "Control and coordination for a group of mobile robots in unknown environments," Ph.D. dissertation, École de technologie supérieure, 2010.

[108] A. Bartoszewicz, "On the robustness of variable structure systems in the presence of measurement noise," in *Industrial Electronics Society, 1998. IECON'98. Proceedings of the 24th Annual Conference of the IEEE*, vol. 3. IEEE, 1998, pp. 1733–1736.

[109] J. M. Maciejowski and C. N. Jones, "Mpc fault-tolerant flight control case study: Flight 1862," *IFAC Proceedings Volumes*, vol. 36, no. 5, pp. 119–124, 2003.

[110] J. M. Maciejowski, "The implicit daisy-chaining property of constrained predictive control," *Applied Mathematics and Computer Science*, vol. 8, pp. 695–712, 1998.

[111] A. P. Deshpande, S. C. Patwardhan, and S. S. Narasimhan, "Intelligent state estimation for fault tolerant nonlinear predictive control," *Journal of Process control*, vol. 19, no. 2, pp. 187–204, 2009.

[112] A. Hennig and G. J. Balas, "Mpc supervisory flight controller: A case study to flight el al 1862," in *AIAA Guidance, Navigation and Control Conference and Exhibit*. Honolulu, Hawaii, 2008, p. 6789.

[113] A. Bemporad and M. Morari, "Robust model predictive control: A survey," *Robustness in identification and control*, pp. 207–226, 1999.

[114] M. Blanke, M. Kinnaert, J. Lunze, M. Staroswiecki, and J. Schröder, *Diagnosis and fault-tolerant control*. Springer, 2006, vol. 691.

[115] N. Roofigari Esfahani, "Control and fault accommodation for attitude control subsystem of formation flying satellites subject to constraints," 2013.

[116] S. Sedaghati, "Formation control and fault accommodation for a team of autonomous underwater vehicles," 2015.

[117] M. Gopinathan, J. D. Boskovic, R. K. Mehra, and C. Rago, "A multiple model predictive scheme for fault-tolerant flight control design," in *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, vol. 2. IEEE, 1998, pp. 1376–1381.

[118] F. Xu, V. Puig, C. Ocampo-Martinez, S. Olaru, and S.-I. Niculescu, "Robust mpc for actuator-fault tolerance using set-based passive fault detection and active fault isolation," *International Journal of Applied Mathematics and Computer Science*, vol. 27, no. 1, pp. 43–61, 2017.

[119] L. Lapierre and B. Jouvencel, "Robust nonlinear path-following control of an auv," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 2, pp. 89–102, 2008.

[120] L. Lao, M. Ellis, and P. D. Christofides, "Proactive fault-tolerant model predictive control," *AIChE Journal*, vol. 59, no. 8, pp. 2810–2820, 2013.

[121] A. Houacine, "Regularized fast recursive least squares algorithms for finite memory filtering," *IEEE Transactions on signal processing*, vol. 40, no. 4, pp. 758–769, 1992.

[122] V. J. Aidala, "Kalman filter behavior in bearings-only tracking applications," *IEEE Transactions on Aerospace and Electronic Systems*, no. 1, pp. 29–39, 1979.

[123] E.-J. Park, E. Bae, J. Yi, Y. Kim, K. Choi, S. H. Lee, J. Yoon, B. C. Lee, and K. Park, "Repeated-dose toxicity and inflammatory responses in mice by oral administration of silver nanoparticles," *Environmental toxicology and pharmacology*, vol. 30, no. 2, pp. 162–168, 2010.

[124] M. Sepasi and F. Sassani, "On-line fault diagnosis of hydraulic systems using unscented kalman filter," *International Journal of Control, Automation and Systems*, vol. 8, no. 1, pp. 149–156, 2010.

[125] M. S. Grewal, "Kalman filtering," in *International Encyclopedia of Statistical Science*. Springer, 2011, pp. 705–708.

[126] C. V. Rao, J. B. Rawlings, and D. Q. Mayne, "Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations," *IEEE transactions on automatic control*, vol. 48, no. 2, pp. 246–258, 2003.

[127] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. IEEE, 2000, pp. 153–158.

[128] H. A. Izadi, Y. Zhang, and B. W. Gordon, "Fault tolerant model predictive control of quad-rotor helicopters with actuator fault estimation," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 6343–6348, 2011.

[129] D. L. Marruedo, T. Alamo, and E. Camacho, "Input-to-state stable mpc for constrained discrete-time nonlinear systems with bounded additive uncertainties," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 4. IEEE, 2002, pp. 4619–4624.

[130] E. Sobhani-Tehrani, H. Talebi, and K. Khorasani, "A nonlinear hybrid fault detection, isolation and estimation using bank of neural parameter estimators," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 7251–7258, 2008.

[131] E. Sobhani-Tehrani, H. A. Talebi, and K. Khorasani, "Hybrid fault diagnosis of nonlinear systems using neural parameter estimators," *Neural Networks*, vol. 50, pp. 12–32, 2014.

[132] A. Mirzaee and K. Salahshoor, "Fault diagnosis and accommodation of nonlinear systems based on multiple-model adaptive unscented kalman filter and switched mpc and h-infinity loop-shaping controller," *Journal of Process Control*, vol. 22, no. 3, pp. 626–634, 2012.

[133] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, Feb 2002.

[134] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear estimation*. Prentice Hall Upper Saddle River, NJ, 2000, vol. 1.

[135] F. Gustafsson and F. Gustafsson, *Adaptive filtering and change detection*. Wiley New York, 2000, vol. 1.

[136] D. Crisan and A. Doucet, "A survey of convergence results on particle filtering methods for practitioners," *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 736–746, Mar 2002.

[137] R. Van Der Merwe and E. A. Wan, "The square-root unscented kalman filter for state and parameter-estimation," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 6. IEEE, 2001, pp. 3461–3464.

[138] D. Guo and X. Wang, "Quasi-monte carlo filtering in nonlinear dynamic systems," *IEEE transactions on signal processing*, vol. 54, no. 6, pp. 2087–2098, 2006.

[139] O. Cappé, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential monte carlo," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 899–924, 2007.

[140] Q. Cheng and P. Bondon, "A new unscented particle filter," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, March 2008, pp. 3417–3420.

[141] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 3562–3568.

[142] F. Gustafsson and G. Hendeby, "Some relations between extended and unscented kalman filters," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 545–555, Feb 2012.

[143] I. Arasaratnam and S. Haykin, "Cubature kalman filters," *IEEE Transactions on automatic control*, vol. 54, no. 6, pp. 1254–1269, 2009.

[144] P. H. Leong, S. Arulampalam, T. A. Lamahewa, and T. D. Abhayapala, "A gaussian-sum based cubature kalman filter for bearings-only tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 2, pp. 1161–1176, APRIL 2013.

[145] E. A. Wan and A. T. Nelson, "Dual extended kalman filter methods," *Kalman filtering and neural networks*, pp. 123–173, 2001.

[146] T. A. Wenzel, K. Burnham, M. Blundell, and R. Williams, "Dual extended kalman filter for vehicle state and parameter estimation," *Vehicle System Dynamics*, vol. 44, no. 2, pp. 153–171, 2006.

[147] P. Hamelin, P. Bigras, J. Beaudry, P.-L. Richard, and M. Blain, "Discrete-time state feedback with velocity estimation using a dual observer: application to an underwater direct-drive grinding robot," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 1, pp. 187–191, 2012.

[148] Z. Wei, G. Yi, M. Detao, L. Zhicheng, and C. Tao, "Research on diving control of underactuated uuv based on model predictive control with artificial bee colony algorithm," in *Control Conference (CCC), 2015 34th Chinese*. IEEE, 2015, pp. 4073–4078.

[149] Z. Gao, T. Breikin, and H. Wang, "Discrete-time proportional and integral observer and observer-based controller for systems with both unknown input and output disturbances," *Optimal Control Applications and Methods*, vol. 29, no. 3, pp. 171–189, 2008.

[150] K. Zhang, B. Jiang, P. Shi, and A. Shumsky, "Reduced-order fault estimation observer design for discrete-time systems," in *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*. IEEE, 2012, pp. 2959–2964.

[151] J. C. Kinsey, Q. Yang, and J. C. Howland, "Nonlinear dynamic model-based state estimators for underwater navigation of remotely operated vehicles," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 5, pp. 1845–1854, 2014.

[152] S. Mahapatra, B. Subudhi, and R. Rout, "Diving control of an autonomous underwater vehicle using nonlinear h measurement feedback technique," in *OCEANS 2016-Shanghai*. IEEE, 2016, pp. 1–5.

[153] K. Zhang, B. Jiang, and P. Shi, "Fast fault estimation and accommodation for dynamical systems," *IET Control Theory & Applications*, vol. 3, no. 2, pp. 189–199, 2009.

[154] X. Yao, G. Yang, and Y. Peng, "Nonlinear reduced-order observer-based predictive control for diving of an autonomous underwater vehicle," *Discrete Dynamics in Nature and Society*, vol. 2017, 2017.

[155] K. Ishii, T. Fujii, and T. Ura, "Neural network system for online controller adaptation and its application to underwater robot," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 1. IEEE, 1998, pp. 756–761.

[156] T. Fujii and T. Ura, "Development of motion control system for auv using neural nets," in *Autonomous Underwater Vehicle Technology, 1990. AUV'90., Proceedings of the (1990) Symposium on*. IEEE, 1990, pp. 81–86.

[157] D. E. Rumelhart, J. L. McClelland, P. R. Group *et al.*, *Parallel distributed processing*. IEEE, 1988, vol. 1.

[158] J. Yuh, "Learning control for underwater robotic vehicles," *IEEE Control Systems*, vol. 14, no. 2, pp. 39–46, 1994.

[159] G. Xia, X. Shi, M. Fu, H. Wang, and X. Bian, "Design of dynamic positioning systems using hybrid cmac-based pid controller for a ship," in *IEEE International Conference Mechatronics and Automation, 2005*, vol. 2. IEEE, 2005, pp. 825–830.

[160] L. Liu, D. Wang, and Z. Peng, "Direct and composite iterative neural control for cooperative dynamic positioning of marine surface vessels," *Nonlinear Dynamics*, vol. 81, no. 3, pp. 1315–1328, 2015.

[161] M. Chen, S. S. Ge, B. V. E. How, and Y. S. Choo, "Robust adaptive position mooring control for marine vessels," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 2, pp. 395–409, 2013.

[162] S.-L. Dai, C. Wang, and F. Luo, "Learning control of uncertain ocean surface ship dynamics using neural networks," in *2011 IEEE 5th International Conference on Cybernetics and Intelligent Systems (CIS)*. IEEE, 2011, pp. 380–385.

[163] A. Healey, "A neural network approach to failure diagnostics for underwater vehicles," in *Autonomous Underwater Vehicle Technology, 1992. AUV'92., Proceedings of the 1992 Symposium on*. IEEE, 1992, pp. 131–134.

[164] Y. Xia and J. Wang, "A general projection neural network for solving optimization and related problems," in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 3. IEEE, 2003, pp. 2334–2339.

[165] Y. Pan and J. Wang, "Model predictive control for nonlinear affine systems based on the simplified dual neural network," in *2009 IEEE Control Applications,(CCA) & Intelligent Control,(ISIC)*. IEEE, 2009, pp. 683–688.

[166] L. Li, L. Ma, and K. Khorasani, "A dynamic recurrent neural network fault diagnosis and isolation architecture for satellites actuator/thruster failures," in *International Symposium on Neural Networks*. Springer, 2005, pp. 574–583.

[167] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[168] A. Cichocki and R. Unbehauen, "Robust estimation of principal components by using neural network learning algorithms," *Electronics Letters*, vol. 29, no. 21, pp. 1869–1870, 1993.

[169] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 5, pp. 554–562, 1988.

[170] Y. Xia, "A new neural network for solving linear and quadratic programming problems," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1544–1548, 1996.

[171] Y. Pan and J. Wang, "A neurodynamic optimization approach to nonlinear model predictive control," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1597–1602.

[172] Y. Shan, Z. Yan, and J. Wang, "Model predictive control of underwater gliders based on a one-layer recurrent neural network," in *Advanced Computational Intelligence (ICACI), 2013 Sixth International Conference on*. IEEE, 2013, pp. 328–333.

[173] Y.-j. PANG, S.-j. FANG, and L.-r. WANG, "Thruster fault diagnosis of autonomous underwater vehicle based on least disturbance wavelet neural network [j]," *Ship Building of China*, vol. 2, p. 015, 2008.

[174] J. MacGregor and A. Cinar, "Monitoring, fault diagnosis, fault-tolerant control and optimization: Data driven methods," *Computers & Chemical Engineering*, vol. 47, pp. 111–120, 2012.

[175] R. Lim and D. Mba, "Fault detection and remaining useful life estimation using switching kalman filters," in *Engineering Asset Management-Systems, Professional Practices and Certification*. Springer, 2015, pp. 53–64.

[176] E. S. Tehrani, K. Khorasani, and S. Tafazoli, "Dynamic neural network-based estimator for fault diagnosis in reaction wheel actuator of satellite attitude control system," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4. IEEE, 2005, pp. 2347–2352.

[177] M. Enayat, "cooperative control and fault recovery for network of heterogeneous autonomous underwater vehicles," Master's thesis, Concordia University, May 2015.

[178] J. Sasiadek, "Guidance and control of ocean vehicles, thor i. fossen," 1999.

[179] X. Liang, B. Wang, L. Wan, and Y. Pang, *Dynamic modelling and motion control for underwater vehicles with fins*. INTECH Open Access Publisher, 2009.

[180] T. Prestero, "Development of a six-degree of freedom simulation model for the remus autonomous underwater vehicle," in *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, vol. 1. IEEE, 2001, pp. 450–455.

[181] R. Hare, "Depth and position error budgets for mulitbeam echosounding," *The International Hydrographic Review*, vol. 72, no. 2, 2015.

[182] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part i: Quantitative model-based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 293–311, 2003.

[183] S. Bhonsale, M. Vallerio, D. Telen, D. Vercammen, F. Logist, and J. Van Impe, "Solace: An open source package for nolinear model predictive control and state estimation for (bio) chemical processes," in *Proceedings of the 26th European Symposium on Computer Aided Process Engineering. Portoroz, Slovenia., June 12th-15th*, 2016.

[184] D. Morrell, "Extended kalman filter lecture notes."

[185] P. S. Maybeck, *Stochastic models, estimation, and control*. Academic press, 1982, vol. 3.

[186] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[187] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: theory and practicea survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[188] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[189] ——, *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[190] D. Kinderlehrer and G. Stampacchia, *An introduction to variational inequalities and their applications*. Siam, 1980, vol. 31.

[191] Y. Xia, G. Feng, and J. Wang, "A recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equations," *Neural Networks*, vol. 17, no. 7, pp. 1003–1015, 2004.

[192] D. Kinderlehrer and G. Stampacchia, *An introduction to variational inequalities and their applications*. Siam, 1980, vol. 31.

[193] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

[194] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[195] Y. Shoukry, M. El-Shafie, and S. Hammad, "Networked embedded generalized predictive controller for an active suspension system," in *American Control Conference (ACC), 2010*. IEEE, 2010, pp. 4570–4575.

[196] A. Dutta, R. De Keyser, Y. Zhong, B. Wyns, G. Pinte, and J. Stoev, "Robust predictive control of a wet-clutch using evolutionary algorithm optimized engagement profile," in *System Theory, Control, and Computing (ICSTCC), 2011 15th International Conference on*. IEEE, 2011, pp. 1–6.

[197] C. Shen, Y. Shi, and B. Buckham, "Integrated path planning and tracking control of an auv: A unified receding horizon optimization approach," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 3, pp. 1163–1173, 2017.

[198] A. T. Nelson, "Nonlinear estimation and modeling of noisy time-series by dual kalman filtering methods," Ph.D. dissertation, 2000.