Sveučilište
u Splitu
University
of Split

FACULTY OF MARITIME STUDIES

Nur Assani

DEVELOPMENT OF AN ANN-BASED DIGITAL TWIN FOR SHIP CONTROL SYSTEM DESIGN AND TUNING

DOCTORAL DISSERTATION

Split, 2026

Sveučilište
u Splitu
University
of Split

FACULTY OF MARITIME STUDIES

Nur Assani

DEVELOPMENT OF AN ANN-BASED DIGITAL TWIN FOR SHIP CONTROL SYSTEM DESIGN AND TUNING

DOCTORAL DISSERTATION

Supervisor: Petar Matić, PhD

Split, 2026

IMPRESSUM

This doctoral dissertation was submitted to the University of Split, Faculty of Maritime Studies, in partial fulfilment of the requirements for the academic degree of Doctor of Science (PhD) in the scientific field of technical sciences, the scientific field of Traffic and Transport Technology.

The dissertation has been written in British English.

Supervisor: Associate professor, Petar Matić, PhD

Faculty of Maritime Studies, Department of Marine Electrical Engineering and Information Technologies, University of Split, Croatia

COMMITTEES FOR EVALUATION AND DEFENCE OF THE DOCTORAL DISSERTATION

Doctoral Dissertation Evaluation Committee:

1. Chair: Professor emeritus Danko Kezić, PhD, Department of Marine Electrical Engineering and Information Technologies, University of Split, Croatia
2. Member: Associate Professor Michele Martelli, PhD, Department of Naval Architecture, Electric, Electronic and Telecommunication Engineering, University of Genoa, Italy
3. Member: Assistant Professor Ivan Pavić, PhD, Department of Marine Electrical Engineering and Information Technologies, University of Split, Croatia

Doctoral Dissertation Defence Committee:

1. Chair: Professor emeritus Danko Kezić, PhD, Department of Marine Electrical Engineering and Information Technologies, University of Split, Croatia
2. Member: Associate Professor Michele Martelli, PhD, Department of Naval Architecture, Electric, Electronic and Telecommunication Engineering, University of Genoa, Italy
3. Member: Assistant Professor Ivan Pavić, PhD, Department of Marine Electrical Engineering and Information Technologies, University of Split, Croatia

Date of defence: DD/MM/YYYY

ACKNOWLEDGEMENT

This section is intended for the author to acknowledge the support and contributions of supervisors, co-supervisors, collaborators, institutions, colleagues, friends, or family members who assisted during the doctoral research and writing process. The Acknowledgement section may be written after the public defence of the doctoral dissertation, before submitting the final hardbound version. This allows the author to acknowledge individuals or institutions who supported him/her during the final stages of his/her research, preparation of the dissertation and/or the defence process itself.

ABSTRACT

The continuous digitalisation in the maritime industry is reshaping the way ship systems are designed, monitored, and controlled. Within this process, the concept of the Digital Twin (DT) has emerged as an enabling technology. While DTs are increasingly applied for monitoring and maintenance, their methodological foundations for control system design remain insufficiently defined, particularly in shipboard applications where safety, reliability, and operational constraints limit direct experimentation.

This dissertation addresses the development and validation of an Artificial Neural Network (ANN)-based Digital Twin for control system design and proportional-integral-derivative (PID) controller tuning in ship flow processes. The research objective is to determine whether a data-driven DT, developed from real system measurements, can enable remote controller tuning with performance comparable to direct experimental methods, while clarifying the conceptual role of the DT and evaluating suitable modelling approaches.

A structured development framework integrates data acquisition, system identification, model validation, and controller tuning. Two modelling approaches are analysed: classical transfer function models and nonlinear autoregressive models with exogenous inputs (NARX) implemented using Artificial Neural Networks. Experimental research is conducted on two laboratory-scale flow control systems representative of shipboard fluid-transfer processes, and the results are validated on the corresponding physical systems.

The findings show that, although transfer function models achieve satisfactory identification accuracy, their suitability for controller tuning within a DT framework is limited. In contrast, ANN-based NARX models more effectively capture nonlinear system dynamics and enable controller tuning with performance comparable to direct experimental procedures. The approach is extended from a single-variable to a multivariable system, confirming its scalability and robustness.

The research establishes and experimentally validates a methodological framework for ANN-based DTs in maritime control applications, supporting controller design in operational environments and contributing to data-driven ship automation.

Keywords: system identification, digital twin, NARX model, Artificial Neural Network, control system design, data-driven modelling, ship systems

CONTENTS

1. INTRODUCTION

Although the concept of a Digital Twin (DT) is increasingly present in the literature, the term is still ambiguously defined, particularly when distinguishing DTs from ordinary simulation models. In some research papers, DTs are associated with cyber-physical systems (CPS) [1], [2], [3], [4], [5]. In others, they are considered a part of CPS [6], while in others, they are described as a combination of a digital model and a digital shadow [7]. This lack of consensus creates conceptual uncertainty, which complicates the development and implementation of DTs across engineering fields.

Flow processes are essential for the transport of fluids such as oil, gas, and water, and they are commonly regulated by PID controllers acting on control valves as actuators. When controllers require readjustment due to malfunction, retrofitting, or system changes, onboard tuning is necessary but poses risks, costs, and practical challenges. For this reason, system models are used as convenient alternatives, and when real-time updates are required, the DT concept becomes particularly valuable. This dissertation adopts a ship's flow control system as a case study to demonstrate and evaluate DT-based approaches for controller tuning.

System identification is central to this research, as it provides data-driven models that approximate the behaviour of real processes. Although such models do not provide physical insight, they reliably describe input-output relations and are therefore well suited for controller design and prediction [8]. Since DTs also rely on continuous measurements to update their models, often in real time, identification methods form the backbone of any DT implementation. Existing research demonstrates a variety of identification methods These methods range from transfer function (TF) models [9], [10], [11] to Artificial Neural Networks (ANN) [12], nonlinear autoregressive exogenous (NARX) models [13], first-order plus time-delay (FOPTD) models [14], fractional-order modelling (FOMCON) [15], [16], and linear parameter-varying (LPV) models [17], [18], [19]. However, direct comparisons between these methods are rarely conducted in the literature, which highlight the significance and motivation of this thesis. It must be noted that no clear guidelines exist for systematic use of these methods within DT development. Therefore, in this thesis modelling methods are selected based on an assumption.

Preliminary research confirmed that TF-based models can achieve high identification accuracy for flow processes [20]. However, when applied to controller tuning, the performance of TF models proved insufficient, highlighting their limited applicability for DT-based control [21]. In contrast, ANN-based models, particularly NARX architectures, demonstrated the ability to capture nonlinear system dynamics more effectively, offering higher accuracy [22]. These findings suggested that ANN-based DTs may be better suited for practical controller tuning tasks in ship systems.

Based on these considerations, this dissertation addresses two key challenges. The first is the conceptual clarification of the DT, ensuring that its role, boundaries, and application in maritime systems are well defined. The second is the methodological evaluation of modelling approaches, comparing TF and ANN-based DTs in terms of their suitability for PID controller tuning. A flow control system is chosen as the experimental platform, as it provides a controlled yet representative environment for validating the research hypothesis.

1.1. DEFINITION OF THE PROBLEM, SUBJECT OF RESEARCH

In maritime transport, the introduction of automation and innovative technologies creates new challenges that require scientific research to ensure safety, efficiency, and reliability. One of these challenges is the implementation of Digital Twins (DTs), whose conceptual and methodological framework remains insufficiently defined, particularly in the context of ship systems [23], [24].

While transfer function (TF) models are widely applied in system identification and can achieve high accuracy [9], [10], [11], they do not reliably support controller tuning, since identification accuracy alone does not guarantee satisfactory tuning performance [21]. This limitation reduces their suitability as the basis for DT development in maritime applications. At the same time, Artificial Neural Networks (ANNs), particularly nonlinear autoregressive models with exogenous input (NARX), offer the potential for more robust and flexible modelling [12], [13], [22], [25], but their applicability for Digital Twin-based controller tuning has not been sufficiently validated.

The subject of this dissertation is the development and evaluation of ANN-based Digital Twins for PID controller tuning in ship systems. The focus is on determining whether ANN-based models can provide a practical and reliable alternative to direct onboard tuning, which is often costly, time-consuming, and risky.

The subject of research can be further specified into the following essential determinants:

- Investigate the conceptual foundations of Digital Twins in maritime systems;
- Compare transfer function and ANN-based approaches for modelling dynamic processes;

- Determine the applicability of both approaches for PID controller tuning on a simpler case study system;
- Investigate and analyse the ability of ANN-based Digital Twins to replicate real system behaviour and enable controller tuning with satisfactory performance on a more complex system.

A two laboratory-scale flow control processes serve as the research object. These processes reflect typical shipboard operations, such as ballast, fuel, and water transfer, while providing a controlled environment for developing, testing, and validating both TF and ANN-based Digital Twin models.

1.2. RESEARCH HYPOTHESIS

Based on the exposed problem, subject, and object of the research, the following working hypothesis is:

A digital twin (DT) developed using real system measurements enables remote controller tuning with performance comparable to experimental methods, offering a practical solution for systems where direct tuning is challenging, such as in-service ships.

Based on such a working hypothesis, the auxiliary hypotheses are formulated:

- AH 1: An alternative modelling approach that overcomes the limitations of transfer function methods can be identified and tested for ship systems, enabling more effective implementation of digital twin concepts in maritime applications.
- AH 2: The alternative modelling approach validated on the flow control unit in proof-of-concept phase can be effectively applied to a larger and more complex flow control system, demonstrating its capability to support digital twin development for control system design in more complex ship processes.

The limitations that are placed before this working hypothesis and auxiliary hypothesis are:

- The hypotheses are tested on a laboratory-scale flow control system, used as a representative case study;

- Only transfer function and ANN (NARX) models are compared, while other modelling approaches are not within the scope;
- Validation is limited to flow processes, although results are expected to be generalisable to more complex ship systems in future research.

1.3. THE RESEARCH PURPOSE AND OBJECTIVES

The purpose of this dissertation is to evaluate, demonstrate, and validate the applicability of Artificial Neural Network-based Digital Twins for PID controller tuning in ship systems, establishing their role as safe, cost-effective, and practical alternatives to direct onboard tuning.

In line with this purpose, the following objectives are defined:

- Clarify and define the Digital Twin concept, with emphasis on maritime systems, and distinguish it from related terms such as simulations and cyber-physical systems;
- Investigate transfer function modelling approaches for ship flow process identification and assess their performance in terms of accuracy and reliability;
- Develop Artificial Neural Network models, particularly NARX architectures, to capture nonlinear dynamics of flow processes;
- Validate and compare the performance of TF and ANN-based models in controller tuning, considering accuracy, robustness, and practical applicability;
- Confirm experimentally the effectiveness of ANN-based Digital Twins by comparing tuning results with those obtained on the real laboratory system;
- Analyse broader applications of ANNs in ship systems, highlighting their potential role in advancing Ship's Digital Twins and supporting the transition toward higher levels of automation;
- Formulate guidelines for the development and application of ANN-based Digital Twins in maritime systems, providing a structured contribution to the field;
- Confirm the effectiveness of the proposed approach in proof-of-concept phase on a more complex flow control system.

2. LITERATURE REVIEW

The purpose of this chapter is to provide a comprehensive overview of the theoretical and technological foundations relevant to this study. It synthesizes existing research to put the proposed work within the scientific context and to highlight the gaps that motivate further research. The chapter includes investigating the concept of Digital Twins in the maritime environment, focusing on their definition, communication mechanisms, and current applications. Furthermore, it includes investigating Artificial Neural Networks in maritime systems, including their architectures, activation functions, training methods, and identified challenges. Building on this, the discussion turns to system identification and modelling approaches, comparing traditional modelling techniques with hybrid and data-driven frameworks. Finally, the chapter outlines performance evaluation metrics used in the assessment of maritime system models and concludes with a summary of key insights and research gaps identified in the reviewed literature.

2.1. DIGITAL TWINS IN THE MARITIME CONTEXT

This section explores the role of Digital Twins (DTs) within the maritime domain and establishes the conceptual basis for their application in this study. Although the idea of DTs originated in manufacturing and aerospace engineering [26], recent technological advances and increasing availability of operational data have enabled their adoption in maritime systems.

The section first outlines the conceptual foundations and definitions of Digital Twins to clarify their distinguishing characteristics relative to conventional simulation models or digital shadows. It then examines the communication mechanisms and interoperability challenges associated with integrating maritime systems, which often combine sensors, vessel subsystems, and external services. An overview of existing Digital Twin applications in maritime practice is presented, with emphasis on modelling, operational monitoring, and performance assessment. Finally, the section reviews published research analyses and identifies unresolved challenges and gaps in current Digital Twin implementation, forming a basis for research performed in this thesis.

2.1.1. Conceptual foundations and definition

The concept of the Digital Twin (DT) emerged in product-lifecycle management and advanced manufacturing, originally introduced in [27] and further developed in [24]. The DT represents a virtual counterpart of a physical system that remains synchronised with the real asset throughout its operational life. It enables the integration of sensing, data analytics, simulation, and feedback mechanisms, providing a basis for monitoring, optimisation, and decision support.

In essence, a digital twin is a model of a physical entity, connected by bidirectional data exchange that allows information and knowledge to flow between the two domains. This continuous interaction distinguishes a digital twin from conventional simulation models or static digital replicas, which typically operate independently of real-time data. Maybe the closest relative to DT can be found in traditional cyber-physical systems (CPS) that integrate sensing, computation, and control. The DT extends this idea further by embedding a high-fidelity behavioural model that evolves with the physical system and provides predictive insights [28]. The relationship between the DT and CPS can be interpreted hierarchically: a CPS provides the enabling infrastructure of sensors, networks, and control loops, while the SDT constitutes the higher-level analytical and predictive layer that leverages these data streams to generate knowledge and optimisation strategies [28], [29].

The DT paradigm was later generalised across multiple industrial sectors, including aerospace, energy, health care, and, more recently, the maritime domain [30], [31]. In the maritime context, the Ship's Digital Twin (SDT) can be understood as a digital representation of a ship or ship subsystem that continuously incorporates operational and environmental data to provide actionable information for decision-making and control, as defined in [23]:

"A simulation of a ship's system that uses the best available models and sensor information as input data to mirror and predict activities or performances of its corresponding physical twin."

This definition captures the key elements of an SDT: continuous data flow, model synchronisation, and the ability to simulate or predict system behaviour under changing conditions.

By integrating historical and real-time data, the SDT can support condition monitoring, predictive maintenance, and performance optimisation across various ship systems.

The bidirectional flow of information between the physical and digital domains ensures that the SDT remains an accurate reflection of the ship's real-time operating condition.

One of the defining features of the SDT compared to conventional ship simulation models is its data-driven adaptability. In contrast to static design-phase simulations, which are built on assumed boundary conditions, the SDT dynamically updates its parameters using measurements from onboard instrumentation [31]. This makes it suitable for in-operation performance evaluation, controller tuning, and decision-support tasks where high fidelity and adaptability are required.

The DT framework in maritime engineering aligns with the general DT lifecycle proposed by Tao et al. (2019), consisting of:

1. physical entity,
2. virtual model, and
3. connections (data and services) between them.

In maritime systems, this lifecycle includes the design stage, where the initial simulation model is created, the operational stage, where live data are used to update and validate the model, and the feedback stage, where model results are used for control and decision-making.

From a systems-engineering perspective, the SDT embodies three essential characteristics [23]:

1. connectivity, ensuring continuous and reliable data exchange,
2. integration, combining heterogeneous sources of information, and
3. intelligence, leveraging models and analytics to transform data into insight.

Together, these features enable the SDT to act as both a digital observer and a digital controller of the ship, thereby providing the foundation for higher levels of automation and autonomy.

2.1.2. Communication and interoperability

Efficient communication and interoperability represent fundamental enablers for the successful implementation of an SDT. Without data flow between the physical and virtual environments, it is impossible to maintain an accurate and continuously updated representation of the ship's operational state [23]. The data connection between the physical vessel and its digital counterpart allows the twin to process, analyse, and predict system behaviour. Therefore, the communication architecture and interoperability standards directly determine the fidelity, reliability, and scalability of an SDT.

The exchange of data within an SDT environment relies on shipboard and shore-based infrastructure that supports continuous acquisition, transfer, storage, and utilisation of information.

Two main data transfer strategies are identified in maritime applications: batch transfer, where operational data are collected on board and transmitted periodically, e.g., after a voyage or operational phase, and real-time streaming, where data are continuously synchronised with remote servers or cloud-based simulation platforms [32]. Both modes can coexist depending on the vessel's communication bandwidth and operational requirements.

In terms of how closely the digital twin is coupled with its physical twin, six principal digital twin topologies were proposed by [33]. These include:

1. Disconnected twin – A twin that operates independently of the physical twin, typically used when real-time connectivity is technically or economically infeasible.
2. Connected twin – maintains a continuous link with the physical twin, ensuring real-time data exchange.
3. Embedded twin – exists as a part of the physical twin itself – residing with onboard control or monitoring units.
4. Aggregated twin – integrates several independent digital twins into a unified model.
5. Multi-device twin – represents a single virtual entity connected to multiple physical systems.
6. Combined twin – incorporates elements of several of the aforementioned configurations.

Within the ship, data are collected from a wide range of heterogeneous subsystems, including propulsion, power generation, navigation, environmental monitoring, and automation systems. Communication takes place via a set of well-established industrial protocols and fieldbuses, such as CAN-bus, Modbus, Ethernet, and RS-485/RS-232, which are commonly employed for sensor-level integration [23]. Navigation and bridge systems typically use the NMEA 0183/2000 standard, while higher-level communication across different vendors and systems is increasingly realised through Open Platform Communications Unified Architecture (OPC UA) and Message Queuing Telemetry Transport (MQTT), which implement flexible publish-subscribe architectures and support semantic data exchange [29], [34], [35].

These protocols facilitate data sharing among subsystems with different data formats, sample rates, and update frequencies. To ensure coherence between sources, data must be time synchronised, commonly through Network Time Protocol (NTP) or Precision Time Protocol (PTP), and tagged with metadata describing measurement units, coordinate systems, and sensor identifiers [33]. Properly synchronised and semantically consistent data are essential for the integrity of subsequent model-based analyses and real-time decision-making.

A major technical challenge in SDT development is the integration of models and simulation tools from different vendors, software environments, and physical domains. This requirement has led to the introduction of Functional Mock-up Interface (FMI) and Functional Mock-up Units (FMU), which define a neutral, tool-independent standard for model exchange and co-simulation [36]. FMI allows subsystem models such as engine, propeller, electrical network, or control algorithms to be wrapped as FMUs and coupled dynamically within a single execution environment. The maritime sector has actively adopted this standard through the Open Simulation Platform (OSP), an industrial collaboration aimed at standardising simulation interfaces for ship system modelling [37].

OSP enables the co-simulation of heterogeneous models by linking FMUs representing various ship components through a shared communication bus. This approach not only enhances interoperability but also protects intellectual property, since subsystem providers can share compiled FMUs without exposing proprietary source code.

By applying FMI/FMU and OSP, engineers can build modular SDTs that integrate propulsion, power, and control models into a single, cohesive virtual environment [23].

High-quality data are a prerequisite for an accurate SDT. Raw sensor signals are subject to noise, drift, outliers, and missing values, hence, data pre-processing is an essential step before the data are used for training or validation of models [29], [35].

Typical pre-processing techniques include:

1. filtering and smoothing to reduce high-frequency noise,
2. outlier detection and removal using statistical or clustering-based methods,
3. resampling and interpolation to produce uniform time-series data,
4. feature extraction (RMS, kurtosis, skewness, spectral energy) to enhance model interpretability,
5. data fusion from redundant or complementary sensors to improve robustness.

Poorly filtered or misaligned data can severely degrade the fidelity of the digital twin, leading to inaccurate predictions and unreliable diagnostics [38]. Consequently, the data-preparation process is often as critical as the model design itself.

An SDT can be understood as an integrated system that combines a shipboard sensor network with simulation models. The onboard instrumentation continuously observes the ship's machinery, structural, and operational systems, while the associated simulation environment processes these data to deliver insights to the user, whether on board, at an onshore control centre, or within a cloud-based infrastructure. In this configuration, the SDT and its physical twin are connected through a live data exchange, forming a feedback loop that enables continuous synchronization and operational awareness. The importance of this data connection introduces an additional layer of complexity in SDT development and has been recognized in [39] as one of the key challenges currently limiting the concept's full potential.

For maritime applications, where operational data from the vessel must be transferred continuously and reliably, the connected topology provides the most functional framework. However, practical implementation often requires a balance between connectivity, technical feasibility, and economic cost, meaning that hybrid or partially connected configurations are also common.

To enable communication between heterogeneous systems, SDTs employ various communication protocols and standards organised according to the Open Systems Interconnection (OSI) reference model, shown in Figure 2.1, which defines a seven-layer architecture for networked communication [40], [41].

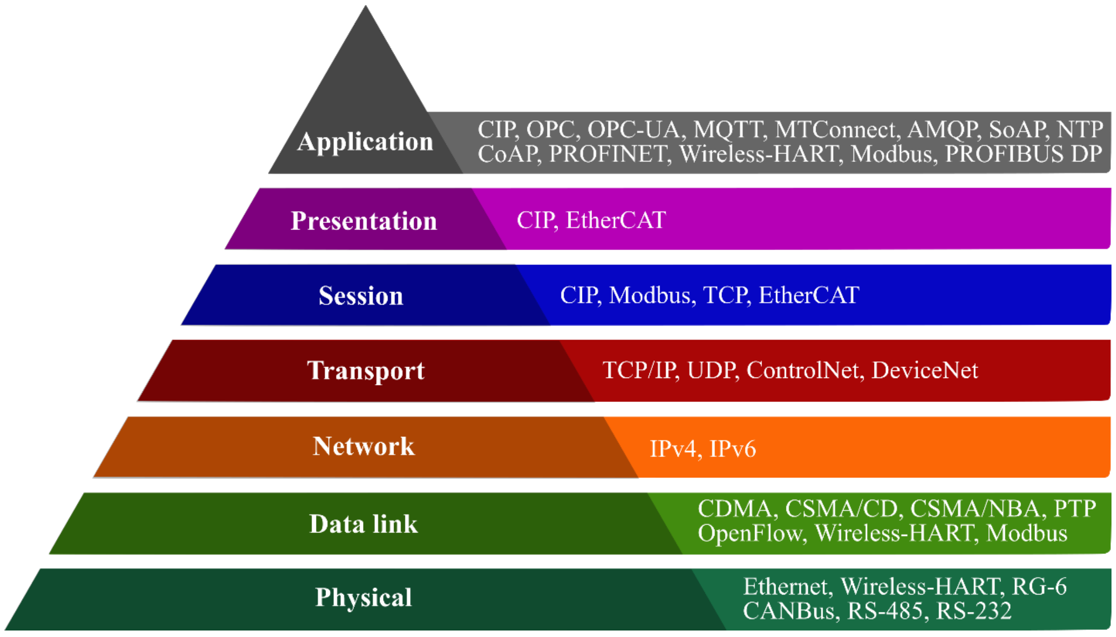

Application
CIP, OPC, OPC-UA, MQTT, MTConnect, AMQP, SoAP, NTP
CoAP, PROFINET, Wireless-HART, Modbus, PROFIBUS DP
Presentation
CIP, EtherCAT
Session
CIP, Modbus, TCP, EtherCAT
Transport
TCP/IP, UDP, ControlNet, DeviceNet
Network
IPv4, IPv6
Data link
CDMA, CSMA/CD, CSMA/NBA, PTP
OpenFlow, Wireless-HART, Modbus
Physical
Ethernet, Wireless-HART, RG-6
CANBus, RS-485, RS-232

Figure 2.1 Communication layers and protocols according to the Open Systems Interconnection (OSI) reference model

Each layer in this model has a distinct role: the physical layer specifies the transmission medium and hardware characteristics, the data-link layer defines the format for data transfer across the link, the network layer handles packet addressing and routing, the transport layer manages error control and data integrity, the session layer establishes and terminates communication sessions, the presentation layer ensures that data are properly formatted and encoded for interpretation, and finally, the application layer provides user-facing services and defines how data are presented to applications and operators.

Commonly used physical-layer technologies in maritime SDTs include Ethernet, WirelessHART, RG-6 coaxial, CAN-bus, RS-485, and RS-232 interfaces [35].

At the data-link layer, standards such as Code-Division Multiple Access (CDMA), Carrier-Sense Multiple Access Collision Detection (CSMA/CD), Carrier-Sense Multiple Access Non-Blocking Arbitration (CSMA/NBA), Modbus, OpenFlow, Precision Time Protocol (PTP), and WirelessHART enable reliable local communication as explained in [35].

The network layer relies primarily on IPv4 and IPv6 addressing [29], while the transport layer uses Transmission Control Protocol/Internet Protocol (TCP/IP), User Datagram Protocol (UDP), ControlNet, and DeviceNet protocols to ensure robust data transfer.

At the session layer, common protocols include Common Industrial Protocol (CIP), Modbus TCP [5], and EtherCAT.

The presentation layer may again utilise CIP and EtherCAT for data formatting, whereas the application layer hosts a variety of high-level industrial communication standards, including the CIP family, OPC and OPC Unified Architecture (UA), Message Queuing Telemetry Transport (MQTT) [35], [42], MTConnect [33], [43], Advanced Message Queuing Protocol (AMQP) [35], Simple Object Access Protocol (SOAP), Constrained Application Protocol (CoAP) [34], [35], Network Time Protocol (NTP), WirelessHART, Modbus [42], PROFIBUS Decentralised Peripherals (DP), and PROFINET.

Together, these technologies establish a multi-layered communication infrastructure that ensures interoperability between sensors, controllers, and simulation environments. The correct selection and configuration of protocols across OSI layers are therefore essential for achieving reliable, secure, and low-latency communication between the physical ship and its digital twin. This architecture supports scalability and modularity, allowing new sensors or subsystems to be integrated with minimal reconfiguration.

Table 2.1 summarises representative shipboard data sources, sampling rates, and typical pre-processing approaches. Modern SDT research recognises that standardisation of data models, interfaces, and synchronisation protocols is essential for interoperability across platforms and organisations [29], [33].

Table 2.1 Representative shipboard data channels and pre-processing methods

Subsystem	**Typical Variables**	**Sampling Frequency**	**Pre-processing Methods**	**Example Application**
Propulsion/Engine	Torque, RPM, fuel index, exhaust temperature	1–10 Hz	Low-pass filtering, outlier removal, trend extraction	Engine-performance monitoring
Electrical/Power	Voltage, current, active/reactive power, frequency	1 Hz	RMS computation, harmonics analysis, spectral features	Power-quality assessment
Hull/Structure	Strain, acceleration, vibration	10–100 Hz	Filtering, FFT, wavelet transform, peak detection	Structural-health monitoring
Navigation	Speed, heading, position, draft	0.1–1 Hz	Interpolation, time alignment, coordinate transformation	Voyage optimisation
Environment	Wind, wave height, temperature	0.1–1 Hz	Outlier removal, moving-average smoothing	Weather correction in performance models

Standards such as FMI and initiatives like OSP promote modular co-simulation, while communication standards such as OPC UA and MQTT enable cross-domain data sharing. Together, these technologies form the backbone of an open and extensible SDT ecosystem.

In summary, communication and interoperability determine how effectively a digital twin can integrate data from diverse ship systems and maintain coherence across distributed models. By adopting common protocols, enforcing rigorous data quality procedures, and employing model-exchange standards, SDTs achieve reliable synchronisation and robust operation across the ship's lifecycle.

2.1.3. Applications of SDT

The implementation of SDTs across maritime systems has expanded rapidly in recent years, primarily due to advances in sensing technologies, high-speed connectivity, and simulation tools. The reviewed literature demonstrates that SDTs span across a broad range of domains. Based on the reviewed papers, these applications can be grouped into five key categories, including condition based and predictive maintenance as in [44], [45], decision-making support as in [39], [45], [46], real-time condition monitoring of onboard systems as in

[47], [48], [49], and cost-reduction strategies as in [37]. Other identified applications include simulation and testing of vessel performance as in [50], integration testing of shipboard subsystems and automation architectures as in [1], crew training and operational familiarization as in [51], as well as overall performance optimisation as in [52], [53], [54].

2.1.3.1. Condition based and predictive maintenance

Condition-based maintenance (CBM) holds a central role in modern maritime operations, as it enables the prevention of unexpected system failures caused by gradual wear, stress fatigue, or other degradation mechanisms that evolve over time. Such failures can lead to major economic losses or even safety-critical incidents. The use of DT technology for CBM has therefore gained particular attention in the maritime sector.

A notable illustration of its potential is provided in [55], which analysed the Viking Sky cruise-ship blackout. According to the study, the accident triggered by a loss of lubricating-oil pressure in the main diesel generators during rough seas and low oil levels could have been anticipated through a properly implemented SDT. The resulting engine shutdown caused a total power loss and propulsion blackout, leading to a near-grounding and an emergency evacuation of passengers. The authors argue that a DT-based simulation framework could have identified such fault scenarios in advance, allowing preventive measures to be taken.

Another relevant contribution was presented in [44], where the influence of propeller and hull fouling on ship speed was evaluated. The fouling process progresses relatively quickly and can have a measurable impact on propulsion efficiency within only a few months, even though typical dry-docking intervals are four to five years. The findings showed a clear positive correlation between hull and propeller condition and vessel-speed loss. Poor surface conditions were estimated to reduce the global fleet’s overall energy efficiency by approximately 9-12 %, underscoring the importance of continuous performance monitoring for maintenance optimisation.

Since DTs provide data acquisition and predictive capability, they allow maintenance activities to be scheduled dynamically, based on real-time performance degradation rather than fixed intervals. Recorded data can also be used post-factum to evaluate maintenance effectiveness by comparing system condition before and after intervention.

Beyond machinery systems, DT frameworks can simulate and predict environmental effects such as wind, waves, and currents on hull structures. For instance, [48] presented an analysis of fatigue damage accumulation on a cruise ship based on data collected over 970 days of operation. The computed fatigue-damage indices were then used to estimate local structural deterioration and residual life of balcony structures exposed to cyclic loading.

Further research has applied DT concepts to various subsystems. In [56], a DT of a ship drivetrain was developed for vibration analysis and maintenance planning. The study observed that higher model fidelity led to reduced simulation accuracy, suggesting that low-fidelity models may be insufficient for reliable DT applications. Structural monitoring studies, such as [57], focused on identifying critical hull areas for strain measurement and optimising sensor placement to capture relevant structural responses.

A digital twin of a ship's electric power system was proposed in [58] for offline maintenance simulation and subsystem-upgrade assessment. The work explored requirements for model-based design (MBD) and assessed the feasibility of implementing hardware-in-the-loop (HIL) testing within such frameworks.

Similarly, [53] reported a full-scale experimental implementation, where sensors were installed on a vessel to collect operational data for DT development. The authors emphasised the importance of integrating real-time data acquisition, processing, and transmission infrastructure to enable noticeable benefits such as optimised maintenance intervals, remote assistance, stress analysis, and enhanced navigation safety.

A further example is provided in [49], which proposed a DT model of a ship synchronous generator for condition monitoring and identification of mechanical defects. Through transient-state analysis, the study detected characteristic anomalies in the frequency spectra of forces, currents, and voltages, thus enabling the diagnosis of mechanical malfunctions. Although promising, the method still requires full validation.

System identification as part of DT development was also discussed in [47], where researchers created a partial twin of an experimental autonomous-ship prototype. The DT was based on a 1:20 scale physical model, and the identification algorithms were applied to motion data collected during manoeuvring tests involving sway, surge, and yaw coupling. This study demonstrated how small-scale experimental setups can be used to train and validate DT components before full-scale implementation.

2.1.3.2. Decision-making support

Ship design and construction depend on expertise drawn from multiple engineering disciplines, which makes the shipbuilding process inherently complex. To manage this complexity, shipyards increasingly rely on computer-aided engineering and simulation tools, forming the foundation for DT development.

At the same time, shipowners are demanding advanced lifecycle management and decision-support solutions that extend beyond initial design. Within this context, DT technology, particularly when applied to power-generation and energy-management systems, offers significant advantages for process optimisation and maintenance planning throughout the vessel's operational life [41].

A number of studies have investigated DT applications aimed specifically at supporting engineering and operational decision-making. In [39], a DT model of a ship's hull structure was proposed to aid decisions related to maintaining structural integrity. Because hull responses evolve slowly and the concept is relatively new, the approach has not yet been experimentally validated, but it demonstrates a promising direction for integrating structural monitoring with digital-twin frameworks.

Further development was presented in [45], which outlined a DT concept for a cruise ship, which is one of the most complex vessel types due to stringent requirements for safety, comfort, entertainment capacity, reliability, and energy efficiency. The proposed cruise-ship twin was designed to deliver real-time decision support and alerting functions, as well as to forecast gradual changes in system behaviour over the vessel's lifetime.

An additional contribution described in [46] introduced a DT application for trajectory prediction and navigation assistance. The system generates a real-time visualisation of the vessel's predicted path, superimposing it on the wheelhouse display to enhance the operator's situational awareness. This capability provides real-time decision support for manoeuvring in constrained or hazardous waters. Although the concept requires further validation, it illustrates the growing potential of digital-twin environments to augment navigational safety and operational decision-making.

2.1.3.3. Cost reduction

In [37], a DT was developed to represent a ship's electrical power and propulsion system, incorporating two diesel generators (DG1 and DG2), a pair of propulsion motors, the fuel-supply network, switchboards, and automatic voltage regulators (AVRs). Simulation experiments were conducted to replicate operational events in a defined sequence, enabling detailed analysis of the vessel's power-system behaviour under varying load conditions.

The results demonstrated that the simulated system exhibited expected dynamic characteristics, such as frequency and voltage variations during load increases or reductions, and accurately reflected the controller response in load sharing between the two generators.

The study further highlighted the benefits of using the Open Simulation Platform (OSP) as the integration framework. OSP allows the connection of subsystem models originating from different engineering disciplines into a unified simulation environment, promoting cross-domain collaboration among experts.

This integrative approach significantly reduces development time and cost by facilitating joint model exchange and verification within a shared standardised platform. Compared with traditional, proprietary simulation tools, OSP offers the additional advantage of being an open-source, online, and standardised ecosystem specifically tailored for the maritime industry, supporting model interoperability and collaborative development.

2.1.3.4. Real-time remote monitoring and control

A study presented in [50] developed a DT of a scaled physical ship model, designed to enable remote control through a digital interface by adjusting operational setpoints in real time. The authors identified four main functions of the proposed DT framework: monitoring system performance, assessing the ability of the dynamic positioning (DP) system to maintain vessel position, predicting the ship's response under future operational conditions, and simulating parameters not directly measurable during normal operation.

The experimental validation of this concept was carried out in the Numerical Offshore Tank at the University of São Paulo. During these trials, sensor measurements obtained from the physical model were transmitted to the DT and used for 3D visualisation of the vessel's motion on an operator interface, thereby demonstrating real-time synchronisation between physical and virtual environments.

Another real-time DT application was described in [42], where researchers proposed a simulation and remote-control centre connected via live data transfer to an operating ship and its onboard crane system. This configuration enabled remote operation, monitoring, and crew support from shore, highlighting the growing potential of DT frameworks for real-time simulation and human-machine collaboration in maritime operations.

2.1.3.5. Various applications

A study presented in [54] investigated the modelling and optimisation of a ship's power system to enhance overall efficiency. The research demonstrated improvements in response time and a reduction in false blackout detections. By optimising both short and long-term generator load distribution, the proposed approach achieved fuel savings of 6-8% and contributed to lower maintenance expenses.

In addition, the propulsion system was fine-tuned to minimise its influence on bus-bar frequency by employing a quasi-static load-limiting controller, which used real-time measurements to predict and compensate for power fluctuations. A complementary power-redistribution control strategy was developed to counteract voltage and frequency deviations affecting thrusters and other high-demand consumers, thereby enhancing grid stability without compromising the vessel's responsiveness.

In [52], the authors introduced an innovative digital twin model for voyage performance assessment, focusing on predicting ship speed and fuel consumption for a bulk carrier. The model utilised Automatic Identification System (AIS) data and noon reports, combined with weather hindcast information, to estimate operational parameters and validate performance. The results confirmed the model's predictive capability and established its potential as a tool for greenhouse gas (GHG) emission reduction, given the direct correlation between ship speed, fuel consumption, and emission levels. Because the modelling framework did not rely on direct sensor inputs, it can be applied to vessels lacking extensive sensor infrastructure, while offering the option for future expansion when real-time data becomes available.

When developing a DT, it is necessary to balance model fidelity and computational performance. To address this trade-off, [59] proposed a marine diesel engine model that achieved an effective compromise between accuracy and real-time capability. The approach outperformed traditional differential-equation models based on the Runge-Kutta method in terms of computational speed, while maintaining low relative error values during hardware-in-the-loop (HIL) validation on a marine-engine test bench. This model thus provides a reliable foundation for anomaly detection, failure-mode analysis, thermal-efficiency tracking, and emission prediction.

The DT concept was further extended in [60], which proposed a method for real-time sensor data reconstruction and error correction within a heat-exchanger system. The approach automatically validated and corrected incoming sensor data, ensuring consistent input quality for model calculations. Although demonstrated for a specific subsystem, the methodology is broadly applicable across various shipboard systems that require dependable sensor data for DT operation.

Finally, [61] explored a DT-based model of a boost converter within the ship's electrical power network. The focus of this study was on refining grey-box parameter-identification techniques, comparing two optimisation algorithms to minimise discrepancies between predicted and measured outputs. The authors recommended further research into global optimisation algorithms to improve parameter fitting and extend model reliability for broader DT applications.

2.1.4. Review analysis

The comprehensive review of published Ship's Digital Twin (SDT) research indicates that the topic has evolved rapidly in recent years, though the literature remains fragmented across disciplines and application domains. Review included fifty-nine publications on DTs indexed in Google Scholar, Web of Science and Scopus databases, out of which nineteen publications were related to SDTs. Table 2.2 summarises the reviewed publications, classifying them by application purpose, modelling method, and targeted ship subsystem.

Table 2.2 Chronologically ordered summary of reviewed publications according to the year of publication

Reference (Author, Year)	Ship subsystem	Methods	Application
Radan, D., 2008, [54]	Ship electric power system	Mathematical modelling	Various
Dufour, C. et al., 2018, [58]	Ship electric power system	MBD, HIL	Maintenance & condition monitoring
Danielsen-Haces, A., 2018, [47]	Electric propulsion system	ANN	Maintenance & condition monitoring
Bekker, A., 2018, [53]	Behaviouristic model of a ship	DDM, Statistics	Maintenance & condition monitoring
Coraddu, A. et al., 2019, [44]	Propulsion system and ship hydrodynamics	ANN, DDM	Maintenance & condition monitoring
Arrichiello, V. et al., 2019, [45]	Hull structure & machinery	MBSE	Decision-making support
Hulkkonen, T. et al., 2019, [48]	Ship structure	FEM	Maintenance & condition monitoring
Johansen, S. et al., 2019, [56]	Ship drivetrain	FEM	Maintenance & condition monitoring
Bondarenko, O. et al., 2020, [59]	Diesel engine	Mathematical modelling, HIL, CMV	Various
Fonseca, I. et al., 2020, [50]	Ship hull & propulsion system behavioural model	BEM	Remote control and monitoring
Manngård, M. et al., 2020,[60]	Heat exchanger of a diesel engine	Mathematical modelling, Statistics, FEM	Various
Liu, M. et al., 2020, [52]	Ship speed and fuel consumption	GBM, ANN	Various

Perabo, F. et al., 2020, [37]	Ship power & propulsion system	Mathematical modelling	Cost reduction
Anyfantis, K. N., 2021, [57]	Ship hull structure	FEM, ANN	Maintenance & condition monitoring
Major, P. et al., 2021, [42]	Ship & ship crane	DDM, HIL	Remote control and monitoring
Galeev, R. E. et al., 2021, [46]	Ship trajectory	Mathematical modelling	Decision-making support
Grinek, A. V., et al., 2021, [49]	Ship synchronous generator	FEM	Maintenance & condition monitoring
Wunderlich, A. et al., 2021, [61]	Boost converter as a part of ship power system	GBM	Various
VanDerHorn, E. et al., 2021, [39]	Ship hull structure	FEM, DDM	Decision-making support

As shown in Figure 2.2, the number of SDT-related papers increased after 2018, coinciding with the broader introduction of cyber-physical systems and the Internet of Things (IoT) into maritime engineering.

Figure 2.3 illustrates the distribution of modelling techniques: mathematical modelling and finite-element methods (FEM) each account for about 22 % of the studies, followed by classical data-driven and artificial-neural-network (ANN) models at 15 % each. Hybrid methods, such as grey-box modelling (GBM) and cycle-mean-value (CMV) modelling, appear less frequently (around 7 % each).

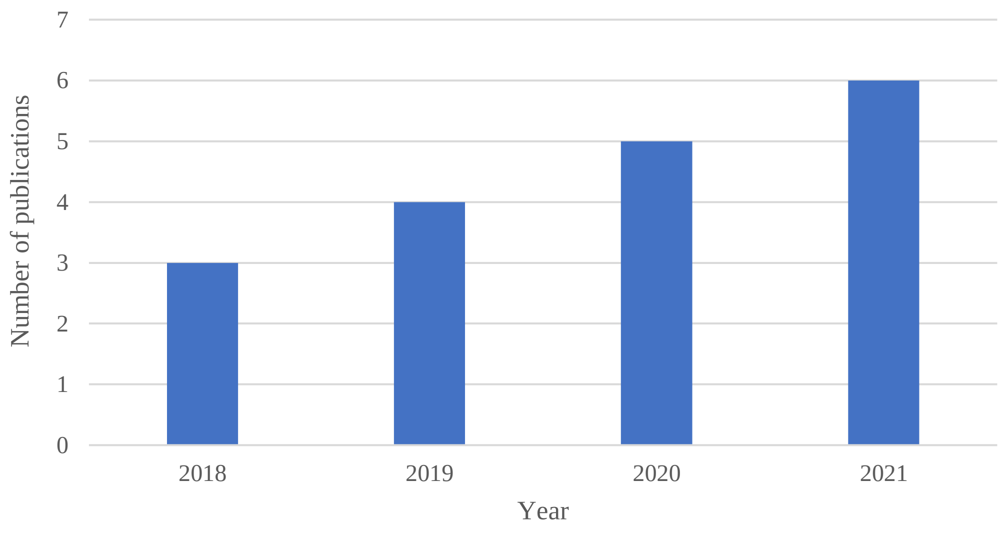


Figure 2.2 SDT related publications in period 2018-2021

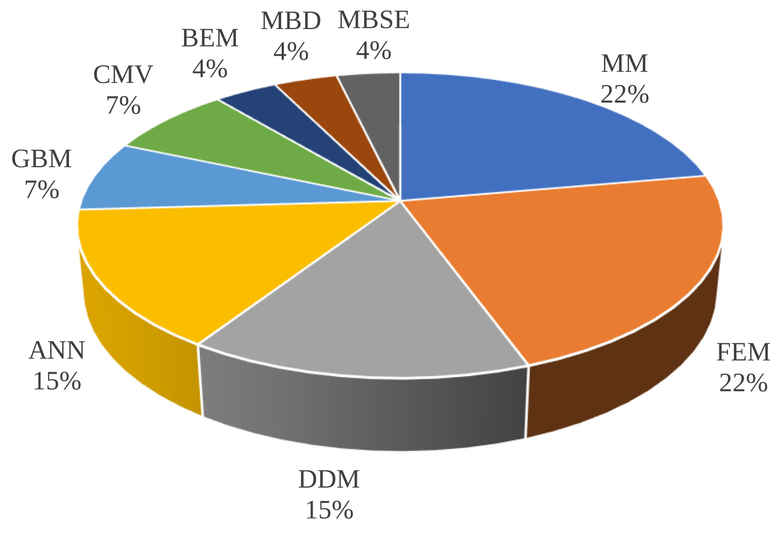


Figure 2.3 Review of modelling methods

The domain distribution in Figure 2.4 shows that research most often targets ship electrical-power systems (23 %), followed by propulsion and hull/structure models (18 % each). Smaller shares focus on ship behaviour (14 %), marine-diesel engines (14 %), trajectory or fuel-consumption prediction (9 %), and hydrodynamics (4 %). Finally, Figure 2.5 summarises application purposes. Maintenance and condition monitoring dominate (42 %), whereas decision support, optimisation, training, and remote-operation functions are less frequently represented.

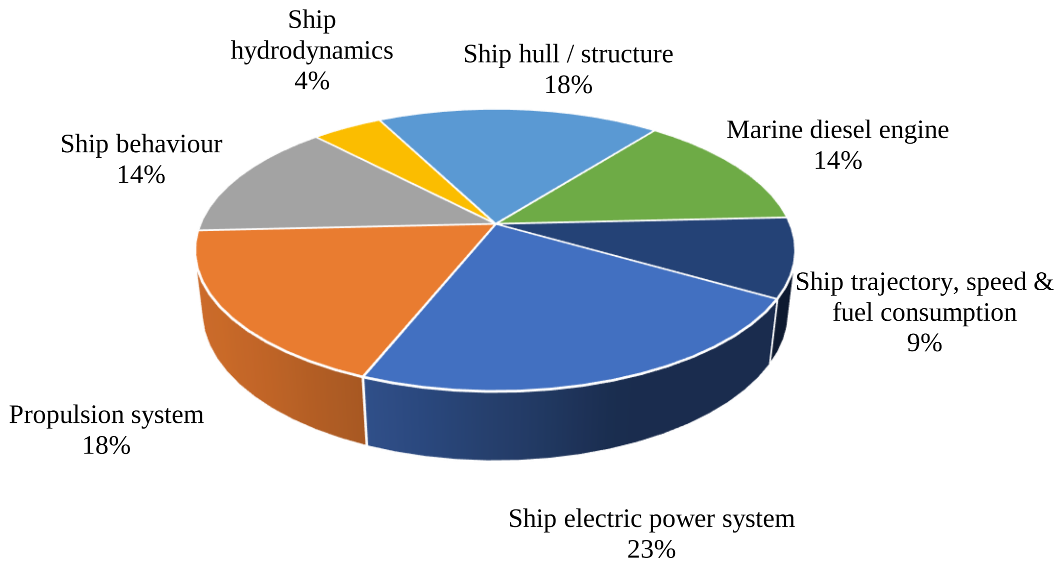


Figure 2.4 Ship's systems modelled in the reviewed literature

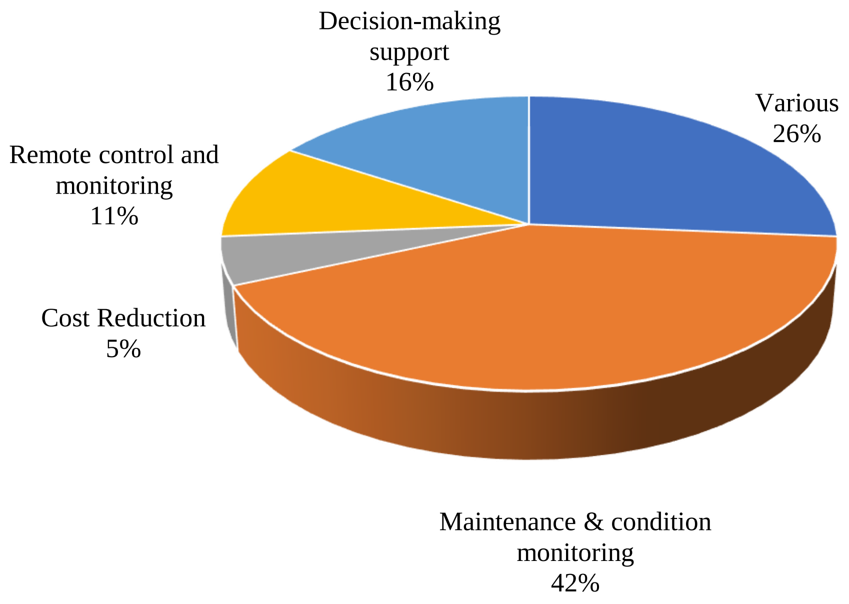


Figure 2.5 Application purposes in the reviewed literature

Together, these data show that SDT adoption in the maritime sector is still in an early phase, with most research concentrating on machinery and power-system applications where high-quality data and established modelling frameworks are already available.

The literature further highlights that SDTs serve as integrated tools for enhancing both individual process control and overall vessel-management strategies. They are increasingly recognised as valuable elements throughout the ship's lifecycle, from design and construction to operation and maintenance. Nevertheless, the reviewed studies also reveal that implementing SDTs involves multi-layered challenges, spanning data acquisition, communication infrastructure, model development, and standardisation.

SDTs are conceptualised as flexible, modular systems that can be extended and refined as new data and methods emerge. This adaptability is vital in the maritime environment, where ships operate under variable conditions and are subject to continual technical upgrades. The capacity of a digital twin to evolve over time ensures that its models remain representative of the actual vessel state and operational profile.

Regarding modelling approaches, the reviewed literature exhibits substantial diversity. Physics-based mathematical and FEM models remain dominant, while data-driven and hybrid strategies, including ANN-based learning, are underutilized. The choice of method remains highly application-specific, depending on data accessibility, computational resources, and the target subsystem.

In terms of communication and interoperability, SDT implementations rely heavily on industrial protocols tested in other engineering domains. The ISO/OSI reference model provides a useful conceptual foundation for classifying these standards. Protocol families such as CIP, EtherCAT, and MTConnect span multiple OSI layers, allowing versatile, distributed data exchange. Yet the literature consistently identifies real-time communication with the physical vessel as one of the principal technical constraints. Ensuring low-latency, high-reliability links between the ship and its digital representation remains a prerequisite for dynamic monitoring and control.

In summary, the reviewed research demonstrates considerable progress in developing SDT concepts and subsystems. However, implementation at the full-ship level is still limited, and methodological inconsistencies persist across studies. The current state of the field can thus be characterised as advancing but fragmented, a foundation upon which more comprehensive, validated, and standardised SDT frameworks can be developed.

2.1.5. Identified research gaps in SDT research

Although the concept of the SDT has achieved considerable attention and early success in specific applications, the reviewed literature reveals several aspects that remain underdeveloped or insufficiently explored.

The data presented in Table 2.2 indicate that most SDT-related studies focus on condition monitoring, condition-based maintenance, and decision-support functions, whereas comparatively fewer address ship-level optimisation, energy-efficiency management, or crew-training applications. Another underrepresented area is the use of SDTs for control-system design. Although digital twins have proven valuable for monitoring, prediction, and maintenance, their potential to support controller development and verification remains largely unexplored in the maritime domain.

A notable gap concerns the flexibility and scalability of SDT architectures. The studies examined suggest that digital-twin implementations are often designed for a single, well-defined purpose and seldom expanded or adapted after initial deployment. A fully mature SDT should instead allow for progressive enhancement, accommodating new data sources, updated models, and changing operational requirements throughout a ship’s service life.

The SDT formulation process itself also needs further investigation. Each stage, i.e., data acquisition, data processing, modelling, and validation presents open questions that remain insufficiently resolved. Data acquisition faces challenges in sensor reliability, calibration, and coverage across distributed systems, data-processing methods must handle high-frequency heterogeneous data streams, modelling approaches must balance fidelity with real-time performance, and validation requires repeatable protocols that link model outputs to measurable operational results. Existing studies tend to treat these stages separately rather than as parts of an integrated iterative loop.

Another gap lies in the integration of hybrid modelling approaches. While physics-based and data-driven methods have been employed individually, only a few studies attempt to combine them in unified frameworks capable of leveraging both physical insight and data adaptability. Research into hybrid modelling, uncertainty quantification, and adaptive learning is needed to enhance predictive accuracy and reliability under varying operational conditions [62].

Real-time communication and interoperability remain persistent obstacles. Even though industrial communication standards, such as CIP, EtherCAT, and MTConnect, provide robust foundations, their implementation within maritime digital-twin systems is inconsistent. Achieving bidirectional communication between the ship and its twin will require harmonised protocol usage, bandwidth optimisation, and redundancy strategies to ensure continuous data flow in challenging marine environments.

Validation is another critical research gap. The review showed that validation is the least implemented step in the SDT-development process. Most existing works rely on simulation-based verification rather than full-scale trials. Developing standardised validation methodologies, shared datasets, and benchmarking procedures would substantially improve the credibility of SDT models and accelerate industry adoption.

In summary, the key research gaps identified include:

1. Lack of concept definition and iterative SDT-development steps.
2. Uneven focus across SDT applications, with limited attention to optimisation, energy efficiency, and control system design.
3. Need for advanced hybrid-modelling and adaptive-learning techniques.
4. Persistent communication and interoperability issues preventing reliable real-time operation.
5. Absence of standardised validation procedures and benchmarking datasets.

2.2. ARTIFICIAL NEURAL NETWORKS IN MARITIME SYSTEMS

Large ships are complex engineering systems that are continuously evolving toward higher levels of automation, ultimately progressing toward fully automated and autonomous operation. These advancements aim to reduce operational costs and improve efficiency through remote control and intelligent management. Achieving these objectives depends heavily on extensive measurement and monitoring systems, as the number of observed variables continues to grow, producing vast quantities of data for analysis, modelling, prediction, control, and decision-making. Data-driven methods, particularly Artificial Neural Networks (ANNs), have consequently found wide application across maritime systems and shipboard processes.

Over time, ANNs have developed into sophisticated and specialised computational tools, supported by numerous network architectures and training algorithms created for different tasks. Simultaneously, advances in measuring equipment and communication systems, particularly in terms of reliability, bandwidth, and data-transfer speed, have further enabled the real-time use of operational data, making ANN-based approaches even more relevant to modern ship systems.

A total of sixty-nine publications indexed in Google Scholar, Web of Science, and Scopus were analysed in this chapter, each reporting applications of ANNs in shipboard or maritime contexts over the past decade. The indexing of the reviewed papers across quartiles according to the Clarivate Journal Citation Reports (JCR) is presented in Figure 2.6.

To obtain a clear overview of current developments, these papers were examined in terms of application purpose, ANN type, activation function, evaluation method, and data-acquisition and processing techniques. The findings are summarised in a table and supported by graphical representations, while the discussion highlights dominant trends such as the most frequently applied architectures, activation functions, and training algorithms.

The increasing commitment of maritime stakeholders to utilise onboard data underscores the importance of data-processing technologies such as ANNs, as analysed in [63]. The present review identifies uncovered areas and limitations related to developing ANN models for specific maritime applications and outlines directions for further work.

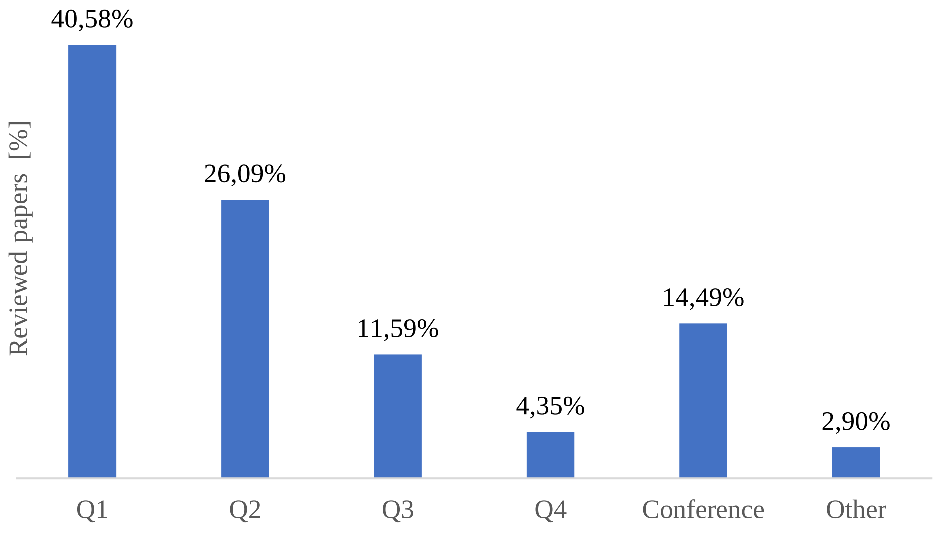


Figure 2.6 Indexing of the reviewed papers according to the JCR

2.2.1. Artificial Neural Network architectures

The foundations of Artificial Neural Networks (ANNs) were established in 1943 when McCulloch and Pitts [64] introduced the first mathematical neuron model inspired by biological neural cells. In their formulation, both input and output signals were Boolean, and the neuron output represented a logical function of its inputs. A major step forward came in 1958, when Rosenblatt [65] proposed the perceptron, which incorporated weighted input connections and enabled processing of continuous-valued data instead of purely binary signals.

To understand the operation of ANNs, it is necessary to examine the functioning of an individual neuron. The biological neuron, illustrated in Figure 2.7, consists of a cell body containing the nucleus, dendrites that receive signals, an axon that transmits impulses, and synapses that connect to other neurons.

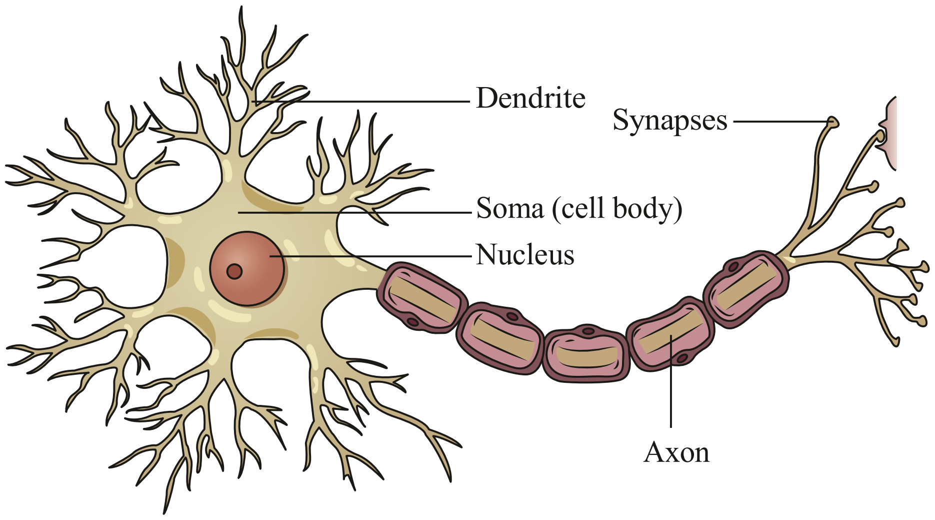


Figure 2.7 Illustration of a biological neuron [66]

Artificial neurons replicate these elements mathematically, where the dendrites correspond to weighted inputs, the synapse to the weighting process, the nucleus to the summation and activation operations, and the axon to the output signal.

An Artificial Neural Network consists of a collection of such neurons organised into layers that are interconnected to form a system capable of information processing. Depending on how neurons are structured and connected, numerous network architectures have emerged and been applied to a variety of scientific and engineering fields [67]. These architectures differ not only in their connection topology but also in the activation functions used by the neurons and the training algorithms applied to adjust their weights. Continuous research and technological progress have made ANNs an increasingly flexible and powerful tool for solving complex problems across domains.

Although many ANN types exist, only those that have appeared in the reviewed maritime literature are discussed in this chapter. The analysed network types include the feedforward Artificial Neural Network (FFANN), Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), Radial Basis Function network (RBF-ANN), Self-Organising Map (SOM), Generalised Regression Neural Network (GRNN), Support Vector Machine (SVM), Long Short-Term Memory network (LSTM), and the Bayesian-Transformer Neural Network (BTNN).

The feedforward Artificial Neural Network (FFANN), shown in Figure 2.8, represents the most fundamental and widely adopted ANN topology. In this architecture, neurons are organised in successive layers, i.e., input, one or more hidden layers, and output, through which signals propagate strictly in the forward direction without any feedback connections. The absence of feedback simplifies both network structure and training, making the FFANN suitable for a broad range of static nonlinear-mapping problems encountered in maritime systems.

Each neuron in an FFANN receives inputs x_i, where n_x is the number of input neurons. These inputs are multiplied by the corresponding connection weights $w^1{}_{ih}$, combined with a bias term $w^1{}_{0h}$. The resulting weighted sum is passed through a nonlinear activation function g, usually sigmoid, to produce the outputs of the n_h hidden neurons. The hidden layer outputs are then multiplied by the weights $w^2{}_{hj}$, summed with the output bias $w^2{}_{0j}$ and finally passed through linear activation function f to produce the network output y_j according to Equation (2.1).

$$y_j = f\left(w_{0j}^2 + \sum_{h=1}^{n_h} w_{hj}^2 \cdot g\left(w_{0h}^1 + \sum_{i=1}^{n_x} w_{ih}^1 \cdot x_i \right)\right) \tag{2.1}$$

During training, the network iteratively adjusts its weights to minimise the difference between the predicted and desired outputs. The back-propagation algorithm and its derivatives, including the Levenberg-Marquardt method, are among the most commonly used techniques for this purpose.

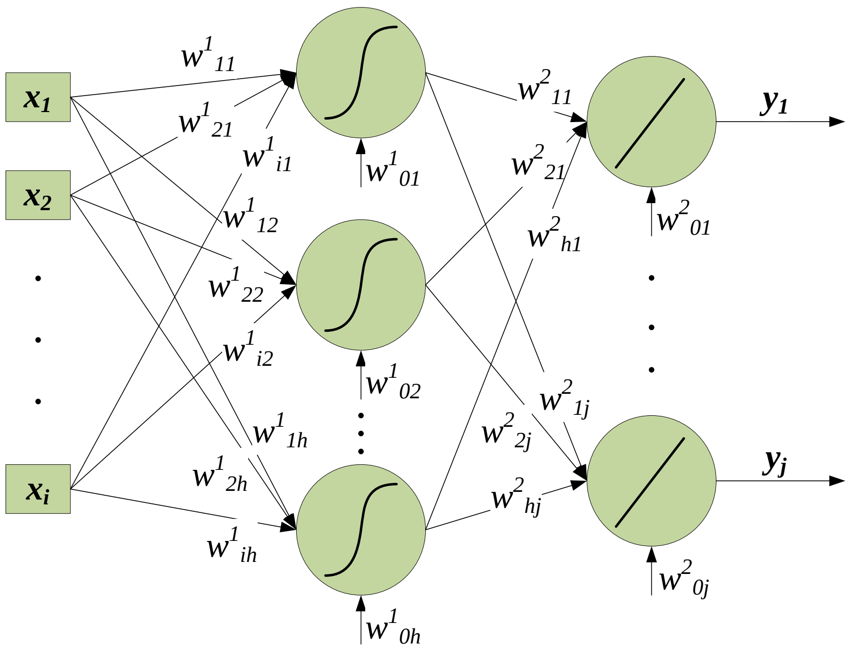


Figure 2.8 Architecture of a feedforward Artificial Neural Network (FFANN)

The Recurrent Neural Network (RNN) differs from a feedforward network in that it includes at least one feedback connection, forming one or more feedback loops. This structure enables the network to process sequential data and capture temporal dependencies between successive inputs. A simplified schematic of an RNN is shown in Figure 2.9.

In this configuration, a neuron can receive not only external input signals but also feedback from its own output or from the outputs of other neurons in the network. The feedback may be directed either toward hidden layers or back to the same neuron, thereby forming a self-feedback loop. As noted in [68], such recurrent connections have a significant influence on the network's learning dynamics and overall performance.

The RNN's dynamic behaviour arises from the inclusion of unit-time delay operators, commonly denoted as z^{-1}, which store the previous output state and reintroduce it as an input in the following time step. This mechanism allows the network to model nonlinear, time-dependent processes. Each neuron in an RNN receives inputs $x_i(t)$, where n_x is the number of input neurons. These inputs are multiplied by the corresponding connection weights w^1_{ih} and combined with a bias term w^1_{0h}. The resulting weighted sum is passed through a nonlinear activation function g, usually sigmoid, to produce the outputs of the n_h hidden neurons. The hidden layer outputs are then multiplied by the weights w^2_{hj}, summed with the output bias w^2_{0j}, and combined with a recurrent feedback term from the previous output $y_j(t\text{-}1)$. Finally, the resulting sum is passed through linear activation function f to produce the network output y_j according to Equation (2.2).

$$y_j(t) = f\left(w^2_{0j} + \sum_{h=1}^{n_h} w^2_{hj} \cdot g\left(w^1_{0h} + \sum_{i=1}^{n_x} w^1_{ih} \cdot x_i(t) \right) + y_j(t-1) \right) \tag{2.2}$$

Where W_x and W_y denote the weight matrices corresponding to input and recurrent connections, b represents the bias term, and f is the activation function.

Despite their capability to model temporal sequences, conventional RNNs often encounter training difficulties, such as the vanishing and exploding gradient problem, which limit their ability to capture long-term dependencies. These issues are mitigated in advanced variants such as the Long Short-Term Memory (LSTM) networks [69], described in subsequent sections.

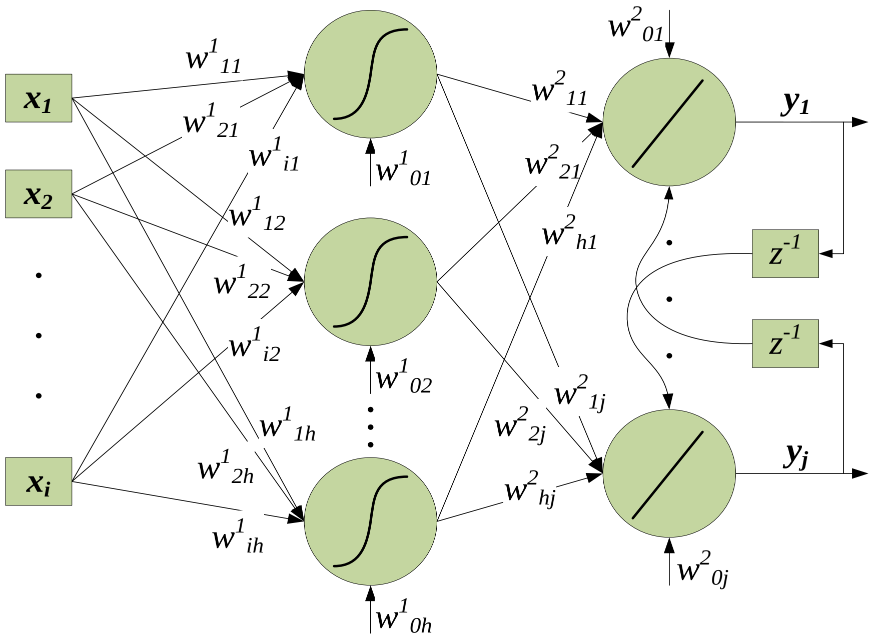


Figure 2.9 Architecture of a Recurrent Neural Network (RNN)

A Convolutional Neural Network (CNN) is a type of neural architecture designed to process data that contain spatial or grid-like structures, such as images, maps, or sensor matrices. Unlike fully connected feedforward networks, where each neuron in one layer connects to all neurons in the next, CNNs use local connectivity and weight sharing, enabling efficient feature extraction and reducing computational complexity. The fundamental building blocks of a CNN are convolutional layers, pooling layers, and fully connected layers, as illustrated in Figure 2.10.

The convolutional layer performs feature extraction by applying a set of learnable filters (kernels) that slide across the input data. Each filter produces a feature map that captures specific patterns or local dependencies. The output y_{ij} of a single convolutional neuron can be expressed by Equation (2.3).

$$y_{ij} = f\left(\sum_{m}\sum_{n} w_{mn}\, x_{(i+m)(j+n)} + b\right) \tag{2.3}$$

where $x_{(i+m)(j+n)}$ represents the input at position $(i+m, j+n)$, w_{mn} denotes the weight of the kernel element, b is the bias, and f is the activation function.

The pooling layer follows the convolutional layer to reduce spatial dimensions and computation, usually by taking the maximum or average value from small subregions of the feature maps. This process retains the most significant features while improving translation invariance. Finally, one or more fully connected layers are used to combine the extracted features and produce the final network output.

As noted in [70] and [71], CNNs are particularly effective at automatically learning hierarchical representations of data, progressing from low-level features in the early layers to high-level abstractions in deeper layers. Their architecture makes them well suited for structured input processing, such as image-based recognition, spatial data analysis, or multidimensional sensor fusion.

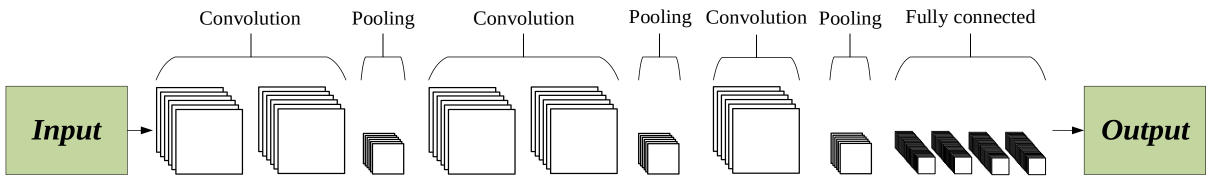


Figure 2.10 Architecture of a Convolutional Neural Network (CNN)

A Radial Basis Function Artificial Neural Network (RBF-ANN) is a feedforward neural network that uses radial basis functions as activation functions in its hidden layer. The activation of each hidden neuron depends mainly on the distance between the input vector and the activation functions centre, allowing the network to produce localized responses. Owing to this property, RBF networks are widely used for nonlinear function approximation and interpolation tasks, explained in [72].

As illustrated in Figure 2.11, an RBF network typically consists of an input layer that receives the input variables, a hidden layer whose neurons apply radial basis activation functions, most commonly of Gaussian form, and an output layer that forms a sum of hidden-layer activations to generate the final output. Each neuron in an RBF network receives inputs x_i, where n_x is the number of input neurons. These inputs are multiplied by the corresponding connection weights w^1_{ih} and combined to compute the distance between the input vector and the centre of each radial basis function. The resulting distance is passed through the radial basis functions Φ_h to produce the outputs of the n_h hidden neurons. The hidden layer outputs are then summed at the output layer to produce the network output y_j according to Equation (2.4).

$$y_j = \sum_{h=1}^{n_h} \Phi_h \left(\sum_{i=1}^{n_x} w_{ih}^1 \cdot x_i \right) \tag{2.4}$$

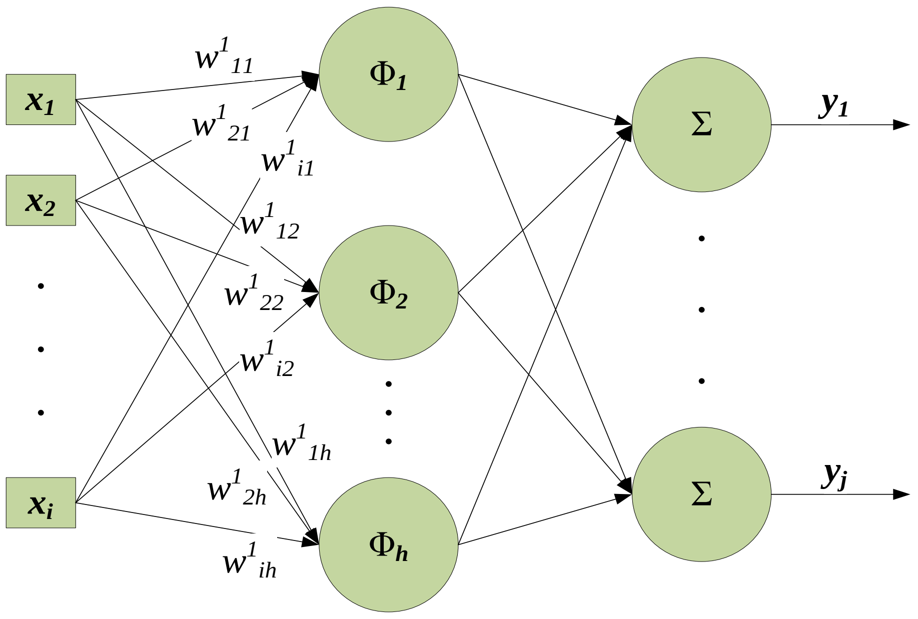


Figure 2.11 Architecture of a Radial Basis Function Artificial Neural Network (RBF-ANN)

A Self-Organising Map (SOM), shown in Figure 2.12, is also known as a Kohonen network. It is an unsupervised learning neural network designed for clustering, pattern recognition, and data visualization. Unlike supervised networks, SOMs learn the internal structure of data without predefined output labels, adapting their organization through a process known as competitive learning [73].

The architecture of a SOM consists of an input layer that receives multidimensional data and a map layer of neurons arranged on a low-dimensional grid, typically two-dimensional. The map layer is commonly organized as an $m \times n$ grid, which defines the fixed size of the SOM and is chosen prior to training as a design parameter. During training, the grid topology and dimensions remain unchanged, and only the weight vectors associated with the neurons are adapted. Each neuron in the map is associated with a weight vector of the same dimension as the input vector. Neurons that are physically close in the grid tend to respond to similar input patterns.

During training, all neurons compete to be activated by an input vector. The neuron whose weight vector is most similar to the input, called the best matching unit (BMU), is selected, and its weight vector, as well as those of its neighbours, are adjusted toward the input. This process gradually leads to the formation of a topology-preserving map, where neighbouring neurons correspond to similar input patterns. The BMU is identified by the minimum Euclidean distance, as expressed by Equation (2.5).

$$\left|\left|x - w_j\right|\right| = min_j \left(\left|\left|x - w_j\right|\right|\right) \tag{2.5}$$

Where x is the input vector and w_j is the weight vector of the j-th neuron. Through this competitive and neighbourhood-based adaptation, the SOM projects high-dimensional input data into a lower-dimensional space while preserving their topological relationships.

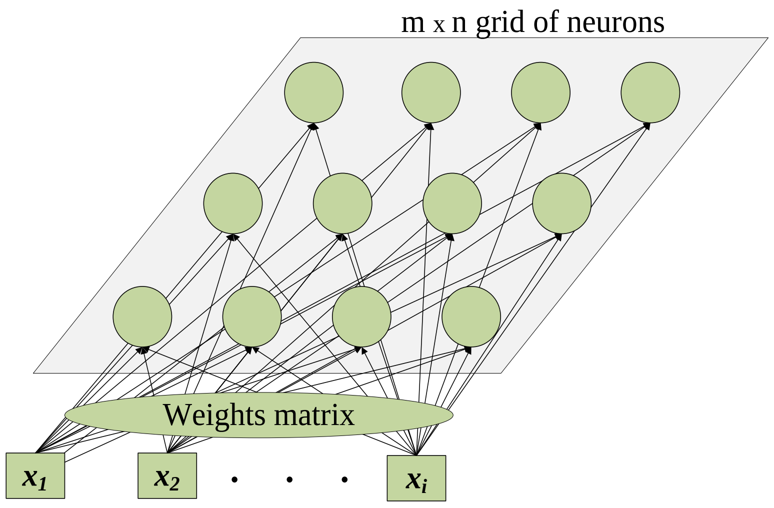


Figure 2.12 Architecture of a Self-Organising Map (SOM)

The Generalised Regression Neural Network (GRNN), introduced in [74], is a type of radial basis function network that performs regression using a probabilistic approach. Therefore, it shares the same architecture as presented in Figure 2.11. Unlike conventional feedforward networks that rely on iterative weight adjustment, the GRNN estimates continuous functions directly from training data using statistical principles derived from probability-density estimation. This allows the network to achieve rapid learning with high interpolation accuracy for nonlinear regression problems.

The GRNN architecture consists of four layers: an input layer, a pattern layer, a summation layer, and an output layer. Each neuron in the pattern layer represents one training sample and computes the distance between the input vector and the stored sample. The activation of the j-th pattern neuron is determined by a radial basis function, typically Gaussian, which quantifies the similarity between the current input and the training sample. The output layer then performs a normalised weighted average of all pattern-layer activations to generate the final network output. The general form of the GRNN output can be by Equation (2.6).

$$y = \frac{\sum_j y_j \, e^{\left(-\frac{\|x-x_j\|^2}{2\sigma^2}\right)}}{\sum_j e^{\left(-\frac{\|x-x_j\|^2}{2\sigma^2}\right)}} \tag{2.6}$$

Where x is the input vector, x_j is the j-th training sample, y_j is the target output associated with that sample, and σ is the smoothing parameter that governs the width of the radial basis function.

Due to its non-iterative training and inherent statistical formulation, the GRNN converges rapidly to an optimal regression surface as the number of training samples increases. It provides robust performance even with sparse or noisy datasets, making it an efficient alternative to traditional back-propagation networks for function approximation and prediction tasks [74].

A Support Vector Machine (SVM), shown in Figure 2.13, is a supervised learning algorithm used for both classification and regression tasks. It works by finding a boundary, called a hyperplane, that separates data points from different classes or describes a regression relationship with the smallest possible error [75].

The SVM aims to find the hyperplane that maximises the margin, i.e., the distance between the hyperplane and the nearest data points from each class known as support vectors. This property provides strong generalisation capability, making SVMs particularly effective for nonlinear and high-dimensional problems while making it computationally inexpensive as stated in [76].

For nonlinear separations, input data are transformed into a higher-dimensional space using a kernel function, where a linear separation becomes feasible. Commonly used kernels include linear, polynomial, radial basis function (RBF), and sigmoid kernels.

The decision function of an SVM can be expressed by Equation (2.7).

$$y_j = \sum_{h=1}^{n_h} w_{hj}^2 \cdot K\left(\sum_{i=1}^{n_x} w_{ih}^1 \cdot x_i, x_h\right) + w_{0j}^2 \tag{2.7}$$

Where x_i denotes the i-th component of the input vector, n_x is the number of input variables, and w^1_{ih} represents the input connection coefficients associated with the h-th kernel unit. The kernel function K measures the similarity between the transformed input and the reference vector x_h. The coefficients w^2_{hj} are the output-layer weights connecting the h-th kernel unit to the j-th output neuron, and w^2_{0j} denotes the bias term. Number of kernel units is represented with n_h, while y_j represents the resulting network output.

During training, the SVM solves a quadratic optimisation problem that maximises the margin subject to correct classification or minimal regression error. The resulting model achieves strong generalisation even with limited training data and avoids overfitting.

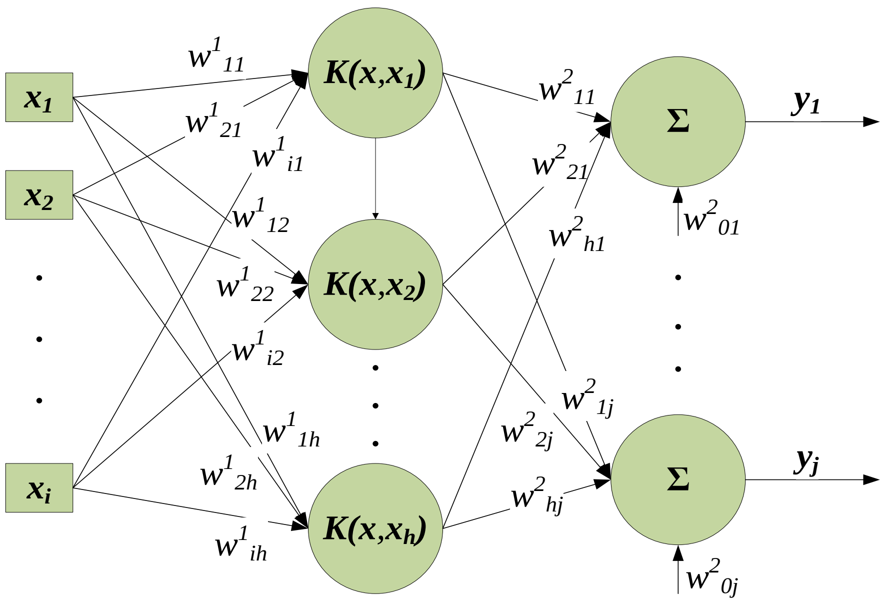


Figure 2.13 Architecture of a Support Vector Machine (SVM)

The Long Short-Term Memory (LSTM) network, shown in Figure 2.14, is a special form of Recurrent Neural Network (RNN) designed to overcome the limitations of conventional RNNs, particularly the problem of vanishing and exploding gradients that arise when learning long-term dependencies in sequential data as stated in [69].

An LSTM unit introduces a memory cell that maintains its state over time and a set of gating mechanisms, i.e., the input gate, forget gate, and output gate that regulate the flow of information. These gates enable the network to decide which information to retain, which to discard, and when to output the stored information, allowing it to effectively model long temporal dependencies as described in [77].

The internal operations of a typical LSTM cell can be described by the Equation (2.8)

$$y_{j(t)} = f\left(\sum_{h=1}^{n_h} w_{hj}^2 \cdot g\left(\sum_{i=1}^{n_x} w_{ih}^1 \cdot x_{i(t)} + w_h^z \cdot z_{h(t-1)}\right) + w_j^y \cdot y_{j(t-1)}\right) \tag{2.8}$$

Where $x_i(t)$ denotes the input at time step t and n_x is the number of input variables. The coefficients w^1_{ih} represent the weights connecting the input layer to the h-th hidden neuron in a layer of n_h hidden neurons, while $z_h(t-1)$ is the previous activation of the h-th hidden neuron weighted by the recurrent coefficient w^z_h. The function g denotes the hidden-layer sigmoid activation function. The coefficients w^2_{hj} are the output-layer weights connecting the hidden layer to the j-th output neuron. The term $y_j(t-1)$ represents the previous output fed back to the output neuron through recurrent weight w^y_j. The function f denotes the output-layer sigmoid activation function and $y_j(t)$ is the network output at time step t.

Through this structure, LSTMs can selectively retain important past information across long sequences, making them highly effective for time-dependent data such as sensor readings, trajectories, or operational logs.

As highlighted in [69], the gating mechanisms of LSTMs significantly enhance their performance over standard RNNs in handling long-term temporal correlations, leading to widespread adoption in sequence prediction and forecasting tasks.

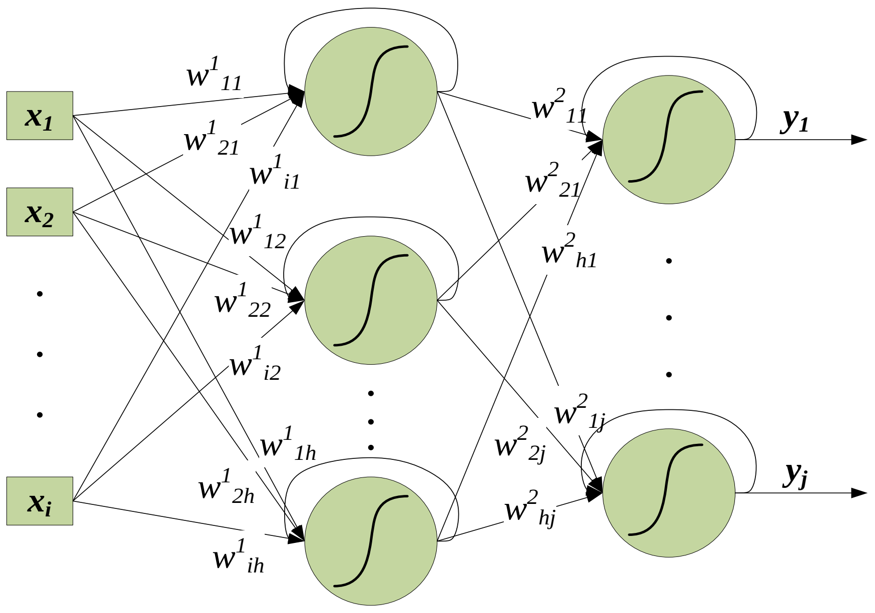


Figure 2.14 Architecture of a Long Short-Term Memory network

The Bayesian-Transformer Neural Network (BTNN) represents a recently introduced hybrid deep-learning architecture that combines the Bayesian inference framework with Transformer-based attention mechanisms as described in [78]. This hybridisation enables the model to leverage the probabilistic uncertainty estimation of Bayesian networks together with the powerful feature-extraction and sequence-modelling capabilities of Transformers.

While conventional neural networks provide point estimates of their outputs, Bayesian neural networks model the uncertainty in the network's parameters by treating weights as probability distributions rather than fixed values. This probabilistic formulation allows BTNNs to quantify prediction confidence, improving model interpretability and robustness, particularly in cases of noisy or incomplete data.

Transformers, originally developed for natural language processing, rely on a self-attention mechanism that evaluates relationships between all input elements simultaneously. By integrating Bayesian inference into the Transformer architecture, the BTNN achieves both context-aware learning and uncertainty estimation within a single model. A schematic structure of such a model is illustrated in Figure 2.15.

Mathematically, the Bayesian formulation of network weights can be represented by Equation (2.9).

$$p(w|x_i, y_j) = \frac{p(y_j|x_i, w) \cdot p(w)}{p(y_j|x_i)} \tag{2.9}$$

Where w denotes the weights of the neural network, x_i represents the input variables, and y_j represents the corresponding network outputs. The term $p(w)$ is the prior distribution over the weights, $p(y_j \mid x_i, w)$ is the likelihood of observing the output y_j given the input x_i and the network weights, and $p\ (y_j \mid x_i)$ is the normalisation term.

In a BTNN, this probabilistic component is embedded within the Transformer's multi-head attention structure, allowing the model to compute not only deterministic predictions but also posterior distributions over outputs. As reported in [78], this results in improved generalisation and reliability in predictive modelling tasks, particularly where data uncertainty is significant.

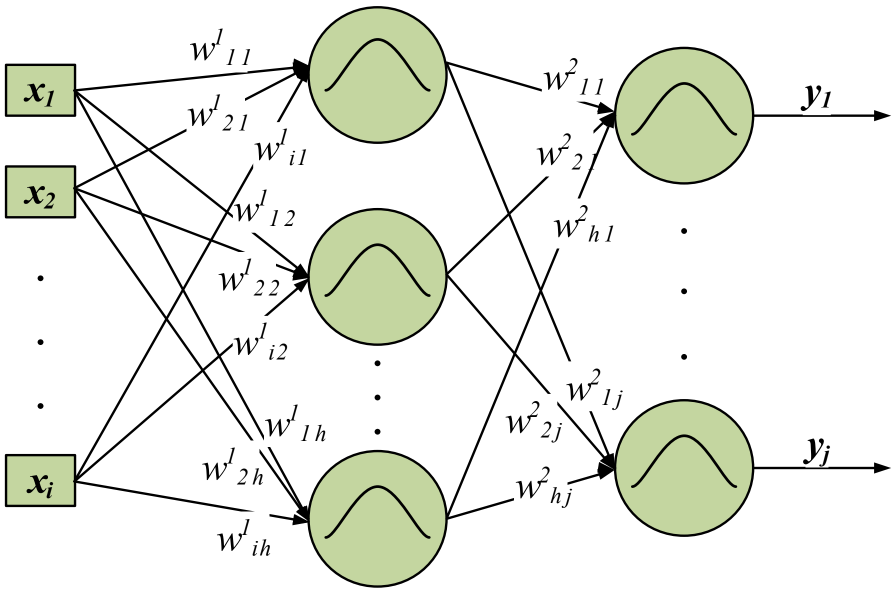


Figure 2.15 Architecture of a Bayesian-transformer neural network (BTNN)

2.2.2. Activation functions

The activation function defines the nonlinear transformation applied to the neuron's input before generating its output, thereby enabling ANNs to approximate nonlinear mappings between input and output data [79].

Without these functions, an ANN, regardless of its size, would behave as a linear model. The selection of an activation function directly affects the convergence speed, stability, and accuracy of the learning process [80].

A variety of activation functions have been proposed in the literature, differing in range, continuity, differentiability, and computational efficiency.

The most frequently used activation functions include the bipolar sigmoid, sigmoid, Rectified Linear Unit (ReLU), Leaky ReLU, linear, Softmax, RBF Kernel and exponential linear units (ELU) functions. These activation functions are defined by following equations where K represents number of classes in multi-class classifier, x represents data point, x_1 and x_2 represent two data points in space and α represents function slope coefficient.

The bipolar sigmoid activation is a variant of the standard sigmoid function that maps input values into the range (−1, 1), rather than (0, 1). It is defined by an Equation (2.10).

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.10}$$

The sigmoid (logistic) activation maps the neuron input to the range (0, 1) according to the Equation (2.11).

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.11}$$

The ReLU activation function is defined by Equation (2.12).

$$f(x) = \text{MAX}(0, x) \tag{2.12}$$

To overcome ReLU's inactivity for negative inputs, the Leaky ReLU introduces a small slope α (< 1), according to Equation (2.13).

$$f(x) = \text{MAX}(\alpha x, x) \tag{2.13}$$

The linear activation function is defined by Equation (2.14) and it does not include nonlinearity and is typically used in output layers of regression models where continuous outputs are desired.

$$f(x) = x \tag{2.14}$$

The Softmax function converts a vector of raw network outputs into a normalized probability distribution according to Equation (2.15). It ensures that all outputs are positive and sum to one, which makes it suitable for multi-class classification.

$$S(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.15}$$

The RBF kernel or Gaussian activation is widely used in RBF networks and Support Vector Machines. It is defined by Equation (2.16).

$$K(x_1, x_2) = e^{-\frac{\left||x_1 - x_2|\right|^2}{2\sigma^2}} \tag{2.16}$$

The ELU activation function combines linear and exponential characteristics, according to Equation 2.17. For negative inputs, the output asymptotically approaches -α, which helps reduce bias shifts and improves training stability.

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1)\,, & x \leq 0 \end{cases} \tag{2.17}$$

The summary of the most commonly used activation functions with corresponding mathematical equations and applications in neural networks is presented in Table 2.3. Furthermore, activation functions are graphically presented in Figure 2.16.

Table 2.3 Summary of activation functions and their use in neural network architecture

Activation function	**Mathematical expression**	**Application**
Bipolar sigmoid	$\sigma(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$	Output & hidden layers
Sigmoid	$\sigma(x) = \frac{1}{1 + e^{-x}}$	Hidden layers
ReLU	$f(x) = max(0, x)$	Input, output & convolutional layers
Leaky ReLU	$f(x) = max(\alpha x, x)$	Input, output & convolutional layers
Linear	$f(x) = x$	Output layers
Softmax	$S(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$	Input & output layers
RBF Kernel	$K(x_1, x_2) = e^{-\frac{\lVert x_1 - x_2 \rVert^2}{2\sigma^2}}$	Hidden layers
ELU	$f(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}$	Input & output layers

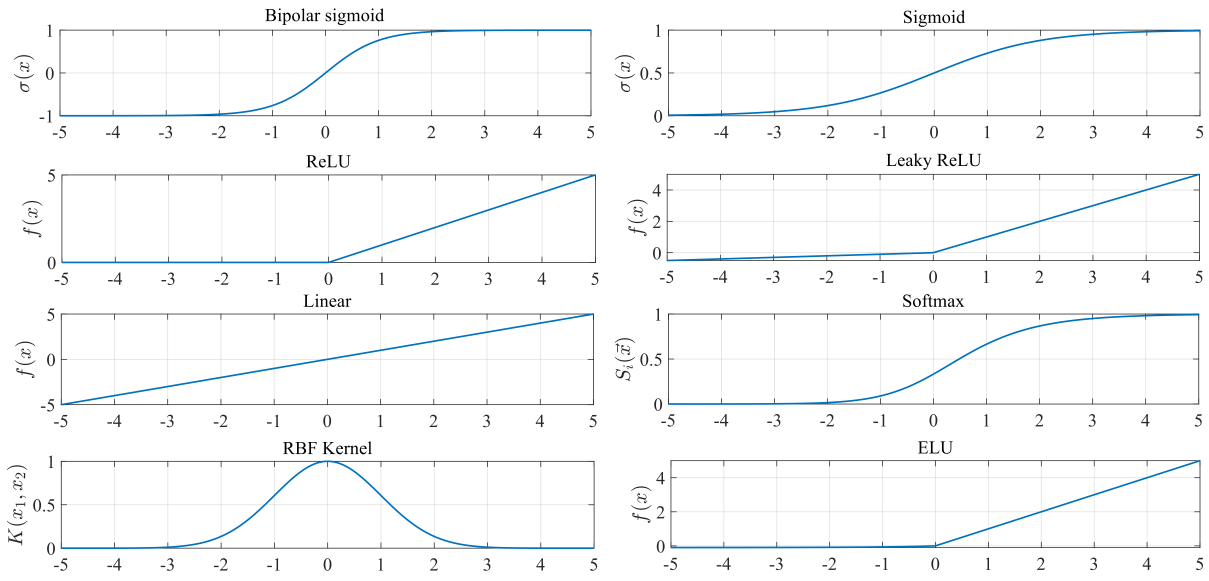


Figure 2.16 Graphical representation of commonly used activation functions

2.2.3. Training algorithms

Artificial Neural Networks learn from data by iteratively updating their weight coefficients to reduce a chosen loss (error) function. Depending on how target information is provided, learning may be supervised, i.e., input-output pairs are known, unsupervised, i.e., targets unknown, competitive learning, or reinforcement-based, i.e., trial-and-error interaction with an environment guided by a critic or teacher. In reinforcement learning, the procedure follows Thorndike's law of effect, increasing the likelihood of actions that lead to favourable outcomes as stated in [79]. During training, one or more algorithms are used to compute errors or gradients and update parameters. Below, the most commonly used algorithms are outlined.

One of the most common supervised learning training algorithms is Backpropagation algorithm (BP). It computes the gradient of the error function with respect to each network weight and updates weights w_{ij} iteratively according to Equation (2.18).

$$\mathrm{w}_{ij}(\mathrm{t}+1) = \mathrm{w}_{\mathrm{ij}}(\mathrm{t}) - \eta \frac{\partial E}{\partial \mathrm{w}_{\mathrm{ij}}} \tag{2.18}$$

Where η is the learning rate, E is the total network error, and $\partial E/\partial w_{ij}$ is the partial derivative of the error with respect to the connection weight between neurons i and j.

BP consists of a forward pass, i.e., computing outputs and a backward pass, i.e., propagating the error until the mean-squared error meets a predefined tolerance.

The Levenberg-Marquardt (LM) or sometimes called damped-least-squares (DLS) method combines the Gauss-Newton and gradient descent approaches. The weights are updated according to Equation (2.19).

$$\Delta w = -(J^T J + \mu I)^{-1} J^T e \tag{2.19}$$

Where J is the Jacobian matrix of partial derivatives of network errors with respect to weights, μ is the damping factor, I the identity matrix, and e the error vector.

When μ is small, LM behaves like the Gauss-Newton method, while for large μ, it approximates gradient descent.

The method was proposed in [81] and reported suitable for nonlinear least-squares curve fitting in FFANN applications in [82].

Furthermore, Density-based spatial clustering of applications with noise (DBSCAN) algorithm is an unsupervised learning clustering algorithm based on data-point density. A data point p from dataset D is considered a core point if at least a pre-defined minimum number of points q lies within its ε-neighbourhood N_ε according to Equation (2.20).

$$N_\varepsilon(p) = \{q \in D | dist(p, q) \leq \varepsilon\} \quad (2.20)$$

Points reachable from core points form clusters, while others are classified as outliers. Unlike the K-means algorithm, the number of clusters are not a priori defined, as stated in [83].

The Almeida-Pineda algorithm, proposed in [84] extends standard BP for recurrent neural networks (RNNs) by including delayed feedback connections.

The error gradient is accumulated over time steps, and weight changes *Δw_{ij}* are computed according to Equation (2.21).

$$\Delta w_{ij}(t) = -\eta \frac{\partial E(t)}{\partial w_{ij}} \quad (2.21)$$

Where η represents learning rate controlling the adjustment magnitude, $E(t)$ is a network error at time step t, and $\partial E/\partial w_{ij}$ is the partial derivative of the error with respect to the connection weight between neurons i and j as in BP algorithm.

In Self-Organising Maps (SOMs), the Euclidean distance algorithm is used to determine the Best Matching Unit (BMU) by finding the neuron whose weight vector is closest to the input vector. The relationship is expressed in Equation (2.22)

$$d_i = \left|\left|x - w_i\right|\right| \quad (2.22)$$

Where d_i represents the Euclidean distance between the input vector x and neuron i's weight vector w_i. The neuron with the smallest d_i is selected as the BMU, after which its weights and those of its neighbours are adjusted to better represent the input space as explained in [68].

The Stochastic Gradient Descent (SGD) algorithm, explained in [85] is used to iteratively minimize the loss function by updating weights based on gradients computed from randomly selected samples instead of the entire dataset. The update rule is expressed in Equation (2.23).

$$w_{ij}(t+1) = w_{ij}(t) - \eta \frac{\partial E(x_t)}{\partial w_{ij}} \quad (2.23)$$

Where η denotes the learning rate, $E(x_t)$ is the error function for the randomly selected training sample x_t, and $\partial E(x_t)/\partial w_{ij}$ represents the gradient of the error with respect to the weight w_{ij}.

A variant known as mini-batch gradient descent uses small subsets of training samples, improving numerical stability and convergence speed as stated in [86].

The Particle Swarm Optimization (PSO) algorithm is a population-based optimization technique inspired by the social behaviour of flocks and swarms, proposed in [87]. Each particle represents a potential solution that moves through the search space under the influence of both its own experience and that of its neighbours.

The particle velocity and position are updated according to Equations (2.24) and (2.25)

$$v_i(t+1) = \omega v_i(t) + c1r1\big(p_i - x_i(t)\big) + c2r2\big(g - x_i(t)\big) \tag{2.24}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.25}$$

Where $v_i(t)$ and $x_i(t)$ are the velocity and position of particle i at iteration t, ω is the inertia weight, $c1$ and $c2$ are acceleration coefficients, $r1$ and $r2$ are uniformly distributed random numbers in [0,1], p_i is the best-known personal position of particle i, and g is the global best position found by the swarm.

The Genetic Algorithm (GA) is an evolutionary optimization method based on the principles of natural selection, crossover, and mutation. Each candidate solution is assigned a fitness value, and the probability of selection for reproduction is calculated using Equation (2.26).

$$P_i = \frac{f_i}{\sum_j f_j} \tag{2.26}$$

Where P_{i} is the selection probability of solution i, f_i represents its fitness value, and $\Sigma_j f_j$ is the total fitness of the population. Over successive generations, fitter neurons are more likely to be selected, resulting in gradual improvement of the population’s overall performance. The recent advances and future directions of GA are analysed in [88].

The A-Star (A*) algorithm, explained in [89] is a graph-based pathfinding method that minimizes a combined cost function comprising the actual path cost and a heuristic estimate of the remaining cost to the goal.

The total cost function is expressed by Equation (2.27).

$$f(n) = g(n) + h(n) \tag{2.27}$$

Where *f(n)* is the estimated total cost of the path through node *n*, *g(n)* is the known cost from the start node to *n*, and *h(n)* is the heuristic estimate of the cost from *n* to the goal node. The node with the lowest *f(n)* value is selected for expansion in each iteration.

The K-Means algorithm, described in [90], is an unsupervised clustering method that partitions a dataset into *K* clusters by minimizing the sum of squared distances between each data point and its cluster centroid.

The objective function is given by Equation (2.28).

$$J = \sum_{i=1}^{K} \sum_{x \in C_i} | x - \mu_i |^2 \tag{2.28}$$

Where *J* represents the total within-cluster variance, *K* is the predefined number of clusters, *x* denotes a data sample from dataset, C_i is the *i*-th cluster, and μ_i is the corresponding cluster centroid. The centroids are updated iteratively until convergence is achieved, producing compact and well-separated clusters.

The K-Nearest Neighbours (k-NN) algorithm, explained in [90], is a non-parametric supervised learning method that classifies an unknown sample based on the classes of its *k* nearest neighbours in the feature space.

The classification decision is determined by majority voting among the *k* samples that have the smallest Euclidean distance to the query sample, as given by Equation (2.29)

$$D(x, x_i) = \sqrt{\sum_{j=1}^{n} \left(x_j - x_{ij}\right)^2} \tag{2.29}$$

Where $d(x,x_i)$ represents the Euclidean distance between the new sample *x* and the *i*-th training sample x_i, x_j and x_{ij} are the *j*-th components of these vectors, and *n* is the number of features.

Once distances are computed for all training samples, the *k* samples with the smallest distances are selected. The predicted class *C*(*x*) of the query sample is then determined according to Equation (2.30).

$$C(x) = mode\{C(x_1), C(x_2), \dots, C(x_k)\} \tag{2.30}$$

Where $C(x_i)$ denotes the class label of the *i*-th nearest neighbour and the operator *mode* selects the class occurring most frequently among the *k* nearest neighbours.

The performance of KNN strongly depends on the value of *k* and the choice of the distance metric. For numerical data, Euclidean distance is typically applied, while for categorical data, other metrics such as Hamming distance may be more appropriate.

The Radial Basis Function (RBF) network, as explained in [91] and [92] is a type of feedforward neural network that uses localised activation functions, typically Gaussian functions, in the hidden layer.

It is particularly effective for interpolation, function approximation, and classification problems due to its ability to represent nonlinear mappings.

The output of an RBF network is computed as a weighted sum of radial basis functions according to Equation (2.31).

$$y(x) = \sum_{i=1}^{N} w_i * \varphi\left(||x - c_i||\right) \tag{2.31}$$

Where $y(x)$ is the network output for input vector x, N is the number of neurons in the hidden layer, w_i denotes the weight associated with neuron i, c_i is the centre of the i-th radial basis function, and $\varphi(||x-ci||)$ is the activation function that depends on the distance between the input vector and the neuron's centre.

The Adam (Adaptive Moment Estimation) algorithm is a stochastic optimization method that extends the Stochastic Gradient Descent algorithm as described in [93]. It computes individual adaptive learning rates for each parameter based on estimates of the first and second moments of the gradients, therefore reducing the number of training iterations as stated in [94]. At each iteration t, the gradient g_t, which measures how the loss function changes with respect to the network parameters, is calculated, and the first ($\underline{m_t}$) and second ($\underline{vt}$) moment estimates are updated according to Equations (2.32)-(2.34).

$$\begin{cases} m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t = \beta_2 v_{t-1} + (1 - \beta_2) {g_t}^2 \end{cases} \tag{2.32}$$

$$\begin{cases} \widehat{m}_t = \dfrac{m_t}{1 - {\beta_1}^t} \\ \hat{v}_t = \dfrac{m_t}{1 - {\beta_2}^t} \end{cases} \tag{2.33}$$

$$\theta_{(t+1)} = \theta_t - \alpha \frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.34}$$

In the above equations, θ_t represents the parameter vector at iteration t, while α denotes the learning rate. The coefficients β_1 and β_2 are exponential decay rates for the first and second moment estimates, respectively. The terms m_t and v_t correspond to the biased first and second moment estimates of the gradient, whereas $\widehat{m}_t$ and $\hat{v}_t$ are their bias-corrected versions. The small constant ε is introduced to prevent division by zero during numerical computation.

The Extreme Learning Machine (ELM) trains a single-hidden-layer feedforward network by randomly assigning hidden-layer parameters and then solving for the output weights in a single step via the Moore-Penrose pseudoinverse. This avoids iterative gradient descent and yields very fast training while retaining good generalization for many regression and classification tasks. In compact matrix form, the output weights are obtained as in Equation (2.35):

$$\beta = H^{+} + T \tag{2.35}$$

and the network output for a given input set is computed by Equation (2.36):

$$Y = H\beta \tag{2.36}$$

In these expressions, H is the hidden-layer output matrix where each column corresponds to a hidden neuron's activation over all training samples, H^{+} denotes the Moore-Penrose pseudoinverse of H, β is the or matrix of output weights, T is the matrix of target outputs, and Y is the matrix of network outputs. Hidden-layer weights and biases are set randomly and remain fixed while β is solved analytically, which eliminates iterative weight updates and substantially reduces training time as explained in [95].

The Dynamic Time Warping (DTW) algorithm is a distance-based technique used to measure the similarity between two temporal sequences that may vary in speed or length as explained in [96]. It determines the optimal alignment between sequences by minimizing the cumulative distance between their elements through dynamic programming.

The accumulated distance between two time series $X=(x_1, x_2, \ldots, x_i)$ and $Y=(y_1, y_2, \ldots, y_j)$ is defined recursively as in Equation (2.37).

$$D(i,j) = \left|x_i - y_j\right| + min\{D(i-1,j), D(i,j-1), D(i-1,j-1)\} \tag{2.37}$$

Here, $D(i,j)$ represents the cumulative distance between the first i elements of sequence X and the first j elements of sequence Y. Furthermore, $|x_i - y_j|$ denotes the point-wise absolute difference, usually Euclidean distance in higher-dimensional cases. The algorithm constructs a cost matrix where each cell $D(i,j)$ corresponds to the minimum total cost of aligning the first i and j samples, and the optimal warping path is found by back-tracking from $D(i,j)$ to $D(1,1)$.

DTW enables comparison of sequences that are time-shifted or speed-scaled, making it especially useful for aligning sensor signals or vibration patterns recorded under varying conditions.

The Random Forest (RF) algorithm, as explained in [97], is an ensemble-based supervised learning method that constructs multiple decision trees during training and outputs the class that is the mode of the predictions in classification or the mean value in regression of the individual trees.

Each tree in the forest is trained on a randomly selected subset of the data and a randomly chosen subset of features, which enhances generalization and reduces overfitting. The prediction of the RF algorithm is given by Equation (2.38).

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x) \tag{2.38}$$

Where $\hat{y}$ represents the final predicted output, T is the number of decision trees in the ensemble and $h_t(x)$ denotes the prediction of the t-th individual decision tree for the input vector x.

During training, each tree is built from a bootstrap sample of the data, and at each split, a random subset of features is considered to determine the best split criterion This combination of bagging (bootstrap aggregation) and feature randomness allows Random Forests to achieve high accuracy and robustness against noise and overfitting.

The QuickBundle (QB) algorithm is a fast, unsupervised clustering method originally developed for grouping streamlines in diffusion tractography [98], but it has been adapted in maritime data analytics for grouping similar trajectories or multivariate time series [99]. The algorithm clusters data points by evaluating a minimum distance threshold between trajectories, grouping them into bundles based on geometric similarity.

For two trajectories s_i and s_j, each represented as a sequence of points, the Minimum Average Direct-Flip (MDF) distance is computed according to Equation (2.39).

$$d_{MDF}(s_i, s_j) = min\left(d_{direct}(s_i, s_j), d_{flipped}(s_i, s_j)\right) \tag{2.39}$$

Here, $d_{direct}(si,sj)$ represents the average point-wise Euclidean distance between the trajectories s_i and s_j in their original orientation, whereas $d_{flipped}(si,sj)$ computes the same distance after reversing the order of one trajectory.
The MDF distance ensures orientation invariance, allowing the algorithm to identify similar trajectories regardless of their direction of traversal.

If $d_{MDF}(si,sj)$ is less than a predefined threshold θ, the trajectories are assigned to the same cluster, otherwise, a new cluster is formed. Cluster centroids are then updated as the mean of all trajectories within the cluster.

The Variational Inference (VI) algorithm, presented in [100], is a probabilistic technique used to approximate complex posterior probability distributions when direct computation is intractable.

Instead of sampling, as in Monte Carlo methods, VI transforms the inference problem into an optimization problem by introducing a family of simpler distributions and finding the member that best approximates the true posterior. The objective is to minimize the Kullback-Leibler (KL) divergence between the approximating distribution $q(z)$ and the true posterior $p(z|x)$, as defined in Equation (2.40).

$$KL(q(z)\,||\,p(z|x)) = \int q(z)\, ln\left(\frac{q(z)}{p(z|x)}\right) dz \tag{2.40}$$

Because the true posterior $p(z|x)$ is typically unknown or computationally expensive to evaluate, the algorithm maximizes an equivalent objective known as the Evidence Lower Bound (ELBO), expressed in Equation (2.41).

$$L(q) = E_{q(z)}[ln(p(x,z)) - ln(q(z))] \tag{2.41}$$

In these equations, z represents the latent variables, x is the observed data, $q(z)$ denotes the approximating variational distribution, and $p(x,z)$ is the joint distribution of data and latent variables.
The goal of training is to adjust the parameters of $q(z)$ such that the ELBO $L(q)$ is maximized, thereby minimizing the divergence between $q(z)$ and the true posterior $p(z|x)$.

Variational inference provides a computationally efficient alternative to exact Bayesian inference and is particularly suitable for large-scale or streaming datasets, where full posterior sampling is infeasible.

The Saliency Detection Algorithm, reviewed in [101], is an unsupervised image-analysis method that identifies visually significant regions or objects in an image based on contrast and feature distinctiveness.

It estimates a saliency map representing the likelihood that each pixel or region attracts visual attention, enabling tasks such as object detection, segmentation, and focus-of-attention modelling.

In this approach, the saliency value of a pixel i is computed relative to its contrast with all other pixels j in the image, as formulated in Equation (2.42).

$$S(i) = \sum_j D(I_i, I_j)\, e^{\left(-\left\|p_i - p_j\right\|^2 / \sigma_p^2\right)} \tag{2.42}$$

Here, $S(i)$ represents the saliency score of pixel i, $D(I_i,I_j)$ is the color or intensity difference between pixels i and j, p_i and p_j denote the spatial coordinates of the pixels, and σ_p is a scaling parameter controlling the spatial weighting.
The exponential term ensures that nearby pixels contribute more strongly to the saliency measure, while distant pixels have a reduced influence.

The resulting saliency map highlights regions that differ significantly in appearance or spatial configuration compared to their surroundings.

The Bayesian Regularization (BR) algorithm is an extension of the standard back-propagation training method that incorporates Bayesian inference principles to improve network generalization and prevent overfitting.

Instead of minimizing only the mean squared error between predicted and target outputs, BR simultaneously minimizes a combination of the sum of squared network errors and the sum of squared weights, resulting in an automatically balanced regularization effect.

The objective function is defined as in Equation (2.43).

$$F = \beta E_D + \alpha E_W \tag{2.43}$$

Where E_D represents the sum of squared errors (data term), E_W is the sum of squared weights (regularization term), and α and β are the regularization parameters controlling their relative influence.

These parameters are adjusted automatically using Bayesian estimation principles:

1. α increases when the model becomes too complex (penalizing large weights),
2. β increases when the training error dominates (focusing on data fitting).

The network weights are updated by modifying the Hessian matrix of the performance function according to Equation (2.44).

$$H = 2\beta J^T J + 2\alpha I \tag{2.44}$$

where J is the Jacobian matrix of the network errors with respect to the weights, and I is the identity matrix. This formulation effectively combines Bayesian inference with the Levenberg-Marquardt approach, providing improved stability and automatic regularization during training.

The Decision Tree (DT) algorithm is a supervised learning technique used for both classification and regression tasks as stated in [102].

It operates by recursively partitioning the feature space into smaller subsets according to decision rules that maximize information gain or minimize impurity.

Each internal node represents a test on an input feature, each branch corresponds to the outcome of that test, and each leaf node represents a final decision or predicted value.

A common measure used for determining the best attribute split is information gain, defined in Equation (2.45).

$$IG(S,A) = Ent(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Ent(S_v) \tag{2.45}$$

In this expression, *IG*(*S*,*A*) denotes the information gain achieved by splitting dataset *S* using attribute *A*. *Ent*(*S*) is the entropy of the dataset before the split, *Values*(*A*) is the set of all possible values of attribute *A*, and S_v represents the subset of *S* corresponding to value *v*.

Entropy itself is calculated according to Equation (2.46).

$$Ent(S) = -\sum_{i=1}^{n} p_i log_2(p_i) \tag{2.46}$$

Here, p_i denotes the proportion of samples in class *i* within dataset *S*, and *n* is the total number of classes.

The attribute yielding the highest information gain is chosen for the next node, and this process continues recursively until the stopping criteria are met.

What is interesting to note is that according to [103], any ANN, regardless of the training algorithm used can be considered as a decision tree.

2.2.4. ANN applied to ship and ship's systems

The application of Artificial Neural Networks (ANNs) in the maritime domain has grown substantially over the last decade, driven by advances in sensor technology, data acquisition, and computational power. Ships are complex systems operating in dynamic and uncertain environments where large volumes of data are continuously generated from navigation, propulsion, machinery, and communication subsystems. Such data streams offer valuable opportunities for the implementation of data-driven methods capable of learning complex nonlinear relationships, identifying hidden dependencies, and providing predictive insight.

ANNs have therefore emerged as a practical and efficient tool to support a wide range of maritime tasks, from visual recognition and navigation assistance to machinery diagnostics, control optimization, and environmental monitoring. They have proven particularly valuable in situations where analytical modelling is difficult or computationally intensive, offering fast approximation and adaptability once properly trained.

The reviewed literature demonstrates that ANNs have been applied to nearly every functional aspect of ship operation, with increasing attention toward real-time applications and integration into intelligent or autonomous ship systems. The studies analysed can be categorized into several major application groups, including:

- ship image classification and type identification,
- ship route and trajectory applications,
- condition monitoring and condition-based maintenance (CBM),
- ship’s autopilot applications,
- controller tuning applications,
- gas emission applications (ecology),
- safety of navigation,
- other applications.

Each of these application categories will be discussed in the remaining part of this subchapter.

2.2.4.1. Ship image classification and type identification

ANN approaches, especially convolutional neural networks (CNNs), have been widely adopted for ship recognition from visual and radar images, with studies reporting strong performance across diverse datasets and evaluation protocols.

In [104], an AlexNet convolutional neural network (CNN) was employed to classify 35 ship types from optical images, with precision used as the principal evaluation metric.

Similarly, [105] implemented a You Only Look Once version 4 (YOLOv4) CNN in combination with a Kalman filter for real-time ship detection within surveillance footage.

A Grid CNN (GCNN) was proposed in [106], where input images were batch-normalized, i.e., the pixel intensities were normalised across each training batch, and divided into training (70%), validation (20%), and testing (10%) datasets. The model, developed in Python/Keras and trained using the Adam optimizer, was evaluated through precision, accuracy, and mean average precision (mAP).

In [107], a SCLANet based on Faster R-CNN was applied to SAR imagery for ship detection. Training employed 70% of the available images, with the remaining 30% used for testing. Developed in Python 3.6 and trained using stochastic gradient descent (SGD), the model's performance was quantified by average precision (AP) and compared with alternative networks including Faster R-CNN, YOLOv4, Spatio-Temporal Auto-Correlation (STAC) network, Channel Selection Dropout Single Shot MultiBox Detector (CSD-SSD) network, and UFO2.

A depth-wise separable CNN (DS-CNN) for high-speed SAR ship detection was introduced in [108]. The images were batch-normalized and divided into 70% training, 20% validation, and 10% testing subsets. Implemented in PyCharm using Python, Keras, and TensorFlow, and trained with the Adam optimizer, the DS-CNN was benchmarked against YOLOv2, YOLOv3, RetinaNet, YOLOv2-Tiny, and YOLOv3-Tiny. Evaluation metrics included precision, recall, mAP, and frames per second (FPS). The paper also examined the effects of pre-training on large datasets, anchor-box selection, concatenation, and multiscale detection mechanisms on model accuracy.

In [109], a YOLOv2-based CNN was used for real-time ship detection and classification into six categories. The model's effectiveness was measured by AP, mAP, recall, precision, and FPS, and its results were compared against YOLOv2, SSD, Fast-VGG, Fast-ZF, and Faster-VGG architectures.

A coarse-to-fine CNN was applied in [110] for ship detection and classification across seven vessel types. Datasets were split into 80% training and 20% testing, and model accuracy was used as the evaluation metric. Developed in MATLAB, the method was compared with CNN, random forest (RF), and K-nearest neighbour (KNN) approaches. It was noted that all tested models struggled to accurately identify LNG carriers.

Paper [111] presented a Bayesian-Transformer Neural Network (BTNN) for ship-type recognition from motion information. Six months of AIS data were normalized within the [0, 1] range, using 80% of samples for training and 20% for testing. The network, implemented in PyTorch, was trained using variational inference (VI) with Kullback-Leibler (KL) regularization. Its performance was measured through accuracy, weighted precision, weighted recall, and weighted F1-score, and benchmarked against SVM, FFANN, LSTM, and RNN models.

In [112], a YOLOv4-LITE CNN was developed for SAR-based ship detection. Image samples were clustered via the K-means algorithm and partitioned into training (70%), validation (20%), and testing (10%) datasets. The network, implemented with PyTorch, Python, and CUDA 10, was evaluated using mAP, AP, precision, figure of merit (FOM), recall, and FPS.

A VGG16 CNN was adopted in [113] for optical-image-based ship classification, trained using gradient descent on data split 80/20 between training and testing sets. The model's performance was measured by classification accuracy.

Similarly, [114] utilized a four-layer CNN trained to categorize ships into six infrared-image-based classes. An Extreme Learning Machine (ELM) algorithm combined with a sigmoid activation was used for training and compared against a BP-trained version of the same network, with accuracy as the evaluation measure.

In [115], CNNs were applied for ship identification in video surveillance. Noise in grayscale imagery was reduced using a wavelet transform, decomposing images into three high-frequency and one low-frequency component. Features extracted via Hu invariant moments were fused with CNN features to enhance discrimination. The modified AlexNet CNN, implemented in PyTorch, MATLAB, and Python, removed one convolutional layer, added a fully connected layer, and reduced neuron counts in the first two convolutional layers to lower complexity. Normalization used the z-score method, and horizontal flipping was applied for data augmentation. Performance metrics included accuracy, F1-score, and average feature-extraction time per image.

A comparative analysis of CNN architectures for multi-class ship classification was presented in [116]. Seven ship types were categorized using ELU and Leaky-ReLU activations, and networks such as YOLOv2, YOLOv3, and the proposed Regressive Deep CNN (RDCNN) were compared. Performance metrics included average intersection over union (AIOU), mAP, and recall, with training based on an iterative convergence procedure.

Reference [117] introduced a Cascade-Coupled CNN (3C2N) for SAR-image-based ship detection, integrating a non-local-means (NLM) filter for speckle-noise reduction. The model was evaluated using precision, recall, and F1-score, commonly adopted in object-detection studies.

In [118], another CNN architecture was used for SAR ship detection, incorporating an attention mechanism with a sigmoid activation to improve localization performance compared to conventional attention models. Evaluation metrics included generalized intersection over union (GIoU), recall, precision, average precision (AP), and F1-score, and results were compared against YOLOv3.

A CNN proposed in [119] for SAR-image ship detection consisted of three convolutional layers with ReLU activations, each followed by max-pooling, and concluded with an additional convolutional layer and softmax output. Model evaluation used precision, recall, and intersection over union (IoU) as performance criteria.

Finally, [120] proposed a Multiscale Adaptive Recalibration Network (MSARN), a CNN variant, implemented in TensorFlow. Training used stochastic gradient descent (SGD), and performance was assessed using average precision (AP) and precision-recall (PR) curves.

The MSARN results were compared against several benchmark models, including RBF Net, YOLOv3, Attention-ResNet, and Faster R-CNN.

2.2.4.2. Ship route applications

In [121], a three-layer feedforward neural network (7–9–2) was developed to predict course-change points during voyages. A sigmoid activation was applied in the hidden layer, and model accuracy was assessed using the mean squared error (MSE) criterion.

A separate model, described in [122], employed an FFANN (5–x–x–1) to estimate wave height for the purpose of optimal route planning. The network, implemented in Python/Keras, used the A* algorithm to support path optimization and was evaluated through MSE.

In [123], an RBF-based neural network was trained to forecast ship position in order to enhance path-following control. A supporting mathematical model was built for simulation and validation, where model outputs were quantified by mean absolute error (MAE) and mean absolute control effort (MAC).

A recurrent neural network (RNN) for short-term trajectory prediction was presented in [124]. The network was implemented using PyTorch and Python, and its parameters were updated through the Adam optimizer. Normalized AIS inputs served as data, while RMSE was used as the primary evaluation metric. The RNN’s performance was compared with a single-hidden-layer FFANN trained via back-propagation (BP).

In [125], a convolutional neural network (CNN) was constructed to classify ship trajectories derived from AIS data. The network, implemented in TensorFlow/Keras, received trajectory representations converted into image form. To increase data diversity and reduce overfitting, flipping and rotation were applied as augmentation techniques. Training employed the Adam optimizer, and results were measured using accuracy, precision, recall, F1-score, and area under the curve (AUC). Additional classifiers such as K-nearest neighbour (k-NN), decision tree (DT), and support vector machine (SVM) were also tested to compare their impact on performance.

A similar CNN-based approach was introduced in [126], where trajectories were categorized into voyage, manoeuvring, and static states. Color-coded trajectory images were generated in the same way as in [125]. The CNN, developed with Keras, TensorFlow, and Python, was assessed by accuracy, precision, and F1-score metrics.

An alternative CNN model proposed in [127] utilized deep convolutional architecture for trajectory classification based on AIS. The AIS data were first converted into image-based trajectories and grouped using an improved QuickBundle clustering algorithm, which was benchmarked against DBSCAN. The network was trained with the Adam optimizer in TensorFlow, and its performance was reported in terms of classification accuracy.

The study in [128] applied a recurrent neural network (RNN) to predict ship trajectories from AIS data. Data clustering was carried out using DBSCAN, while bipolar sigmoid and sigmoid activations were used in the hidden layers. Model results were compared against an LSTM-based solution and evaluated through MSE.

In [129], a hybrid LSTM-CNN architecture was trained to predict time series of electric-propulsion energy consumption during navigation. The objective was to optimize route planning for reduced environmental impact. Training used mini-batch gradient descent, while the bipolar sigmoid activation limited to the [−2, 2] range was applied within the hidden layer. Model performance was validated using RMSE.

Similarly, [130] introduced two LSTM-RNN models designed for marine transport surveillance and anomaly detection in vessel routes. Due to the absence of standardized datasets, validation was carried out through graphical comparisons with other approaches rather than quantitative error measures.

2.2.4.3. Condition monitoring and condition-based maintenance

In [131], a recurrent neural network (RNN) was applied to monitor the condition of a steam-plant regeneration heat exchanger under various operating regimes to detect potential malfunctions. The model’s three-layer architecture was determined empirically through trial and error. Sigmoid activation functions were used in the first two layers, while the output layer adopted a linear activation. The network was trained using the Pineda-Almeida algorithm and evaluated according to the mean squared error (MSE) criterion.

A self-organizing map (SOM) was developed in [132] for monitoring the piston-cooling oil pressure, exhaust-gas temperature, and piston-cooling oil temperature of a Panamax container ship’s main engine. Sensor data were clustered to form the input to the SOM, which comprised 16 neurons arranged in a 4×4 topology. The model’s performance was validated using classification accuracy as the evaluation metric.

In [133], a three-layer feedforward neural network (FFANN) incorporating a wavelet activation function was implemented for real-time engine monitoring and fault analysis. The Morlet function served as the wavelet basis, and a linear activation was used at the output.

Training was carried out using a genetic algorithm, while fitness and MSE were employed as performance measures.

A hybrid FFANN-fuzzy system for condition-based maintenance (CBM) of a ship electric-propulsion system was proposed in [134]. The model integrated fuzzy comprehensive evaluation with neural-network outputs to assess the propulsion system's health and support predictive maintenance decisions.

In [135], a CNN-RNN hybrid was utilized for hull corrosion detection using an autonomous inspection robot. The system employed GoogleNet Inception v3 architecture with ReLU and softmax activation functions. Model performance was evaluated using classification accuracy.

Paper [136] introduced an FFANN (100-50-10-3) designed to classify acoustic emission (AE) signals. AE analysis relies on detecting changes in wave propagation through a material caused by irreversible structural damage. Training combined unsupervised pre-training followed by BP fine-tuning, with model accuracy used for evaluation.

In [137], a three-layer FFANN was implemented for predictive maintenance through exhaust-gas-temperature monitoring of a two-stroke marine diesel engine. The network predicted exhaust temperatures up to five hours in advance, employing a bipolar-sigmoid activation in the hidden layer and a linear function in the output. Training utilized back-propagation (BP) with Bayesian regularization. Performance metrics included MSE, the correlation coefficient (R), and error autocorrelation.

A Random Forest–based decision-tree model was developed in [138] for ship-power-station fault diagnosis, and compared against an FFANN approach. The decision-tree system was built in Simulink, using Random Forest for ensemble training, whereas the FFANN used the BP algorithm. Model performance was assessed by R^2 and prediction speed.

In [139], a convolutional neural network (CNN) was proposed for fault diagnosis of ship electrical power systems comprising diesel and turbo generators. The power-plant model was implemented in MATLAB/Simulink, while the CNN was developed using Python/TensorFlow and trained via the BP algorithm. Accuracy was used to evaluate diagnostic performance.

An FFANN for main-engine exhaust-temperature prediction on a Panamax container ship was presented in [140]. The network was trained using the Levenberg-Marquardt algorithm and validated by MSE and correlation coefficient (R). Bipolar-sigmoid activations were applied in the hidden layers, while a linear activation was used at the output.

Paper [141] described a three-layer FFANN for fuel-consumption and voyage-speed prediction as part of hull-condition monitoring. The model was trained using Levenberg-Marquardt optimization combined with BP error minimization. A sigmoid activation function was used in the hidden layer, and performance was validated through Pearson's correlation coefficient.

In [142], a convolutional neural network (CNN) implemented in TensorFlow was used to estimate sea state and its influence on the structural response of a container-ship hull. Measurements collected on board included motions and hull-stress responses. The CNN estimated directional wave spectra, and the results were compared to established wave models. Evaluation metrics included MSE, 95th-percentile error, and Fisher's correlation factor, confirming the network's accuracy in sea-state estimation.

2.2.4.4. Ship's autopilot applications

In [143], a feedforward Artificial Neural Network (FFANN) was implemented as a neural controller for ship track-keeping. A single-input-multi-output (SIMO) configuration was adopted, allowing on-line training of the controller during operation. The network, structured as 10–10–1, was trained using the back-propagation (BP) algorithm. Performance assessment was conducted by comparing the tracking error between reference and actual course values under three route scenarios, demonstrating stable control behaviour.

Paper [144] introduced two FFANN models, (5–10–2) and (6–15–1), for autopilot applications based on an inverse modelling approach. This approach enabled the identification of the system's behavioural characteristics directly from operational data. Both networks were trained using the Levenberg-Marquardt algorithm, and simulations were performed for three sea states and calm conditions. Results indicated reduced rudder deflection amplitude and rate of change, thereby lowering mechanical stress on the steering gear. A hyperbolic tangent activation was used in hidden layers, and a linear function was employed in the output layer.

In [145], an LSTM recurrent neural network (RNN) was developed to predict wave height for integration into autopilot route optimization. Two independent networks were configured for two maritime regions, with the objective of selecting routes through areas of minimum wave height. Sigmoid and bipolar-sigmoid activations were applied in hidden layers. Model accuracy was benchmarked against FFANN, Random Forest (RF), and Support Vector Machine (SVM) models using MAE, RMSE, and correlation coefficient (R) as evaluation metrics.

A six-layer FFANN (3–12–12–12–12–2) was implemented in [146] for autonomous ship docking under variable wind conditions. The controller relied on a mathematical ship model to simulate manoeuvring dynamics during training. A genetic algorithm was used for optimization, while wind force acted as a disturbance input to test robustness. The hidden layers used bipolar-sigmoid activations, and the output layer applied a linear function. Performance evaluation was based on MSE between the predicted and desired docking paths.

In [147], an RBF-based neural network (RBF-ANN) was proposed to optimize a PID controller within an autopilot control loop. A sliding-mode PID controller was simulated using a MATLAB ship model, and the RBF network was trained using gradient descent. Performance was demonstrated by graphically comparing the system response, confirming improved tracking and disturbance rejection.

Finally, [148] proposed a Sea State Estimation Network (SSENet) built upon a CNN architecture to estimate wave height and direction for autopilot integration. The network was implemented in Python 3.6 using TensorFlow, Keras, and Anaconda environments. Training data were collected from two zig-zag manoeuvres, capturing ship motion in nine degrees of freedom (DOF). The model was trained using the Adam optimizer, and its performance was benchmarked against FFANN, LSTM, ResNet, and SeaStateNet architectures. Accuracy was used as the evaluation measure, with SSENet showing superior estimation capability.

2.2.4.5. Controller tuning applications

In [149], two feedforward Artificial Neural Networks (FFANNs) were implemented for self-tuning of a PID controller governing the stabilization fins of a ship. The first network (6–7–1) was designed to identify and predict the nonlinear input-output behaviour between the six degrees of freedom (DOF) motion inputs and the roll-angle output, using a bipolar-sigmoid activation function in the hidden layer. The second network (6–7–3) received as inputs the predicted roll angle from the first network and roll angular velocities from the previous five time steps, producing at its three output neurons the PID control parameters. A sigmoid activation function was used in its hidden layer. The performance of the overall control system was evaluated using the Roll Reduction Rate (RRR), defined as the percentage decrease in roll motion relative to the mean roll amplitude without the self-tuning control approach.

In [150], an FFANN (1–5–1) was employed as an excitation-voltage controller for a synchronous generator. The goal was to prevent excitation overcurrent during engine-speed drops and to maintain voltage stability at the generator terminals. A three-layer architecture was developed in MATLAB/Simulink, using a bipolar-sigmoid activation in the hidden layer and a linear function at the output. Performance assessment was carried out by analysing the generator’s output-voltage variation as a function of speed, confirming effective overcurrent protection and voltage regulation.

An RBF-based neural network (RBF-ANN) was utilized in [151] to model the ship’s dynamic response to environmental forces such as wind, waves, and sea currents for the purpose of developing an adaptive control system for the ship’s crane. The aim was to minimize cargo swing and improve disturbance rejection during lifting operations. The model was first verified through MATLAB simulations, followed by experimental validation on a physical test bed. Performance was evaluated by comparing swing-response curves and positioning errors obtained using the proposed predictive controller, a nonlinear tracking controller without predictive terms, and a Linear Quadratic Regulator (LQR) controller.

Finally, [152] introduced a four-layer FFANN serving as a neural network controller for a ship-mounted crane, compensating for load oscillations caused by wave-induced ship motion while neglecting wind effects. The model’s effectiveness was measured using the relative positioning error of the suspended load, confirming the neural controller’s capability to suppress load swing and enhance operational stability.

2.2.4.6. Gas emission applications

In [153], a VGG16 convolutional neural network (CNN) was applied to detect the sulphur content in marine-engine fuel by analysing ultraviolet (UV) images of exhaust gases. The same CNN architecture, previously used in [113], was adopted with a transfer-learning approach to accelerate convergence and improve generalization. The model was trained to perform classification of exhaust images, employing ReLU and sigmoid activation functions in the hidden layers. Cross-entropy served as the loss function, and performance was evaluated in terms of accuracy, calculated using a weighted-average method to account for class imbalance.

A related study [154] proposed a modified LeNet-5 CNN to determine sulphur concentration in exhaust plumes. The original LeNet-5 was extended by introducing a dropout layer that randomly deactivates 50 % of neurons during training, thereby improving network regularization and preventing overfitting. ReLU activation was employed in hidden layers and sigmoid activation at the output layer. The model was implemented in Python using the TensorFlow library, and trained via the Adam optimizer. Its accuracy, the primary evaluation metric, demonstrated the network’s ability to detect sulphur content reliably in real-time exhaust-monitoring applications.

In [155], a three-layer feedforward Artificial Neural Network (FFANN) with architecture 7–12–1 was used to predict fuel consumption of a diesel-mechanic tanker operating under variable conditions, supporting a decision-support system (DSS) aimed at reducing greenhouse-gas emissions. The model was trained using the Levenberg-Marquardt algorithm. Its performance was assessed through mean-squared error (MSE), root-mean-square error (RMSE), and coefficient of determination (R^2), and results were cross-validated using multivariate regression (MR) analysis in IBM SPSS.

Finally, [156] presented an FFANN-based fuel-consumption prediction model developed in TensorFlow for a cruise ship, designed to minimize GHG emissions using AIS-derived operational data. The network was integrated with four particle swarm optimization (PSO) algorithms for enhanced parameter tuning. Training utilized both stochastic and batch gradient-descent methods, while evaluation criteria included accuracy and variance. The hidden layers incorporated a mix of bipolar-sigmoid, softmax, sigmoid, and ReLU activation functions to balance convergence speed and nonlinear expressiveness.

2.2.4.7. Safety of navigation applications

In [157], a recurrent neural network (RNN) was developed for real-time detection of marine-traffic anomalies, aiming to enhance navigational safety and situational awareness. The model was trained using Automatic Identification System (AIS) data, which was first clustered with the DBSCAN algorithm to extract relevant trajectory features. Experimental validation demonstrated that the RNN effectively detected abnormal behaviour in vessel motion, including irregularities in course, speed, and trajectory patterns, thereby supporting early-warning systems for potential navigational hazards.

In [158], a convolutional neural network (CNN) was applied to classify ship collision risk based on AIS-derived encounter data. To prepare inputs for the model, MATLAB was used to generate visual representations of ship-encounter scenarios, which were then fed into the CNN for image-based risk assessment. Model performance was evaluated using Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE), demonstrating the method’s capability to quantify collision risk with high precision.

A long short-term memory (LSTM) recurrent neural network was proposed in [159] to construct a collision-probability model using Monte Carlo simulation techniques on AIS datasets. The approach, implemented in Python via the PyCharm development environment, incorporated temporal dependencies in ship-motion sequences to estimate probabilistic risk levels in dynamic maritime environments.

Finally, in [160], an RNN-based manoeuvring model was designed to support collision-avoidance prediction. The network architecture consisted of 21–21–21–3 layers and was initially trained as a feedforward ANN using the back-propagation (BP) algorithm with sigmoid activation functions in the hidden layers. After the training phase, feedback connections were introduced, transforming the architecture into a recurrent form capable of modelling dynamic ship responses. Training data were generated from both mathematical modelling and free-run scaled model experiments in a test basin. Performance evaluation was carried out using Mean Squared Error (MSE) and absolute error metrics, confirming the model's potential in ensuring safe navigation and effective collision avoidance.

2.2.4.8. Other applications

In [161], multiple neural-network architectures, including a Generalized Regression Neural Network (GRNN), feedforward Artificial Neural Network (FFANN), Fuzzy Neural Network (FNN), and Elman Neural Network (ENN), were developed and compared for port-traffic prediction. Prior to training, input data were processed using the Hilbert-Huang transform to decompose the signal into high and low-frequency components, while the Dynamic Time Warping (DTW) method was applied for clustering. Model performance was evaluated using Relative Error (RE), standard deviation, and residual analysis, with comparative results highlighting the benefits of non-linear neural structures for complex maritime traffic forecasting.

A three-layer FFANN (6–6–1) was implemented in [162] to estimate the added wave resistance coefficient, and its performance was compared with GRNN, RBF-ANN, and Linear Network models. Several training algorithms were tested, back-propagation, gradient descent, and Levenberg-Marquardt, to identify the most effective optimization method. The model was developed in Lazarus IDE using the Free Pascal compiler, and evaluated via STATISTICA software using Pearson's correlation coefficient (R) and Mean Squared Error (MSE). Hidden-layer activations included sigmoid and bipolar-sigmoid functions.

In [163], an FFANN trained with the back-propagation (BP) algorithm was applied to reduce sea clutter in medium-range marine radar systems, improving the detection of sea-surface objects. The network used a bipolar-sigmoid activation function in the hidden layer and a sigmoid function in the output layer. Performance was measured using MSE and Signal-to-Clutter Ratio (SCR), showing enhanced signal clarity and object differentiation.

A three-layer FFANN (6–25–1) was developed in [164] to determine the optimal rated output of a ship's main engine during propulsion-system design. Input parameters included draught, length overall, beam, tonnage, speed, and main engine power. The model used the BP algorithm for training and a unipolar-sigmoid activation in the hidden layer. Performance was assessed using the Root Mean Square Error (RMSE).

Reference [165] compared FFANN, RBF-ANN, and GRNN models for spatial modelling of sea-depth measurements. Each network employed various activation functions, and their results were evaluated using Mean Absolute Error (MAE) and maximum error, confirming the robustness of neural approaches in hydrographic mapping.

An FFANN model was developed in [166] using MATLAB to construct a regression model for ship fuel consumption prediction. The network was trained with the Levenberg-Marquardt algorithm, employing sigmoid, ReLU, and bipolar-sigmoid activations in the hidden layers. Evaluation was conducted using MSE and correlation coefficient (R).

In [167], an LSTM RNN was used to identify a multi-input multi-output (MIMO) ship model for manoeuvring-motion prediction. The network used sigmoid and bipolar-sigmoid activations in the input layer, and leaky ReLU in the output layer. The Adam optimizer minimized the error function, while RMSE and MSE were used for performance evaluation, confirming the model's ability to capture nonlinear dynamics.

An FFANN was applied in [168] to predict measurement results in order to reduce the number of required real measurements during steam-turbine efficiency assessments. Performance evaluation relied on Mean Absolute Error (MAE) and the coefficient of determination (R^2), showing that ANN-based inference could reliably substitute direct measurement campaigns.

A three-layer FFANN developed in MATLAB Neural Network Toolbox was utilized in [169] to predict torque, power, specific fuel consumption, and exhaust gas temperature of a four-stroke marine diesel engine. The Levenberg-Marquardt algorithm was applied for training, and results were validated in IBM SPSS using MSE and R^2 as evaluation metrics. The bipolar-sigmoid function was employed in the hidden layer.

In [170], an RBF-ANN was developed to model a diesel generator for a real-time marine power-system simulator. The architecture (5–99–3) was trained using the BP algorithm, with Gaussian activation functions in the hidden layer and a linear output. Performance was evaluated through error threshold and computation time, confirming the suitability of the model for real-time applications.

Paper [12] modelled significant wave height at a fixed location in the Adriatic Sea using an FFANN, alongside linear and nonlinear regression models. Optimal input and hidden-layer configurations were determined through three experiments, with evaluations based on RMSE, MAE, Nash-Sutcliffe coefficient of efficiency (CE), percent bias (PB), persistency index (PI), and RMSE-to-standard-deviation ratio (RSR). The ANN consistently outperformed regression models, particularly in scenarios involving multiple input variables.

Similarly, an FFANN was employed in [171] to predict wave-induced ship motion. A three-layer architecture with 25 hidden neurons was trained in MATLAB using the Bayesian regularization algorithm. Full-scale measurement data were collected from both the ship and an accompanying wave buoy. The dataset was split into 70 % training and 30 % validation subsets, and model performance was evaluated using RMSE, MAE, correlation coefficient (R), and coefficient of efficiency (CE).

2.2.5. Analysis of the ANN applications

Building on the reviewed literature, a comprehensive analysis of ANN applications in the maritime domain has been conducted. The assessment covered the frequency and purpose of applications, the types of networks, training algorithms, and evaluation measures used to construct models. Additionally, the analysis examined trend of published papers related to ANN in maritime domain, data-acquisition methods, data-processing tools, and the organization of training, validation, and testing datasets. This provides a detailed overview of how ANNs are currently implemented aboard ships and within broader maritime research.

The overall distribution of ANN applications is illustrated in Figure 2.17. The analysis reveals that ship image classification and type identification represent the most frequent application area, followed by condition monitoring, other applications, and ship-route optimization. On the other hand, comparatively fewer studies address ship autopilot, gas-emission estimation, control system design, and navigational safety tasks.

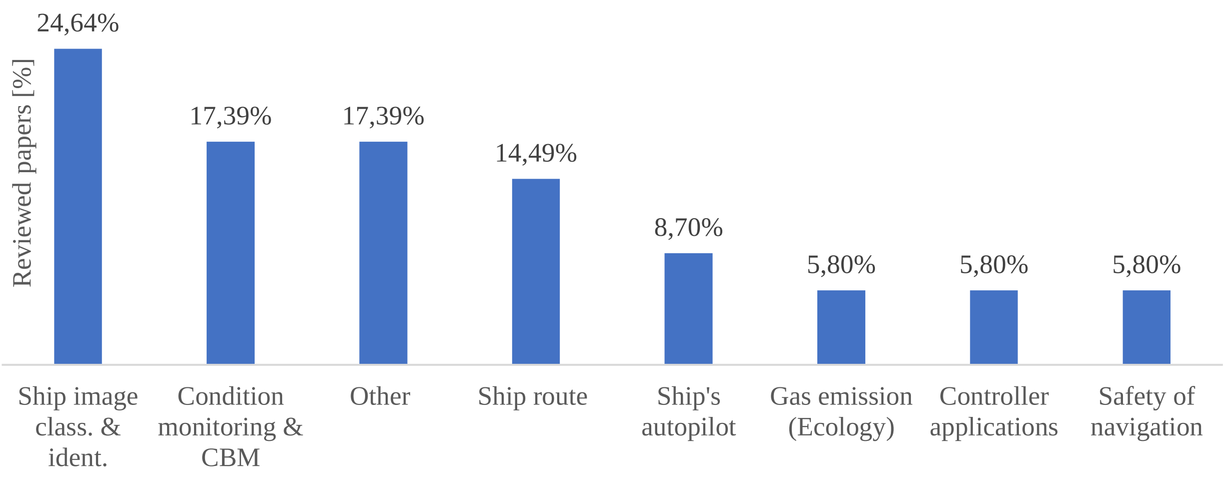


Figure 2.17 Distribution of application purposes in the reviewed literature

Regarding the types of neural networks employed, feedforward Artificial Neural Networks (FFANN) and Convolutional Neural Networks (CNN) dominate, as shown in Figure 2.18, while Recurrent Neural Networks (RNN) follow closely. Less frequently encountered are Radial Basis Function Networks (RBF-ANN), Generalized Regression Neural Networks (GRNN), Self-Organizing Maps (SOM), and Fuzzy Neural Networks (FNN), with Bayesian Transformer Neural Networks (BTNN) and Decision Trees being the least common.

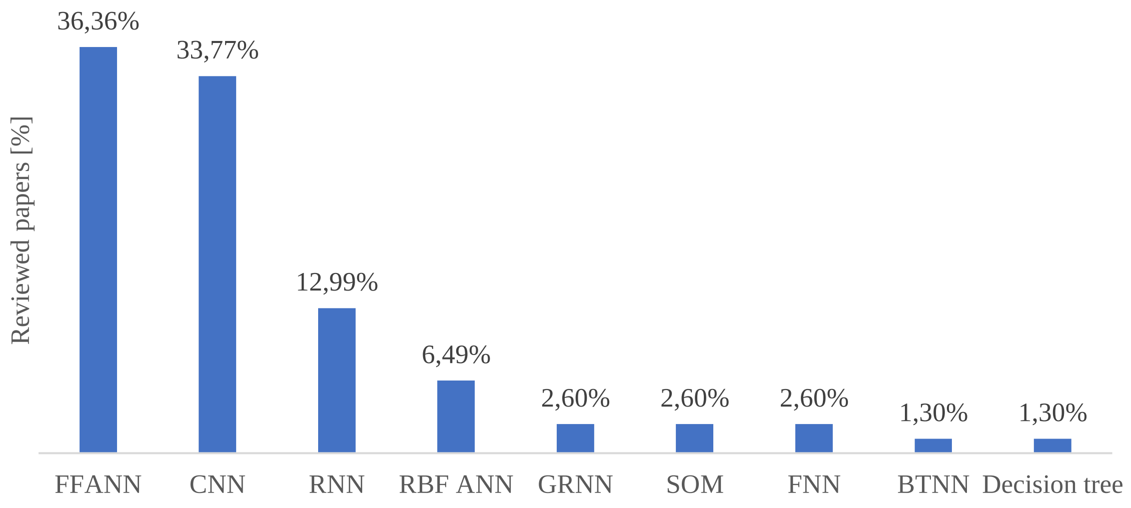


Figure 2.18 Distribution of neural network types used in the reviewed literature

In terms of training algorithms, back-propagation (BP) is by far the most frequently applied method, as shown in Figure 2.19. It is followed by Levenberg-Marquardt, Adam optimizer, transfer learning, and stochastic gradient descent (SGD). Less commonly, algorithms such as K-means, DBSCAN, RBF, and Saliency-Detection appear, while batch gradient descent, Genetic Algorithm (GA), Variational Bayesian, and Extreme Learning Machine (ELM) methods are used only occasionally.

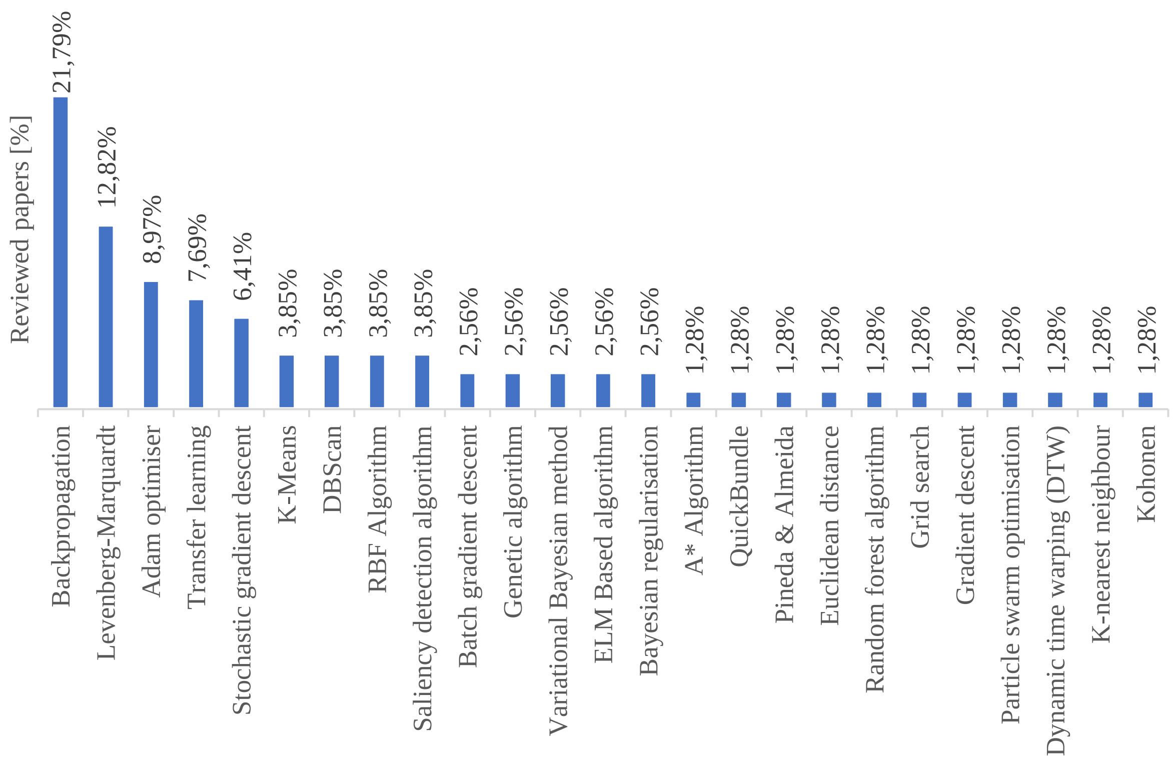


Figure 2.19 Distribution of training algorithms used in reviewed literature

The most frequently used evaluation metrics are accuracy and mean squared error (MSE), followed by precision, recall, average precision (AP), and root mean squared error (RMSE). Less-frequently applied metrics include correlation coefficient (R), mean average precision (mAP), F1 score, mean absolute error (MAE), and relative error (RE), as summarized in Figure 2.20.

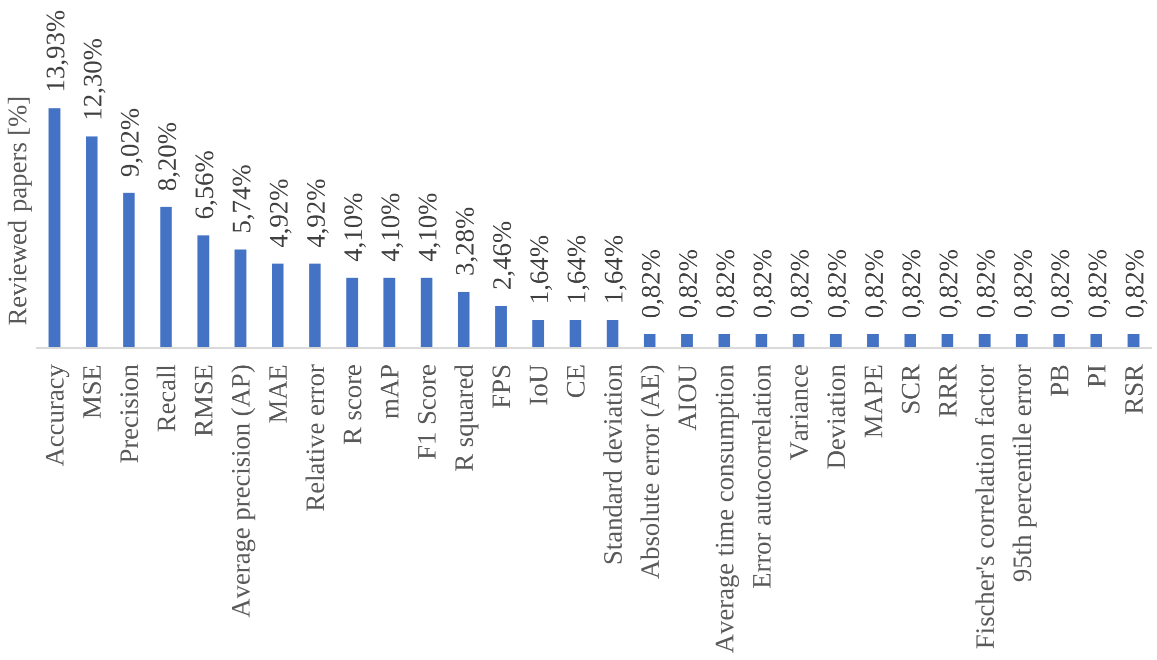


Figure 2.20 Distribution of evaluation measures used in reviewed literature

A cross-comparison of network type per application area is presented in Figure 2.21. CNNs are primarily employed for ship-image classification, while FFANNs are commonly used for fuel-consumption estimation and other applications. RNNs appear mainly in ship-route prediction, and RBF networks are distributed across machinery modelling, control system design, autopilot systems, and miscellaneous applications.

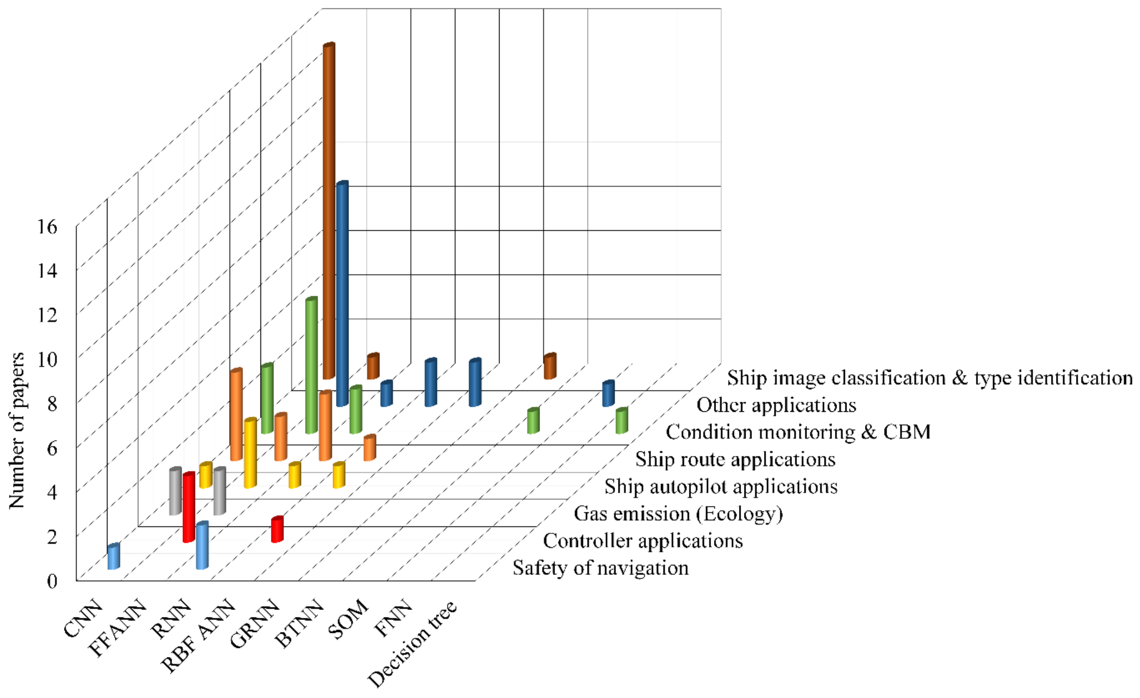


Figure 2.21 Distribution of neural network types per application purpose

Similarly, Figure 2.22 shows the distribution of training algorithms per application type. The BP algorithm is predominant, particularly within condition monitoring, autopilot, and other applications, followed by Levenberg-Marquardt, which is mostly used for autopilot and fuel-consumption prediction. The Adam optimizer appears mainly in ship-image classification and route prediction, while transfer learning is used almost exclusively for image classification.

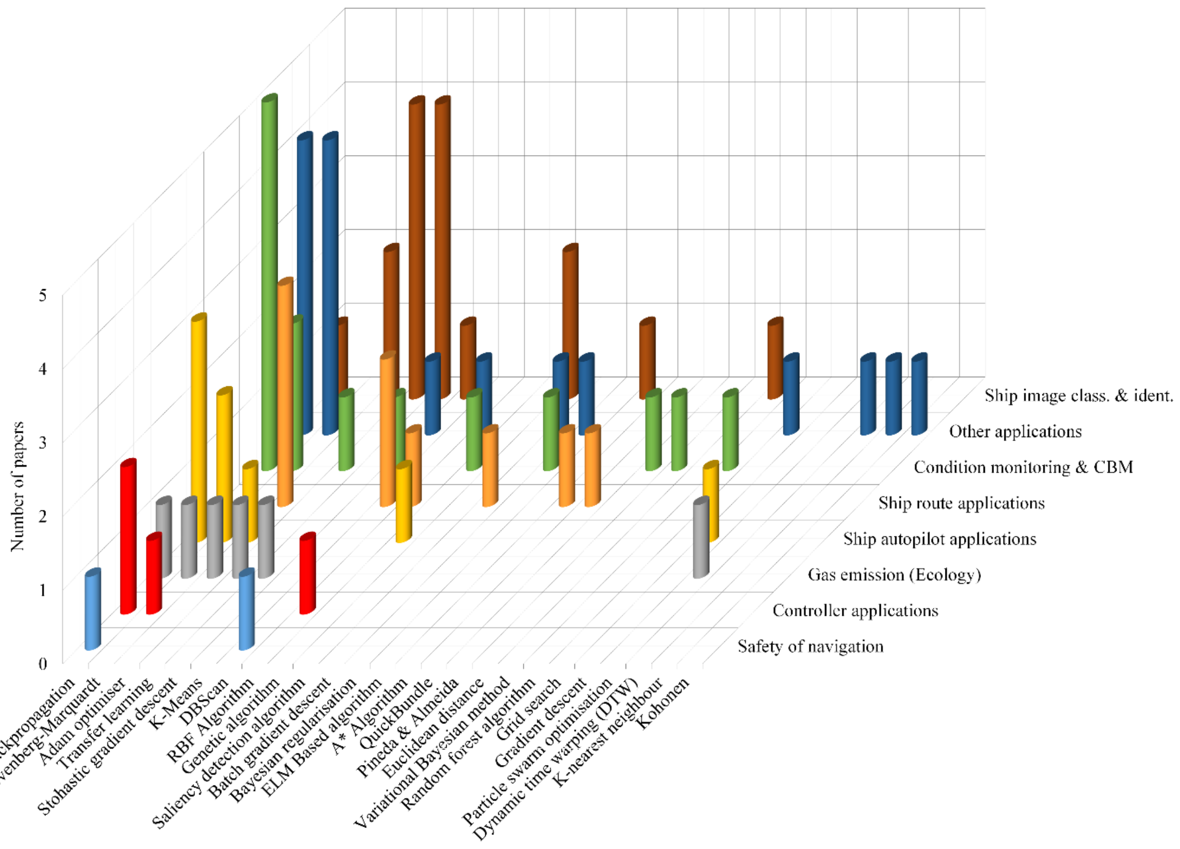


Figure 2.22 Distribution of training algorithms per application purpose

When it comes to evaluation criteria, MSE remains the most widely used across all categories, while accuracy, precision, and recall dominate in image-classification contexts, as shown in Figure 2.23.

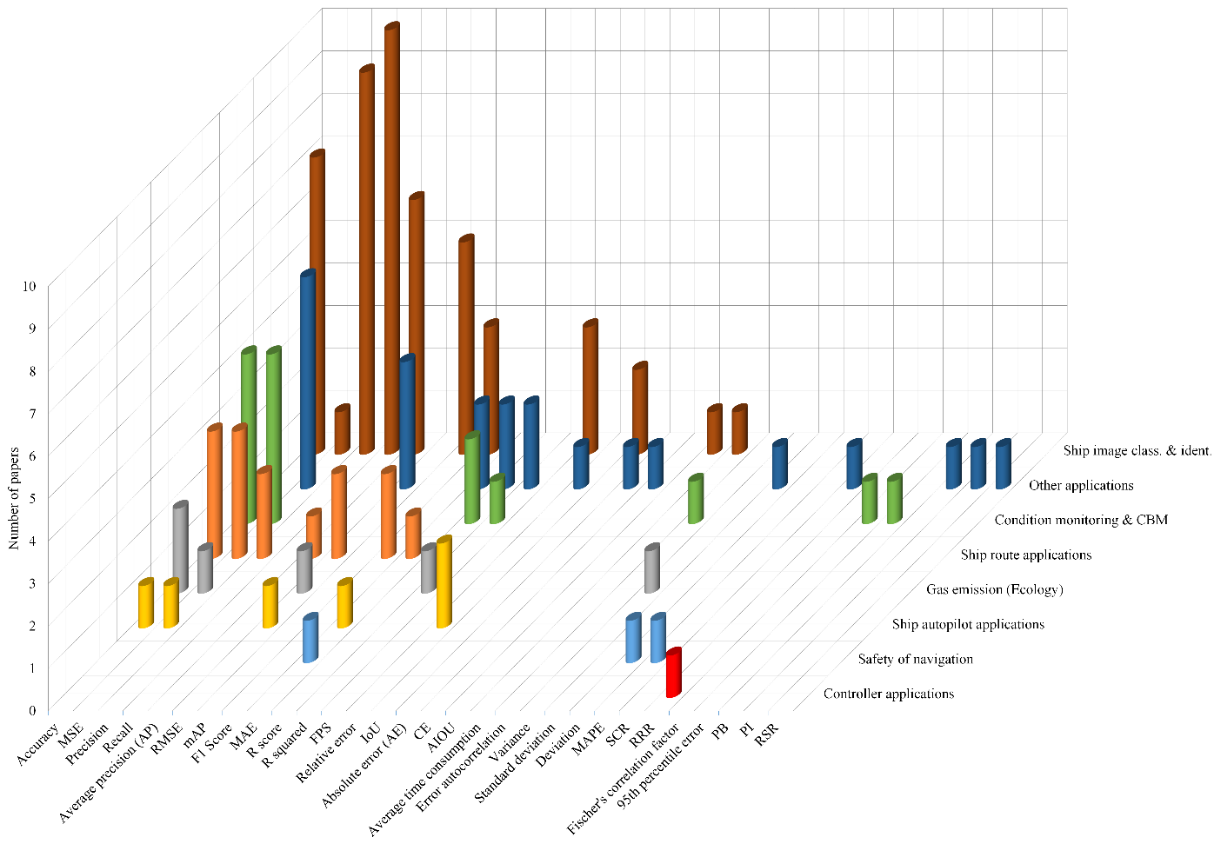


Figure 2.23 Distribution of evaluation measures per application purpose

The summarized results of ANN applications across the maritime domain are provided in Table 2.4, detailing application purposes, network types, training algorithms, and evaluation metrics identified in the reviewed studies.

Table 2.4 Summarized application-based analysis of the ANN applications

Group of application	Reference	Application purpose	Network type	Training algorithm	Evaluation measure
Ship image classification & type identification	Dao-Duc, C. et al., 2015	Ship classification from images	CNN (AlexNet)	-	Precision
	Petković, M. et al., 2021	Ship detection in video surveillance	CNN (YOLOv3)	-	-
	Zhang, T. et al., 2019	Ship detection in SAR images	CNN (Grid CNN)	Adam optimizer	Precision, accuracy, mAP
	Hou, B. et al., 2022	Ship detection in SAR images	SCLANet (Faster CNN)	SGD	AP
	Zhang, T. et al., 2019	Ship detection in SAR images	DS-CNN	Adam optimizer	Precision, recall, mAP, FPS
	Shao, Z. et al., 2020	Ship detection in video surveillance	CNN (YOLOv2)	-	AP, mAP
	Chen, X. et al., 2020	Ship classification from images	CNN (CFCCNN)	-	Accuracy

	Kong, Z. et al., 2022	Ship type identification from motion data	BTNN	Variational inference, Kullback-Leibler	Accuracy, precision, recall and F1 score
	Liu, S. et al., 2022	Ship detection in SAR images	CNN (YOLOv4LITE)	K-means (for clustering data)	mAP, AP, Precision, Recall, FOM, FPS
	Mishra, N. K. et al., 2022	Ship classification from images	CNN (VGG16)	Gradient descent	Accuracy
	Khellal, A. et al, 2018	Ship classification in infrared images	CNN	ELM	Accuracy
	Ren, Y. et al., 2021	Ship type identification in video surveillance	CNN (AlexNET)	-	Accuracy, F1 score, time consumption
	Huang, Z. et al., 2020	Ship classification from images	CNN (RDCNN)	Iterative convergence	AIOU, mAP, recall
	Zhao, J. et al., 2018	Ship detection in SAR images	CNN (3C2N)	-	Precision, recall, F1 score
	Chen, C. et al., 2019	Ship detection in SAR images	CNN	-	Precision, recall, GIoU, F1 score
	Wu, F. et al., 2018	Ship detection in SAR images	CNN	-	Precision, recall, IoU
	Chen, C. et al., 2019	Ship detection in SAR images	CNN (MSARN)	SGD	AP, PRC
Ship route applications	Daranda, A., 2016	Course change location prediction	FFANN	-	MSE
	Jin, X. et al., 2021	Wave height prediction	FFANN	A*	MSE
	Zhang, H. et al., 2021	Ship position prediction for path following improvement	RBF-ANN	-	MAE, MAC
	Zhu, F. et al., 2021	Prediction of ship trajectory	RNN	Adam optimizer	RMSE
	Chen, X. et al., 2020	Ship trajectory classification	CNN	Adam optimizer, K-NN, SVM, DT	Accuracy, precision, recall, F1 score, AUC
	Yang, T. et al., 2022	Ship trajectory classification	CNN	-	Accuracy, precision, F1 score
	Guo, T. et al., 2022	Ship trajectory classification	CNN	QuickBundle, DBSCAN, Adam optimizer	Accuracy
	Suo, Y. et al., 2020	Ship trajectory prediction	RNN	DBSCAN	MSE

	Kim, J.-Y. et al., 2021	Propulsion plant energy consumption prediction	LSTM-CNN	Minibatch gradient descent	RMSE
	Nguyen, D. et al., 2022	Anomaly detection in ship routes	LSTM-RNN	-	-
Condition monitoring & condition-based maintenance	Niksa-Rynkiewicz, T. et al., 2021	Steam plant malfunction detection	RNN	Pineda and Almeida	MSE
	Raptodimos, Y. et al., 2018	Ship engine condition monitoring	SOM	-	Accuracy
	Jiang, Y. et al., 2021	Ship engine real time monitoring and fault analysis	FFANN	Genetic algorithm	MSE, fitness
	Liang, S. et al., 2014	CBM of electric propulsion system	FFANN	-	Fuzzy comprehensive evaluation
	Le, A. V. et al., 2021	Corrosion detection	CNN, RNN	-	Accuracy
	Karvelis, P. et al., 2021	Ship structure condition monitoring	FFANN	BP	Accuracy
	Raptodimos, Y. et al., 2016	Condition monitoring and predictive maintenance of a diesel engine	FFANN	BP, Bayesian regularization	MSE, R, error autocorrelation
	Zhou, Y. et al., 2021	Ship power station fault diagnosis	FFANN	Random forest algorithm, BP	R^2, speed of prediction
	Yu, C. et al., 2022	Ship power plant fault diagnosis	CNN	BP	Accuracy
	Raptodimos, Y. et al., 2020	Main engine exhaust gas temperature prediction	FFANN	Levenberg-Marquardt	MSE, R
	Moreira, L. et al., 2021	Fuel consumption prediction for hull condition monitoring	FFANN	Levenberg-Marquardt	Pearson's correlation coefficient
	Kawai, T. et al., 2021	Sea state estimation for hull condition monitoring	CNN	-	MSE, 95th percentile error, Fischer's correlation factor

Ship's autopilot applications	Zhang, Y. et al., 1996	Neural controller for ship track-keeping	FFANN	BP	Error
	Le, T. T., 2021	Modelling ship behavior to develop autopilot	FFANN	Levenberg-Marquardt	Trial voyage
	Lou, R. et al., 2021	Wave height prediction	LSTM RNN	-	MAE, RMSE, R
	Shuai, Y. et al., 2019	Autonomous ship docking	FFANN	Genetic algorithm	MSE
	Yu, J. et al., 2022	Ship autopilot optimization	RBF-ANN	Gradient descent	Analysis of graphs
	Cheng, X. et al., 2020	Wave height estimation	SSENET (CNN based)	Adam optimizer	Accuracy
Controller tuning applications	Fang, M.-C. et al., 2010	Self-tuning of a PID controller for stabilization fins	FFANN	-	RRR
	Jeon, H. et al., 2020	Excitation voltage controller of a synchronous generator	FFANN	-	Analysis of graphs
	Qian, Y. et al., 2022	Prediction of ship response for adaptive controller development	RBF-ANN	-	Experimentally
	Yang, T. et al., 2020	Development of neural controller for ship's crane	FFANN	-	Relative error
Gas emission applications	Cao, K. et al., 2021	Fuel sulphur detection in exhaust gas	CNN (VGG16)	Transfer learning	Accuracy
	Han, Y. et al., 2022	Fuel sulphur detection in exhaust gas	Le-Net5 (CNN based)	Adam optimizer	Accuracy
	Beşikçi Bal, E. et al., 2013	Fuel consumption prediction for decision support system	FFANN	Levenberg-Marquardt	MSE, RMSE, R^2, Multivariate regression
	Zheng, J. et al., 2019	Fuel consumption prediction	FFANN	SGD, Batch gradient descent	Accuracy, variance
Safety of navigation	Zhao, L. et al., 2019	Marine traffic anomalies detection for augmented situational awareness	RNN	DBSCAN	Experimentally
	Zhang, W. et al., 2019	Classification of collision risk	CNN	-	MAE, MAPE
	Hao, L. et al., 2022	Modelling the ship manoeuvrability for predicting collision	RNN (trained as FFANN)	BP	MSE

Other applications	Li, Y. et al., 2019	Port traffic prediction	GRNN, FFANN, FNN, ENN	-	Relative error, standard deviation, and residual analysis
	Cepowski, T. et al., 2020	Estimation of added wave resistance coefficient	FFANN	Gradient descent, Levenberg-Marquardt, BP	MSE, Pearson's correlation coefficient
	Vicen-Bueno, R. et al., 2010	Sea clutter reduction on radar	FFANN	BP	MSE, SCR
	Meler-Kapcia, M. et al., 2005	Main engine optimal rated power when designing main propulsion system	FFANN	BP	RMSE
	Stateczny, A., 2000	Spatial modelling based on sea depth	FFANN, RBF-ANN, GRNN	-	MAE, maximum error
	Jeon, M. et al., 2018	Ship fuel consumption prediction	FFANN	Levenberg-Marquardt	MSE, R
	Jiang, Y. et al., 2022	Ship manoeuvring motion prediction	LSTM RNN	Adam optimizer	RMSE, MSE
	Baressi Šegota, S. et al., 2020	Steam turbine efficiency calculation for reduction of real measurements	FFANN	-	MAE, R^2
	Mohd Noor, C. W. et al., 2016	Marine diesel engine parameters prediction	FFANN	Levenberg-Marquardt	MSE, R^2
	Shi, W. et al., 2005	Diesel generator model for marine power system simulator	RBF ANN	-	Error threshold
	Mudronja, L. et al., 2017	Significant wave height modelling	FFANN	-	RMSE, MAE, NSC (CE), PB, PI, RSR

In total, sixty-nine papers were analysed, including nine network types, eight activation functions, and twenty training algorithms. These were categorized into eight primary application groups, each described earlier in Section 2.2.4.

The analysis confirms the increasing adoption of ANNs across diverse maritime applications, with a steady growth in publications over the past decade, as shown in Figure 2.24.

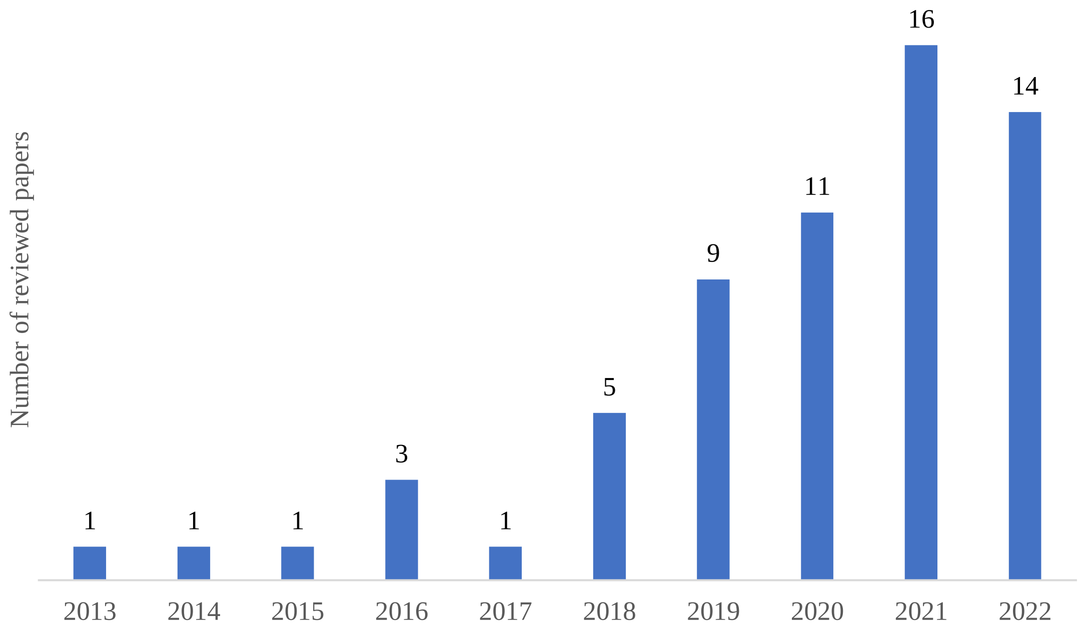


Figure 2.24 Number of published papers over the past decade

Despite this trend, several potential applications remain unexplored. For instance, fault detection in electric motors, as demonstrated in non-maritime studies [172], [173], [174], [175], could significantly benefit shipboard systems. Electric-motor condition monitoring based on current, magnetic-flux, torque, vibration, or acoustic measurements has proven effective in other industries but has yet to be adapted for shipboard machinery.

Similarly, while ANNs have been successfully applied to wave-induced motion prediction [171] and wave-parameter estimation [148], no studies were found addressing the influence of wave dynamics or onboard fluid movements on ship dynamic stability, despite the recognized impact of such effects [176]. Furthermore, there is a lack of ANN use for control system design while data availability is increasing.

These areas represent promising directions for future maritime ANN research.

2.2.5.1. Data acquisition methods

The reviewed literature demonstrates that data used for developing ANN-based ship models are most commonly obtained through onboard sensor systems.

Examples include temperature and pressure sensors as utilized in [132], vibration, rotational speed (rpm), temperature, and pressure sensors as in [133], and acoustic-emission sensors applied in [136].

Other studies employed specialized sensors such as rudder-angle sensors [144], motion, wind-speed, and rudder-angle sensors [146], and combinations of oxygen, knock, air-pressure, rpm, throttle-position, and temperature sensors [169].

In applications addressing hull or structure response, motion sensors integrated within inertial measurement units (IMUs) were used [148], while other studies employed mechanical stress and acceleration sensors to capture structural behaviour [142]. A smaller number of papers obtained measurement data from external sources rather than onboard systems.

For instance, wave-buoy measurements were used in [145] and [171] to record motion or sea-state characteristics, while publicly available environmental models, such as meteorological and wave datasets, were incorporated in [12].

In addition, some studies, such as [160] relied on synthetic data generated from mathematical ship-motion models.

However, the analysis shows that in 82.91 % of the reviewed studies, authors did not specify the exact data-acquisition approach used. Among the subset of papers that did provide this information, the identified acquisition methods are summarized in Figure 2.25, while Figure 2.26 presents a breakdown of the onboard sensor types most frequently encountered in the reviewed research.

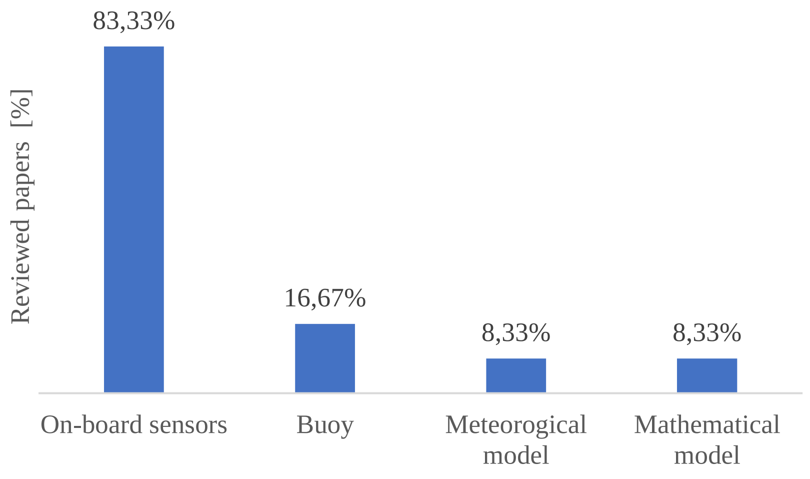


Figure 2.25 Distribution of data-acquisition methods reported in the reviewed papers

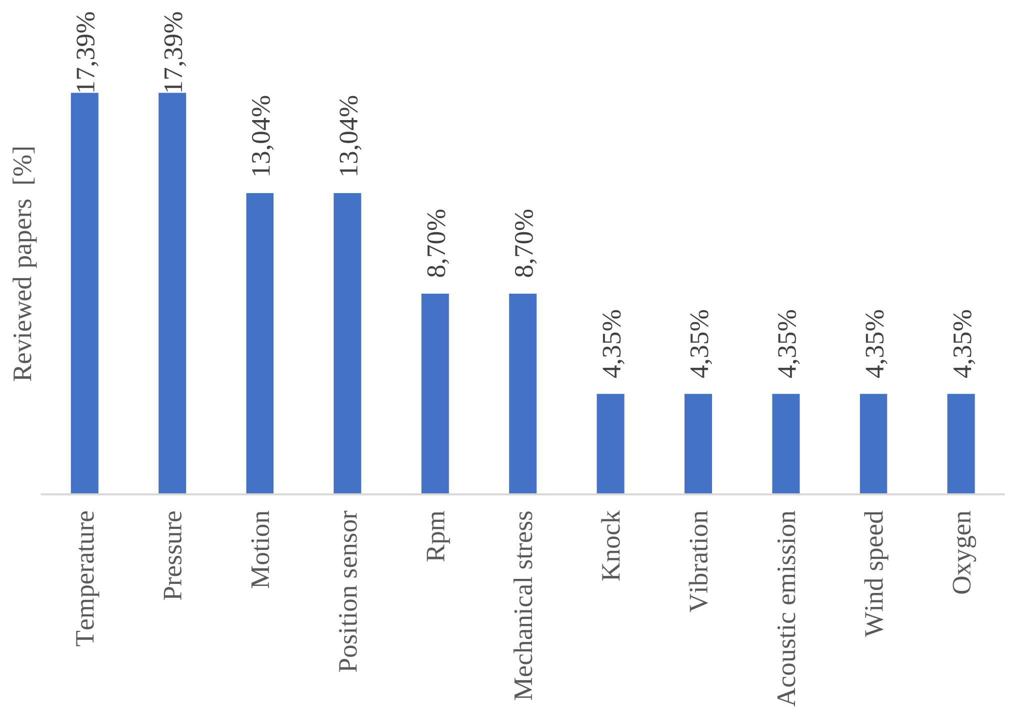


Figure 2.26 Classification of onboard sensor types used in ANN-based maritime applications

2.2.5.2. Data processing tools

The reviewed literature shows that a variety of software environments, frameworks, and online platforms were employed to process and analyse the data used in ANN-based maritime applications. Among these, MATLAB, Python, TensorFlow, Keras, and PyTorch are the most widely used tools. TensorFlow, an open-source library developed by Google for machine learning applications, was applied in several studies, including [148] and [156].

Similarly, Keras, an open-source high-level deep learning interface that integrates seamlessly with Python, was utilized in [118] and [148] for developing and training ANN architectures.

Python itself, often in combination with PyTorch, a flexible and widely adopted deep learning framework, was used in [115] and [124]. PyTorch provides similar functionality to TensorFlow but allows for more dynamic computation-graph manipulation, which makes it suitable for recurrent architectures and experimental model development.

Several studies also relied on MATLAB, either as the main programming environment or as a supplementary analysis tool, as in [115], [158], and [171]. In particular, the MATLAB Neural Network Toolbox was applied in [169] to construct neural models, while the results were statistically validated using IBM SPSS Statistics.

Comparable statistical post-processing was also conducted in [162] using Statistica software, while Simulink, an integrated MATLAB environment, was employed in [138] to develop simulation models for testing control algorithms.

The analysis of the reviewed studies that explicitly defined their processing environments shows that 75 % of the papers used standalone software, while the remaining 25 % relied on frameworks and platforms.

As illustrated in Figure 2.27, MATLAB/Simulink and Python are the dominant software tools, followed by Keras, IBM SPSS, and Statistica. Among the frameworks and platforms, TensorFlow and PyTorch were the most prevalent, each appearing in half of the cases where frameworks were reported.

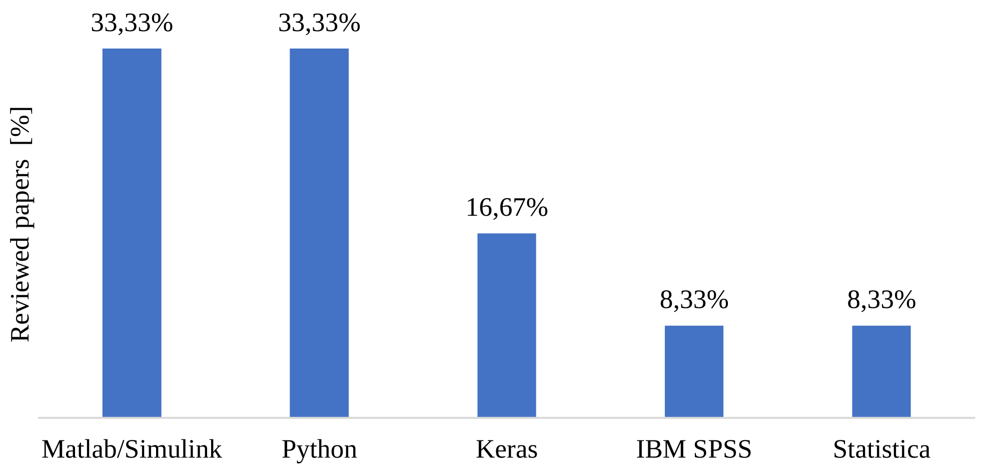


Figure 2.27 Distribution of software tools, frameworks, and platforms used for data processing and ANN model development in the reviewed papers

2.2.5.3. Data organization in training, validation and test data sets

A detailed analysis of the reviewed publications reveals consistent patterns in how datasets are partitioned for training, validation, and testing of ANN models. Most studies follow the conventional practice of assigning the majority of data to the training stage, while smaller portions are reserved for model verification and evaluation.

As shown in Figure 2.28, the training dataset most frequently contains between 70 % and 80 % of the total available data, a configuration observed in 72.22 % of the reviewed literature.

A somewhat larger allocation, where 81 %-90 % of the data is used for training, appears in 22.22 % of studies. Conversely, using less than 70 % of data for training is relatively uncommon.

Regarding validation datasets, most authors employ 10 %-20 % of the total data for model fine-tuning, as depicted in Figure 2.29. A smaller subset of works either omit a validation set entirely or allocate 21 %-40 % of the data to validation. Allocating less than 10 % for validation is rare among the reviewed studies.

For test datasets, the predominant configuration also includes 10 %-20 % of the total data, as illustrated in Figure 2.30. However, a notable number of studies either did not specify a separate testing phase or used no explicit test dataset at all. Only a few cases used smaller (<10 %) or larger (>20 %) test partitions, indicating that while data segmentation practices are broadly aligned with machine-learning conventions, clear documentation of data organization remains inconsistent across maritime ANN research.

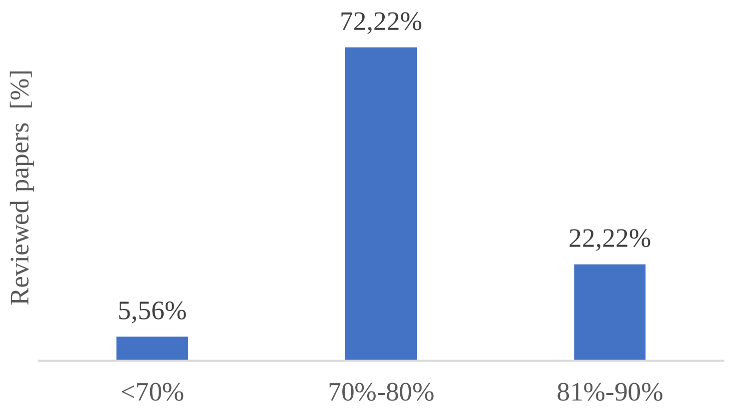


Figure 2.28 Distribution of training dataset proportions used in reviewed papers

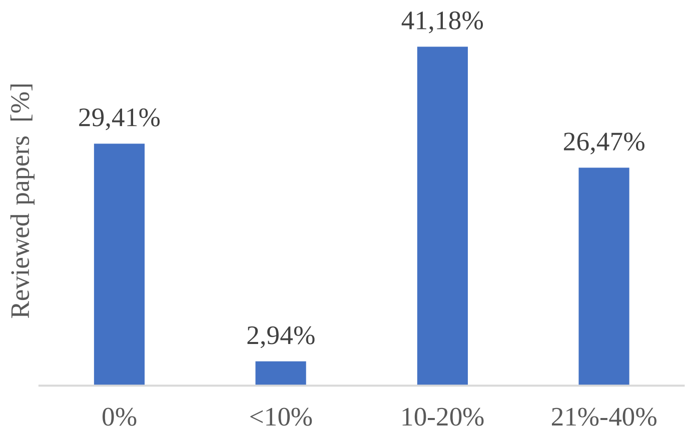


Figure 2.29 Distribution of validation dataset proportions used in reviewed papers

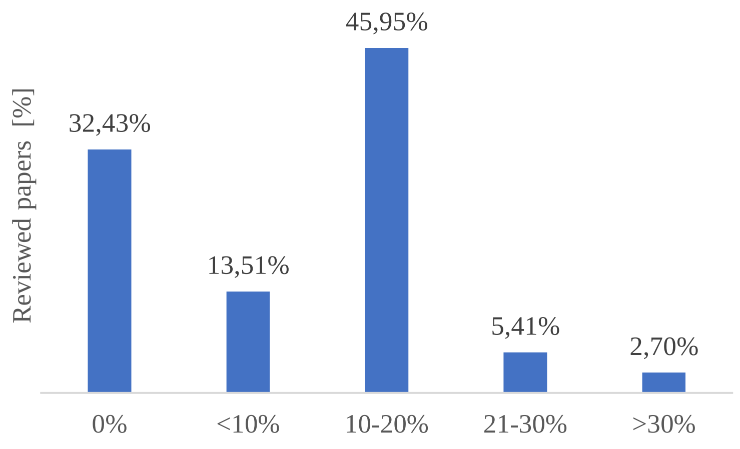


Figure 2.30 Distribution of test dataset proportions used in reviewed papers

2.2.5.4. Network architecture analysis

An analysis of network architectures described in the reviewed literature was carried out wherever sufficient data were available. Because detailed architectural information was often not reported, the analysis primarily focuses on feedforward Artificial Neural Networks (FFANNs), which constitute the majority of explicitly documented cases. The results indicate that the three-layer FFANN, consisting of a single hidden layer between the input and output, is by far the most common structure used in maritime applications.

This architecture appears to offer an effective compromise between model simplicity, computational efficiency, and approximation capability. The next most frequently encountered configuration is the four-layer FFANN, which contains two hidden layers. Networks with more than two hidden layers are relatively rare, as shown in Figure 2.31, suggesting that deeper topologies are not yet a prevailing trend in maritime ANN implementations.

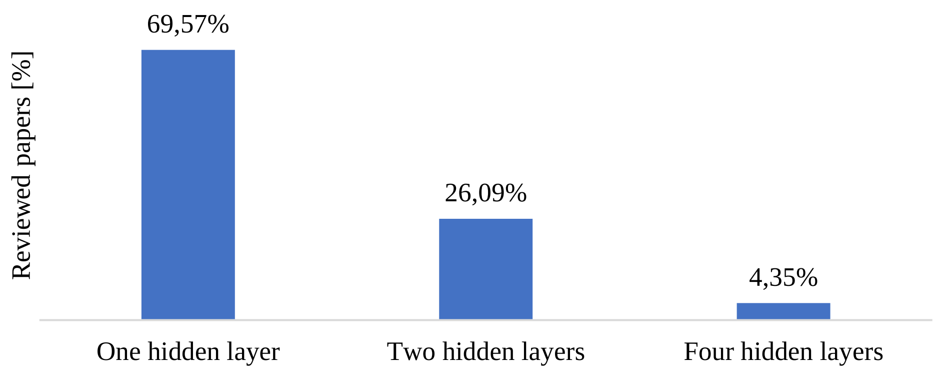


Figure 2.31 Distribution of number of hidden layers in reviewed papers

As illustrated in Figure 2.32, the typical number of neurons per hidden layer reported in the literature is ten, though values ranging from four to fifty appear depending on the modelling objective and data dimensionality. It is important to note, however, that the reviewed papers do not provide a consistent methodology for determining the optimal number of hidden neurons but one of the options is to use BR algorithm which optimises the number of neurons. Consequently, no clear rule or generalizable pattern could be identified regarding the optimal FFANN architecture for maritime applications, i.e., the choice remains case-specific and guided primarily by empirical testing.

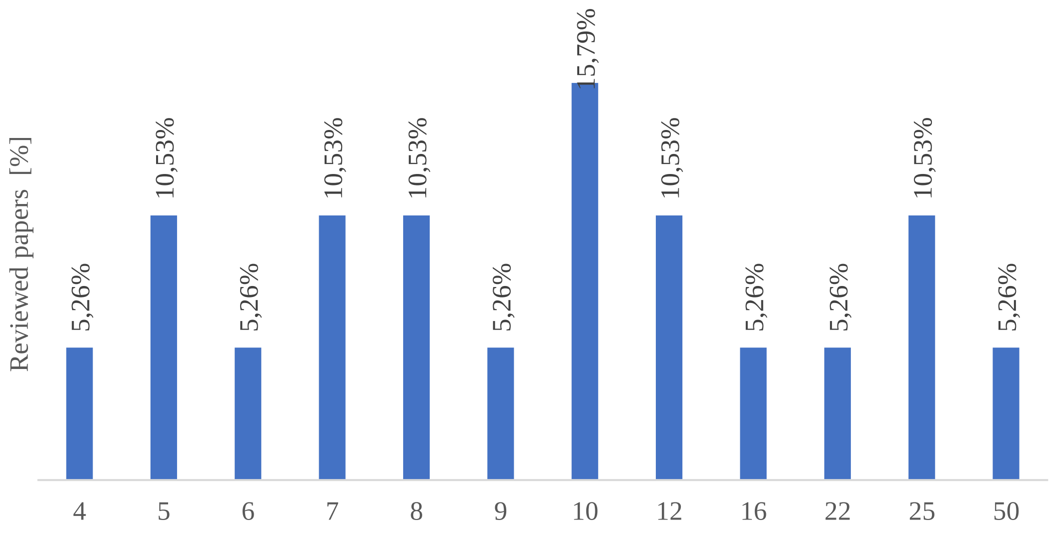


Figure 2.32 Distribution of number of neurons in hidden layers in reviewed papers

2.2.6. Identified research gaps in ANN applications

Despite the growing number of studies addressing Artificial Neural Network (ANN) applications in the maritime domain, the overall analysis reveals several persistent research gaps that limit wider adoption of these techniques for shipboard systems and operational decision-making. A major observation emerging from this review is the uneven distribution of ANN research across maritime applications.

As summarized earlier, the majority of published works are concentrated in image classification and ship-type identification, condition monitoring, and route optimization. In contrast, significantly fewer studies investigate real-time control, autonomous decision support, or system-level optimization, which represent the next logical stage of digital transformation at sea.

One of the most evident gaps lies in the application of ANN-based controller tuning.

While classical PID controllers are still dominant in ship automation, very few studies have leveraged neural networks for control system design.

The examples found in this review, such as ANN-supported PID tuning for stabilization fins [149] or neural network-based ship crane control [152], remain isolated prototypes rather than generalized frameworks.

Moreover, most of these implementations rely on offline training using simulated data, without closed-loop online adaptation or hardware-in-the-loop validation, leaving a wide gap between simulation-level proof of concept and shipboard deployment.

Another research gap concerns the integration of hybrid models using data and physics-based approaches. While ANNs have proven effective in pattern recognition and regression tasks, they are often detached from the underlying physical models that govern ship dynamics, propulsion systems, or energy flows. Combining neural network architectures with concepts such as digital twins could significantly enhance model interpretability and robustness, especially in safety-critical maritime operations.

However, the lack of guidelines on data acquisition and modelling approaches makes process more difficult. Most reviewed studies rely on proprietary or small-scale experimental datasets, often with inconsistent data organization, validation splits, and performance metrics. This constraint makes cross-comparison between models difficult and limits reproducibility.

2.3. SYSTEM IDENTIFICATION AND MODELLING APPROACHES

System identification plays a central role in developing dynamic models for monitoring, control, and Digital Twin applications. In practice, systems evolve over time and their properties change due to internal factors such as wear and aging, as well as variations in operating conditions and external influences. Consequently, continuous or periodic model updating is beneficial as it ensures that the model remains accurate and representative of the current system condition. In this context, system identification based on measured data provides a meaningful and effective approach for maintaining up-to-date models. It involves creating mathematical representations of a system based on measured input-output data, with the objective of capturing the essential dynamics relevant for simulation and controller design. Identification methods are commonly categorised into white-box, grey-box, and black-box approaches, depending on the degree of reliance on physical knowledge versus data-driven learning. Regardless of the modelling framework, the identification process typically includes several key steps:

- experiment design and data acquisition,
- pre-processing and feature extraction,
- model structure selection, parameter estimation,
- model validation.

The following subsections provide an overview of the main modelling approaches relevant to this research.

2.3.1. Data-driven modelling principles

Data-driven modelling refers to the class of identification approaches in which system behaviour is extracted directly from measured input-output data, without relying on an explicit physical model of the underlying process. In these methods, the available data are treated as the primary source of information about the system dynamics, and the modelling task involves selecting an appropriate structure and estimating the parameters that best reproduce the observed responses.

A fundamental consideration in data-driven modelling is the distinction between linear and nonlinear model structures. Linear models, such as transfer functions and state-space representations, are well established, computationally efficient, and often sufficient for systems that exhibit near-linear behaviour around a fixed operating region as discussed in [177]. Nonlinear models such as ANNs offer greater flexibility and can capture complex relationships, but require more extensive datasets and careful validation to avoid overfitting [178].

Successful application of data-driven approaches depends heavily on the quality and representativeness of the collected data. This includes proper experiment design, adequate excitation of system dynamics, avoidance of multicollinearity, and the use of appropriate sampling rates. Pre-processing is also critical, including filtering, detrending, normalization, and segmentation into training, validation, and testing subsets. These steps help ensure that the identified model generalises well rather than simply memorising noise or isolated patterns [179].

An important aspect of data-driven modelling is multi-step prediction versus one-step prediction. One-step prediction evaluates the model's ability to estimate the next sample given the current state, while multi-step prediction assesses how errors accumulate over time when the model is used recursively, an aspect particularly relevant for controller design and Digital Twin applications [180].

Model validation commonly includes metrics such as MSE, RMSE, GoF, correlation tests, and residual analysis, which provide insight into the accuracy and reliability of the identified model.

In summary, data-driven modelling provides a flexible framework for identifying dynamic systems from measurements alone. It offers the foundation for both linear identification methods, such as transfer function modelling, and nonlinear approaches, such as ANN-based NARX models, which are addressed in the following sections.

2.3.2. Transfer function modelling

Transfer function modelling is one of the most widely used linear system identification approaches and provides a compact representation of input–output dynamics in the Laplace domain. A transfer function expresses the process behaviour as a ratio of polynomials in the complex variable s as expressed by Equation (2.47).

$$G(s) = \frac{b_0 + b_1 s + \cdots + b_m s^m}{1 + a_1 s + \cdots + a_n s^n} \tag{2.47}$$

Where the coefficients b_i and a_i define the zeros and poles of the model, while m and n denote the orders of the numerator and denominator polynomials. This structure is well suited for flow-control processes due to its transparency, low computational complexity, and compatibility with classical controller design methods as discussed in [181] and [182].

Identifying a suitable transfer function model involves selecting an appropriate structure and estimating its parameters from measured input-output data. Structure selection consists of choosing the number of poles, zeros, and potential time delays that sufficiently capture the system dynamics.

Although transfer functions are commonly expressed in the continuous domain, their parameters are typically estimated from discrete-time input-output data. A widely used linear model structure for discrete-time identification is expressed by Equation (2.48).

$$y(k) + a_1 y(k-1) + \ldots + a_n y(k-n) = b_1 u(k-1) + \ldots + b_m u(k-m) \tag{2.48}$$

Where $u(k)$ and $y(k)$ denote the sampled input and output, respectively. Parameter estimation for such models can be performed using several identification techniques, including Least Squares (LS) [183], Instrumental Variables (IV) [184], and Prediction Error Methods (PEM) [185].

The LS estimator is simple and efficient but sensitive to measurement noise, especially when noise corrupts both the input and output signals. To mitigate this issue, the IV method introduces an instrument matrix Z, composed of signals correlated with the regressors but uncorrelated with measurement noise. The IV estimator is defined by Equation (2.49).

$$\hat{\theta}_{IV} = (\Phi^T Z)^{-1} Z^T y \tag{2.49}$$

Where Φ is the regressor matrix and y is the output vector. This approach significantly reduces bias in the estimated parameters and improves robustness when dealing with noisy flow measurements, which are common in marine and hydraulic systems.

The estimation problem in system identification is often nonlinear in its parameters, particularly when time delays or higher-order dynamics are involved. For this reason, iterative numerical optimisation algorithms such as the Gauss-Newton method were used in this research. The iterative Gauss–Newton update is given by Equation (2.50).

$$\theta_{k+1} = \theta_k - (J^T J)^{-1} J^T e \tag{2.50}$$

Where J is the Jacobian of the prediction error vector e with respect to the parameters θ. This procedure refines the parameter estimates by iteratively minimising the sum of squared prediction errors, yielding a more accurate model representation.

Validation of transfer function models is a critical step in the identification process. Time-domain and frequency-domain evaluation metrics, such as mean squared error (MSE), goodness of fit (GoF), best-fit percentage, residual analysis, and correlation tests, are used to assess the accuracy and generalisation capability of the model.

Despite their advantages, transfer function models exhibit limitations when applied to marine flow-control systems. These processes often display significant nonlinearities, actuator constraints, and operational variations that cannot always be captured by linear structures. For this reason, transfer function modelling in this research is complemented by nonlinear identification approaches, such as Artificial Neural Network (ANN)-based NARX models, which are discussed in the following subsection.

2.3.3. Artificial Neural Network modelling

Artificial Neural Networks (ANNs) represent a class of nonlinear, data-driven modelling techniques inspired by the structure and generalization properties of a biological neuron, and by the way data are stored in biological neural systems. Their ability to approximate complex nonlinear functions makes them well suited for dynamic modelling in situations where first-principles i.e., white-box or mathematical descriptions are incomplete, overly complex, or impractical to derive. Given the sufficient number of neurons, ANNs can approximate any continuous nonlinear function [186]. For dynamic systems, this approximation must account not only for instantaneous relationships but also for temporal dependencies. One of the most widely used architectures for such applications is the nonlinear autoregressive network with exogenous inputs (NARX) model. The NARX model represents the system output as a nonlinear function of past outputs and past inputs according to Equation (2.51).

$$y(k) = F\big(y(k-1), \dots, y(k-n_y), u(k-1), \dots, u(k-n_u)\big) \tag{2.51}$$

Where n_y and n_u denote the number of output and input delays, respectively, and $F(\cdot)$ is the nonlinear mapping implemented by the neural network. This structure explicitly incorporates dynamic memory and is particularly suitable for processes such as flow regulation, where the system response exhibits delayed and nonlinear behaviour.

Training an ANN-based NARX model involves adjusting the network parameters i.e., weights and biases to minimise the discrepancy between the predicted and measured outputs. This is typically achieved using gradient-based optimisation methods such as Levenberg-Marquardt, scaled conjugate gradient, or Adam optimisers. Reliable model development requires careful preparation of the dataset, including segmentation into training, validation, and testing subsets, normalisation or standardisation of input and output variables, and the use of multiple training initialisations to mitigate local minima effects. Overfitting prevention measures such as early stopping, regularisation, or reducing network complexity are also essential to ensure that the model generalises well to unseen data.

Validation of ANN-based models involves more than achieving low error on the training dataset. Multi-step prediction tests, residual analysis, and performance evaluation on independent datasets provide insight into the model's ability to reproduce system dynamics under varying conditions. These validation procedures are particularly important for control-oriented applications and Digital Twin development, where the model must remain accurate outside the specific conditions used during identification [25].

ANN-based modelling offers several advantages for nonlinear system representation, including high flexibility, strong approximation capability, and the ability to capture complex relationships without requiring an explicit physical or mathematical model. However, these benefits come with challenges. Neural networks typically demand large and representative datasets, require careful tuning of hyperparameters, and may exhibit reduced interpretability compared with linear models. In addition, training can be computationally intensive, and poor data quality may lead to unstable or unreliable models.

Despite these limitations, ANN-based NARX models represent a powerful tool for capturing nonlinear dynamics in marine systems. Their use in this research provides an alternative modelling pathway that could support controller tuning and Digital Twin applications.

2.3.4. Hybrid and Grey-Box Modelling

Hybrid and grey-box modelling approaches combine elements of first-principles modelling (white-box) with data-driven (black-box) identification to obtain models that balance physical interpretability with the flexibility required to capture complex system behaviour. These approaches have gained increasing attention in recent years, particularly within the context of Digital Twins, where accurate, real-time representations of physical systems must integrate both known physical laws and unknown or nonlinear dynamics derived from operational data.

In a traditional grey-box model, part of the system structure is defined by physical equations, while the remaining components, typically terms that represent unmodeled dynamics, friction, actuator nonlinearities, or sensor distortions, are identified from measured data. This reduces the parameter space to be estimated and improves interpretability compared with purely data-driven methods. For flow-control systems, grey-box models often incorporate conservation laws, valve characteristics, or hydraulic relationships, while allowing data-driven terms to capture phenomena not explicitly represented by these equations.

Hybrid models extend this concept by combining physics-based and data-driven components in parallel or series architectures. A common formulation involves using a physical model to estimate baseline behaviour and a data-driven model, such as an ANN or NARX structure, to model residual errors according to Equation (2.52).

$$y(k) = y_{phys}(k) + y_{data}(k) \tag{2.52}$$

Where $y_{phys}(k)$ represents the output of the physics-based model and $y_{data}(k)$ captures the remaining discrepancy using a machine-learning component. This additive framework allows the hybrid model to retain physical meaning while adapting to nonlinearities, parameter variations, and real-world uncertainties.

Another hybrid structure applies the data-driven component as a corrective term to the model parameters rather than directly to the output according to the Equation (2.53).

$$\theta(k) = \theta_{nom} + \Delta\theta_{ANN}(k) \tag{2.53}$$

Where $\Delta\theta_{ANN}(k)$ is learned from data to update the nominal parameter set θ_{nom}. Such methods are used in Digital Twin applications where online adaptation and model updating are required.

Hybrid and grey-box models bring several advantages: they integrate physical interpretability with the flexibility of machine learning, they often require less training data than fully black-box models [61], they support scenario testing by preserving structural properties of the system, and they enable real-time updating, an essential feature of Digital Twin architectures.

However, these models also present challenges. Constructing an appropriate hybrid structure requires expert knowledge of both the physical system and machine-learning methods, parameter identifiability may become an issue, and additional computational overhead is introduced when physical and data-driven components must be solved simultaneously. For marine flow-control systems, challenges include varying operating conditions, valve nonlinearities, and hydraulic delays.

Despite these challenges, hybrid and grey-box approaches represent an important direction for improving fidelity in Digital Twins. They provide a pathway to combine the strengths of transfer function models and ANN-based NARX models, particularly when neither category alone can fully capture the dynamics of the system under investigation.

2.4. PERFORMANCE EVALUATION METRICS

Performance metrics provide quantitative means to assess accuracy, reliability, and generalization capability of the models when predicting, classifying, or optimizing ship-related parameters. They allow comparison between different neural network types, training strategies, and data-processing approaches under uniform conditions.

Based on the reviewed literature, a total of twenty-nine performance measures have been identified and classified according to their frequency of appearance and field of application. The most frequently used measures include Accuracy, Mean Squared Error (MSE), Precision, Recall, and Root Mean Squared Error (RMSE), while other measures such as Average Precision (AP), Mean Absolute Error (MAE), and Relative Error are used less often. More specialized indices such as Signal-to-Clutter Ratio (SCR), Roll Reduction Rate (RRR), and Persistence Index (PI) have been reported in specific control or stability analyses. Each metric provides a distinct insight into model behaviour, from classification reliability to numerical stability or physical interpretability.

Accuracy is the most straightforward metric for evaluating classification models. It measures the ratio between correctly predicted outcomes and the total number of predictions made by the network. Although intuitive, accuracy can become misleading in imbalanced datasets, where a model might achieve high accuracy by simply favouring the dominant class. As defined in Equation (2.54), accuracy is computed as the ratio of correct predictions to all predictions.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{2.54}$$

Where T_P and T_N represent true positive and true negative predictions, respectively, while F_P and F_N correspond to false positives and false negatives.

The Mean Squared Error (MSE) is an evaluation metric for regression and prediction problems. It represents the average of the squared differences between the actual and predicted values, providing a measure of how close the predictions are to the real observations as defined in Equation (2.55). Smaller MSE values indicate more accurate models, but the squaring operation gives higher weight to large errors, which makes the metric sensitive to outliers.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.55}$$

Where y_i denotes the true observed value, $\hat{y}$ the predicted value, and n the total number of observations.

The Root Mean Squared Error (RMSE), calculated according to Equation (2.56), is a widely used metric for evaluating regression models, providing an interpretable measure of the average magnitude of prediction errors.

It is derived as the square root of the Mean Squared Error (MSE) and maintains the same units as the output variable, which makes it easier to interpret in practical contexts. RMSE penalizes larger errors more severely due to the squaring operation, which is useful when large deviations are undesirable.

$$RMSE = \sqrt{\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right)} \tag{2.56}$$

Where y_i denotes the true observed value, $\hat{y}$ the predicted value, and n the total number of observations. Lower RMSE values indicate better model performance, with an RMSE of zero corresponding to perfect prediction.

RMSE to standard deviation ratio (RSR) standardizes RMSE using the standard deviation of observations, providing a normalized efficiency index that simplifies model comparison across datasets. It is calculated according to Equation (2.57) where values near 0 indicate excellent performance, while values near 1 suggest weak predictive ability.

$$RSR = \frac{RMSE}{\sigma_y} \tag{2.57}$$

Where RMSE represents the root mean square error and σ_y represents standard deviation of the actual values.

Precision is an evaluation metric in classification tasks, particularly in situations where false positives are costly. It quantifies the proportion of correctly predicted positive observations among all predicted positive cases, indicating the model's ability to avoid false alarms. High precision means that when the model predicts a positive outcome, it is usually correct. Precision is calculated according to Equation (2.58).

$$Precision = \frac{TP}{TP + FP} \tag{2.58}$$

Where T_P and F_P represent true positive and false positive predictions, respectively. While precision measures the accuracy of positive predictions, it does not consider the false negatives, making it most effective when paired with recall.

Recall, also known as sensitivity or true positive rate, measures the proportion of correctly identified positive samples out of all actual positive cases.

It reflects the model's ability to detect positive outcomes and is particularly important in applications where missing a positive instance has serious consequences. It is calculated according to Equation (2.59).

$$Recall = \frac{TP}{TP + FN} \tag{2.59}$$

Where T_P and F_N represent true positive and false negative predictions. A high recall indicates that most of the positive samples are correctly identified, although this may sometimes come at the cost of lower precision.

Average Precision (AP), calculated according to Equation (2.60) is a performance metric commonly used in object detection and classification tasks to evaluate how well a model ranks positive predictions across different confidence thresholds. It summarizes the trade-off between precision and recall by computing the area under the precision-recall curve. This makes

AP particularly suitable for models that output probabilistic confidence scores rather than binary classifications.

$$AP = \int_0^1 P(R)dR \tag{2.60}$$

Where *P(R)* represents precision as a function of recall *R*. In discrete implementations, AP is often approximated by summing the precision values at various recall levels. Higher AP values indicate that the model maintains strong precision across a wide range of recall levels, which is especially important in maritime image detection and classification tasks.

The Mean Absolute Error (MAE), calculated by Equation (2.61), is a fundamental evaluation metric for regression models, measuring the average magnitude of prediction errors without considering their direction.

Unlike MSE or RMSE, MAE treats all errors equally by using absolute differences, providing a more intuitive measure of average deviation.

It is particularly valuable when the presence of large outliers should not disproportionately influence the evaluation.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.61}$$

Where y_i denotes the true observed value, $\hat{y}$ the predicted value, and *n* the total number of observations. A lower MAE value indicates better predictive accuracy, with zero denoting a perfect fit. Because MAE is expressed in the same units as the target variable, it is often preferred for its interpretability in real-world maritime prediction contexts.

Relative Error, calculated according to Equation (2.62), quantifies the magnitude of prediction error relative to the true value, providing a normalized indication of model performance that is independent of the variable's scale.
It is particularly useful when comparing prediction accuracy across datasets with differing magnitudes, such as ship speed, temperature, or fuel consumption measurements.

$$RE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(y_i - \hat{y}_i)}{y_i}\right| \tag{2.62}$$

Where y_i denotes the true observed value, $\hat{y}$ the predicted value, and *n* the total number of observations. The use of the absolute term ensures that over-predictions and under-predictions do not cancel each other out.

A smaller RE value signifies higher predictive accuracy, while large values indicate poor model performance or potential scaling issues.

The R Score measures the strength and direction of the linear relationship between predicted and observed values. It reflects how closely the model's predictions follow the actual data trend, with values ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation). An R value close to zero indicates little or no linear correlation. It is calculated according to Equation (2.63).

$$R = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}\ \sqrt{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2}} \tag{2.63}$$

Where y_i represents the actual value, $\hat{y}_i$ the predicted value, $\bar{y}$ the mean of observed values, and $\bar{\hat{y}}$ the mean of predicted values.

A high positive R value indicates that the ANN model captures the relationship between variables effectively, while negative values suggest an inverse relationship.

Mean Average Precision (mAP), calculated according to Equation (2.64), is a widely used evaluation measure for object detection and classification tasks. It averages the precision values obtained across multiple recall levels, providing an overall measure of model performance in detecting and classifying objects. In maritime applications, mAP is often used to assess the accuracy of ship-detection CNNs operating on optical or SAR imagery.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{2.64}$$

Where AP_i represents average precision for class i, and N_c represents total number of object classes or categories considered. Higher mAP values indicate better overall detection and classification consistency across all categories.

The F1 Score is the harmonic mean of precision and recall, balancing the two measures to provide a single evaluation criterion particularly suitable for imbalanced datasets. Calculated according to Equation (2.65), an F1 score of 1 represents perfect precision and recall, while 0 indicates complete failure in either or both.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.65}$$

The coefficient of determination, R², measures how well predicted values explain the variability of observed data. Calculated according to Equation (2.66), it quantifies the proportion of variance in the dependent variable that is predictable from the independent variables.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.66}$$

Where y_i represents the actual value, $\hat{y}_i$ the predicted value, and $\bar{y}$ the mean of actual values. An R² value close to 1 signifies a strong predictive model, whereas values near 0 indicate poor explanatory power.

Frames per second (FPS) measures the computational efficiency of models used for real-time ship detection or tracking.

It quantifies how many frames the network processes per second, as defined in Equation (2.67).

$$FPS = \frac{N_f}{t_p} \tag{2.67}$$

Where N_f is the total number of frames processed and t_p the total processing time in seconds. Higher FPS values indicate faster model inference, critical for onboard surveillance and navigation systems

Intersection over union (IoU) evaluates the overlap between the predicted and the ground-truth bounding boxes in object-detection models. It is defined as the ratio between the area of overlap and the area of union between the two bounding boxes, as shown in Equation (2.68).

$$IoU = \frac{A_{overlap}}{A_{union}} \tag{2.68}$$

Where $A_{overlap}$ represents the intersected area between predicted and actual bounding boxes, and A_{union} represents the total area covered by both boxes. Higher IoU values indicate more accurate object localization.

Cross-Entropy (CE), calculated according to Equation (2.69), measures the divergence between predicted probabilities and true class labels. It is used as a loss function in classification tasks to quantify how well the model's predicted probability distribution matches the true distribution.

$$CE = -\frac{1}{n}\sum_{i=1}^{n} y_i \ln(\hat{y}_i) + (1 - y_i)\ln(1 - \hat{y}_i) \tag{2.69}$$

Where y_i represents the actual value and $\hat{y}_i$ the predicted value. A smaller CE value corresponds to better classification accuracy and probability calibration.

Standard deviation, defined by Equation (2.70), quantifies the dispersion of model errors around their mean, describing model stability.

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(E_i - \bar{E})^2} \tag{2.70}$$

Where E_i represents individual prediction error and $\bar{E}$ represents mean of prediction errors.

Absolute Error, calculated according to Equation (2.71), measures the magnitude of deviation between predicted and true values without considering direction.

$$AE_i = |y_i - \hat{y}_i| \tag{2.71}$$

Where y_i represents the actual value and $\hat{y}_i$ the predicted value. AE forms the basis of aggregated metrics such as MAE and RMSE.

Average intersection over union (AIoU) averages IoU scores across multiple objects or classes to summarize global detection performance, according to Equation (2.72).

$$AIoU = \frac{1}{N}\sum_{i=1}^{N} IoU_i \tag{2.72}$$

Where IoU_i represents the intersection over union for the object i, and N represents total number of objects. Higher AIoU values indicate more consistent detection accuracy.

Error autocorrelation coefficient, calculated according to Equation (2.73), measures the correlation of residuals across time steps, evaluating whether prediction errors are independent.

$$r_k = \frac{\sum_{t=k+1}^{n}(e_t - \bar{e})(e_{t-k} - \bar{e})}{\sum_{t=1}^{n}(e_t - \bar{e})^2} \tag{2.73}$$

Where r_k represents autocorrelation coefficient at lag k, e_t represents the error (residual) at time t, and $\bar{e}$ represents mean value of error (residual). Low $|r_k|$ values indicate desirable error independence.

Variance captures the average squared deviation of prediction errors, reflecting model uncertainty, as defined by Equation (2.74).

$$Var = \frac{1}{n}\sum_{i=1}^{n}(e_i - \bar{e})^2 \tag{2.74}$$

Where, e_i represents the error for sample i, and $\bar{e}$ represents mean value of error. Low variance denotes stable model performance.

Deviation measures the mean signed difference between predicted and actual values, indicating whether the model systematically over-predicts or under-predicts. A positive value means underestimation, while a negative one means overestimation. It is calculated according to equation (2.75).

$$D = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i) \tag{2.75}$$

Where y_i represents the actual value and $\hat{y}_i$ the predicted value.

Mean absolute percentage error (MAPE) expresses the prediction error as a percentage of the actual value, allowing for intuitive interpretation of model performance across variables with different scales. Calculated according to Equation (2.76), it is especially useful in operational contexts where relative deviation matters more than absolute error.

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{(y_i - \hat{y}_i)}{y_i}\right| \tag{2.76}$$

Where y_i represents the actual value and $\hat{y}_i$ the predicted value. Lower MAPE values indicate better predictive accuracy, while higher values suggest greater deviation from actual data.

The Signal-to-Clutter Ratio (SCR), calculated according to Equation (2.77), evaluates the clarity of detected signals relative to surrounding noise, especially useful in radar-based maritime applications.

$$SCR = 10log_{10}\left(\frac{P_s}{P_c}\right) \tag{2.77}$$

Where P_s represents power of the target signal and P_c represents power of the clutter or background noise. A higher SCR indicates better distinction between target and background.

The Roll reduction rate (RRR) quantifies the percentage decrease in roll amplitude, achieved through control or stabilization methods. It is calculated according to Equation (2.78) and often used for assessing active fin performance.

$$RRR = \left(\frac{A_0 - A_c}{A_0}\right) \cdot 100 \tag{2.78}$$

Where A_0 represents mean roll amplitude without control and A_c represents mean roll amplitude with control applied.

Fischer’s correlation factor, calculated according to Equation (2.79), evaluates correlation between predicted and measured parameters such as sea-state estimation or vibration response. It represents the squared correlation coefficient, used to assess linear dependency.

$$F = \frac{\left(\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})\right)^2}{\left(\sum_{i=1}^{n}(y_i - \bar{y})^2\right)\left(\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2\right)} \tag{2.79}$$

Where y_i represents the actual value, $\hat{y}_i$ the predicted value, $\bar{y}$ the mean of actual values, and $\bar{\hat{y}}$ the mean of predicted values.

The 95th percentile error, calculated according to Equation (2.80), defines the error threshold below which 95% of prediction errors fall, describing the tail behaviour of model deviations.

$$E_{95} = Q_{0.95(|y_i - \hat{y}_i|)} \tag{2.80}$$

Where $Q_{0.95}$ represents 95th percentile function, y_i represents the actual value, and $\hat{y}_i$ represents the predicted value.

Percent Bias (PB), calculated according to Equation (2.81), measures the average tendency of predicted values to be larger or smaller than actual values, expressed as a percentage. Negative values indicate overestimation, while positive values indicate underestimation.

$$PB = 100 \cdot \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)}{\sum_{i=1}^{n} y_i} \tag{2.81}$$

Where y_i and $\hat{y}_i$ represent actual and predicted values, respectively.

The Persistence Index (PI), calculated according to Equation (2.82), evaluates model performance relative to a baseline persistence model that assumes no change between successive time steps. Values close to 1 denote superior predictive skill.

$$PI = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - y_{i-1})^2} \tag{2.82}$$

Where y_i and $\hat{y}_i$ represent actual and predicted values, respectively.

2.5. SUMMARY OF LITERATURE REVIEW FINDINGS

The concept of the Ship's Digital Twin (SDT) has gained significant traction in recent maritime research. The reviewed literature shows that DT technology is still in the early maturity stage for ships, with most implementations focused on monitoring, simulation, and diagnostics rather than on closed-loop control.

Current SDT applications primarily address power systems and energy efficiency (23%), hull structure health (18%), propulsion system (18%), and voyage performance optimisation (9%). These systems rely on extensive sensor networks and IoT infrastructure to feed physical data into simulation or machine-learning models, providing insight into ship behaviour under varying operational conditions.

Although the benefits of SDTs are well recognised, including real-time situational awareness, predictive maintenance, and life-cycle performance optimisation, several technical and methodological challenges remain. Studies highlight a lack of interoperability between simulation environments and operational systems, and real-time bidirectional data coupling between the digital and physical twins.

In summary, the current body of research portrays SDT as a rapidly developing yet fragmented domain, whose evolution depends on unified frameworks for data integration and model standardisation.

Furthermore, the literature analysis revealed an extensive and growing application of Artificial Neural Networks (ANNs) in maritime systems. Over the last decade, ANNs have been adopted as data-driven tools for solving complex problems involving prediction, classification, fault detection, and system modelling.

The most common applications include ship image classification and type identification (24,64%), condition monitoring and maintenance (17,39%), and ship route prediction (14,49%). These use cases typically benefit from the availability of large, structured datasets such as AIS records or sensor logs. Conversely, controller tuning (5,8%), autopilot design (8,7%), and gas-emission prediction (5,8%) remain under-explored, reflecting challenges in collecting representative dynamic data and ensuring model transparency for safety-critical applications.

Among network architectures, the FFANN (36,36%) and the CNN (33,77%) dominate, followed by RNN (12,99%) and RBF (6,49%) networks. More advanced architectures, such as LSTM and BTNN (1,3%), are beginning to appear, mainly in sequence-modelling or uncertainty-aware applications.

Training is typically performed using the BP algorithm (21,79%), with Levenberg-Marquardt (12,82%) and Adam optimisers (8,97%) frequently used for faster convergence. Other methods such as DBSCAN (3,85%), K-means (3,85%), and Genetic Algorithms (2,56%) occur mostly in hybrid models.

The most common activation functions are sigmoid and ReLU, while evaluation metrics are dominated by Accuracy (13,93%), MSE (12,3%), Precision (9,02%), and RMSE (6,56%), reflecting a preference for traditional statistical validation techniques. Domain-specific measures such as Roll Reduction Rate (0,82%) or Signal-to-Clutter Ratio (0,82%) demonstrate a gradual move toward application-oriented performance assessment.

Most studies rely on on-board sensor data (14,49%) such as temperature, vibration, pressure, or motion, often combined with AIS trajectories or simulation outputs. Common data splits of 70-20-10 or 80-10-10 percent for training, validation, and testing (72,22%) reflect emerging experimental standardisation.

Software usage is similarly concentrated around MATLAB/Simulink (33,33%) and Python (33,33%), along with other frameworks (16,67%) such as TensorFlow, Keras, and PyTorch.

Despite steady progress, several research gaps remain evident:

- There is no precise definition of an SDT, development guidelines or standardized frameworks to support its development.
- Limited integration between SDT and ANN approaches. Most digital twins rely on rule-based or physics-based models, while ANN-based learning frameworks are seldom embedded to enable adaptive, data-driven control.
- Under-representation of controller-oriented studies. ANN applications in controller tuning and autonomous decision-making are sparse, and SDT implementations rarely support closed-loop feedback or real-time optimisation.

- Insufficient interpretability and validation. ANN models often act as black boxes, and SDT validation frequently lacks experimental verification.

The absence of such elements highlights an evident gap in the literature and implies the need for further clarification to ensure consistent and meaningful use of the concept.

3. RESEARCH METHODOLOGY AND MODEL DEVELOPMENT

This chapter presents the methodological framework adopted in this research and outlines the process of digital twin development for controller tuning. The methodology is based on a quantitative and experimental approach, supported by transfer-function modelling, machine learning techniques, and validation using laboratory systems that emulate shipboard flow control dynamics.

The central idea of the research is that controller tuning may be performed not only on a physical system, but also on a sufficiently accurate model capable of replicating its dynamic behaviour. In the context of recent technological developments, such a model is referred to as a digital twin. The digital twin is therefore not treated only as a simulation, but as a functional twin capable of providing reliable performance for controller tuning. To investigate this concept, two modelling approaches are explored: Continuous-time transfer function models, representing linear system dynamics around an operating point and suitable for classical control design, and Nonlinear Autoregressive Exogenous (NARX) Artificial Neural Network models, capable of capturing nonlinear dependencies and temporal correlations in time-series data.

Both model types are trained and evaluated using datasets obtained from a laboratory flow control system. The development process includes data acquisition, model identification, performance evaluation, and iterative refinement of the model parameters. The models are not evaluated solely based on identification accuracy, but also for their practical applicability for controller tuning. This dual evaluation perspective makes it possible not only to evaluate model accuracy, but to determine whether digital twins can provide a safe and reliable alternative to tuning on real systems.

The overall research workflow is shown in Figure 3.1. To ensure clarity, reproducibility, and consistency across different modelling scenarios, the research methodology is structured into five phases. These phases reflect both the chronological order of the research and the functional relationship between conceptual development, data collection, model development, and validation activities.

Phase 1 – Research Planning

In the initial stage, the research topic is formulated and the scientific problem is defined. A broad review of available literature is performed in order to examine the digital twin concept, identify relevant modelling approaches, and determine gaps in current knowledge related to remote controller tuning. Based on these findings, the research questions, objectives, scope, and constraints are established. This stage also includes the selection of representative physical systems for model development relevant for fluid flow and maritime control applications.

Phase 2 – Methodology Design

The second phase focuses on developing the scientific strategy of the research. A more targeted literature review is conducted to evaluate suitable modelling techniques, system identification methods, and model validation procedures. This phase results in the adoption of a dual-modelling approach, i.e., analytical transfer function-based models and ANN-based NARX models. At this stage, performance indicators, controller tuning strategies, evaluation metrics, and software tools are selected to ensure that the methodology remains coherent and comparable throughout the research.

Phase 3 – Data Acquisition and Preparation

The third phase consists of recording representative input-output data from a laboratory flow control system. The experiments are designed to excite system dynamics in a controlled manner and capture relevant transient and steady-state behaviour. Data acquisition includes the selection of actuator signals, sampling rates, and sensor interfaces. The collected data are then pre-processed through segmentation into training, test, and validation subsets, and removal of corrupted or unrepresentative samples. This ensures that modelling tasks are performed on reliable, physically meaningful datasets.

Phase 4 – Model Development and Proof of Concept

In the fourth phase, the transfer function and ANN models are developed and evaluated. Transfer function models are estimated using instrumental variable techniques to counteract noise and measurement disturbance. In parallel, ANN-NARX models are developed and trained using combinations of delay sequences, Bayesian Regularization, k-fold cross-validation, and multi-start strategies to ensure generalization and robustness. Both modelling approaches are assessed using statistical measures, i.e., RMSE, Goodness of Fit, and NRMSE and examined for their suitability in controller tuning applications. This stage represents the proof-of-concept demonstration of digital twin applicability.

Phase 5 – Full-Scale Model Deployment and Hypothesis Testing

The final phase extends the methodology to a more complex or multivariable system. The objective is to determine whether the modelling and controller tuning results obtained in the proof-of-concept stage scale to systems with operational variability. The digital twin is evaluated not only in terms of its ability to reproduce system behaviour, but also in terms of its capacity to support remote controller tuning. Outcomes of this phase serve as the basis for hypothesis confirmation and assessment of digital twin applicability in maritime engineering contexts.

Together, these five phases provide a structured pathway for investigating digital-twin-assisted controller tuning, allowing the research to progress from concept development to practical validation on increasingly complex systems.

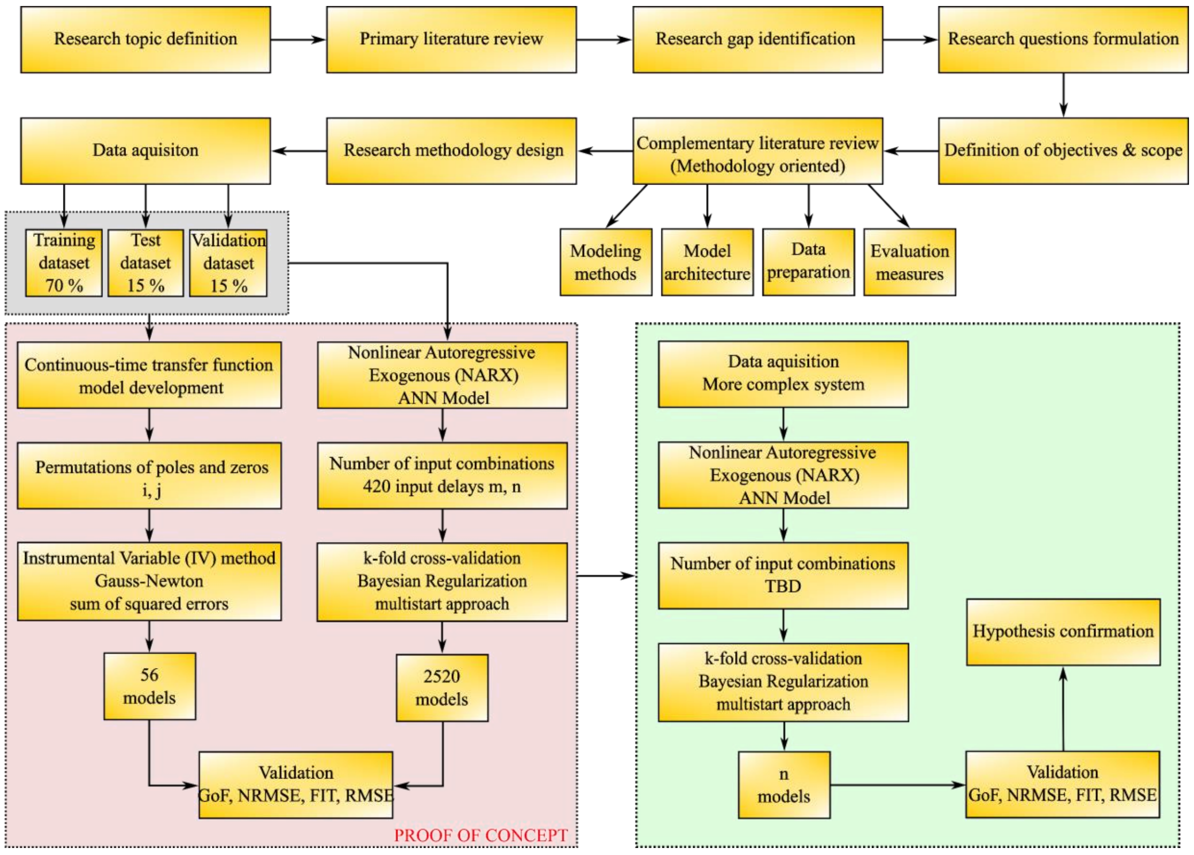

Research topic definition
Primary literature review
Research gap identification
Research questions formulation
Definition of objectives & scope
Complementary literature review (Methodology oriented)
Research methodology design
Data aquisiton
Modeling methods
Model architecture
Data preparation
Evaluation measures
Training dataset 70 %
Test dataset 15 %
Validation dataset 15 %
Continuous-time transfer function model development
Permutations of poles and zeros i, j
Instrumental Variable (IV) method Gauss-Newton sum of squared errors
56 models
Nonlinear Autoregressive Exogenous (NARX) ANN Model
Number of input combinations 420 input delays m, n
k-fold cross-validation Bayesian Regularization multistart approach
2520 models
Validation GoF, NRMSE, FIT, RMSE
PROOF OF CONCEPT
Data aquisition More complex system
Nonlinear Autoregressive Exogenous (NARX) ANN Model
Number of input combinations TBD
k-fold cross-validation Bayesian Regularization multistart approach
n models
Validation GoF, NRMSE, FIT, RMSE
Hypothesis confirmation

Figure 3.1 Full research methodology

3.1. SHIPS DIGITAL TWIN DEVELOPMENT FRAMEWORK

Preliminary research has shown that DT as a concept is not precisely defined, especially in terms of specific application. This ambiguity creates challenges in both understanding and implementing the concept effectively across different contexts. Furthermore, there are no established guidelines or standardized frameworks to support the development of the DT concept. The absence of such elements highlights an evident gap in the literature and implies the need for further clarification to ensure consistent and meaningful use of the concept. Therefore, the first objective of this research was to demystify and precisely define DT as a concept, especially in terms of specific application, and to provide guidelines for its development, setting a standard, which, as the preliminary research has shown, is lacking.

A variety of definitions exist for the digital twin (DT), but they largely converge on a common core: the combination of modelling, continuous data acquisition, data processing, and simulation, as surveyed in [26].

The ship's digital twin (SDT) is defined as:

"A simulation of a ship's system that uses available models and sensor information as input data to mirror and predict activities or performances of its corresponding physical twin."

Building on proposed definition and the related work, the SDT can be developed through a structured procedure comprising four principal steps, shown in Figure 3.2 and proposed in [23]:

1. Define the purpose of the SDT,
2. acquire and process the necessary data,
3. model the system using appropriate methods, and
4. validate the model.

After the SDT's purpose is established, the next three steps should be performed iteratively, i.e., repeating as data availability improves and as the connectivity between the ship and its twin evolves to ensure continuity.

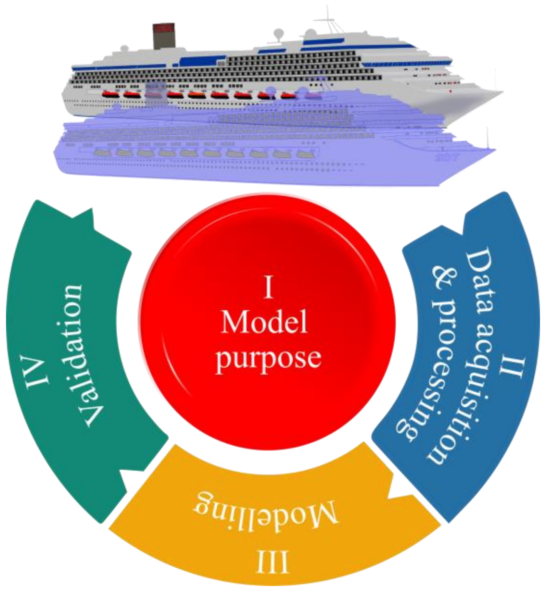


Figure 3.2 Proposed SDT concept and formulation steps

As outlined in [1], SDT applications span multiple domains. Data gathered for a given domain are subsequently processed and used for modelling, simulation, and behavioural analysis to support control, prediction, and decision-making. Consequently, the first step is to articulate a clear purpose for the SDT, because this decision constrains the scope of signals to be measured, the level of model fidelity required, and the expected operational outputs.

Depending on the intended use, an SDT relies on onboard sensors to capture variables such as heading, speed, GPS position, azimuth angles, propulsion power, engine speed and load, vibration, structural strain, and others. Raw sensor logs are rarely directly usable, instead, data processing is typically required to derive physically meaningful features. The acquisition method together with the SDT purpose will dictate suitable processing methods, for example, stress estimated from optical sensors requires different treatment than stress from strain gauges. Furthermore, sampling rate and time stamps should also be considered, as different variables tend to be acquired using different sampling rates or in different moments of time. If used for the same purpose, this should be adjusted.

Processed data are then used to construct the SDT model. While conventional physics-based mathematical modelling remains common, DTs can also benefit from data-driven approaches such as time-series models, Artificial Neural Networks (ANNs), fuzzy logic, or genetic algorithms.

As additional data become available, re-calibration can be performed, after which the model must be re-validated before operational use, hence the iterative nature of the SDT development and maintenance cycle.

The development framework introduced in this Section requires accurate representation of system dynamics, which depends on the quality of experimental data. For this reason, the next section describes the laboratory testbed and the procedures used to gather input-output measurements. The design of experiments, excitation signals, and acquisition hardware are presented to ensure reproducibility and to provide a reliable foundation for subsequent modelling tasks.

3.2. PROOF-OF-CONCEPT EXPERIMENTAL SETUP AND DATA ACQUISITION

This section describes the laboratory setup used to develop and validate the models and controllers in the proof-of-concept phase of the research. The setup provides a controlled environment in which repeatable excitation signals can be applied, measurements can be archived, and data quality can be managed systematically before modelling and controller tuning.

3.2.1. Description of the flow control system GUNT RT 674

The proof-of-concept plant is a laboratory-scale flow-control unit that mimics shipboard fluid circuits. As shown in Figure 3.3, the loop comprises a closed-circuit water line with a supply tank (1), single-speed centrifugal pump (2), inline rotameter-type flow sensor (3), proportional solenoid valve (4), flow controller (5), manual ball valves (6) and (8), and a secondary (level) tank (7).

Water is used as the working medium. The manipulated variable is the solenoid drive voltage, which directly governs the valve position and thereby the flow rate. The pump provides a nominal 4 m^3/h at up to 6 m head, typical of auxiliary pumps found on board. Flow regulation is achieved by a proportional solenoid valve driven through a PWM signal-conditioning module that accepts a standard 0-10 V analogue command and translates it into precise valve opening for continuous control. The measurable flow range is 0-800 L/h, captured by a reed-switch rotameter that outputs an analogue 0-10 V signal proportional to flow. The rotameter's mechanical pitch of 2.9-3.4 mm per contact provides sufficient resolution for real-time identification and control experiments. Unit and its corresponding equipment in a laboratory are shown in Figure 3.4.

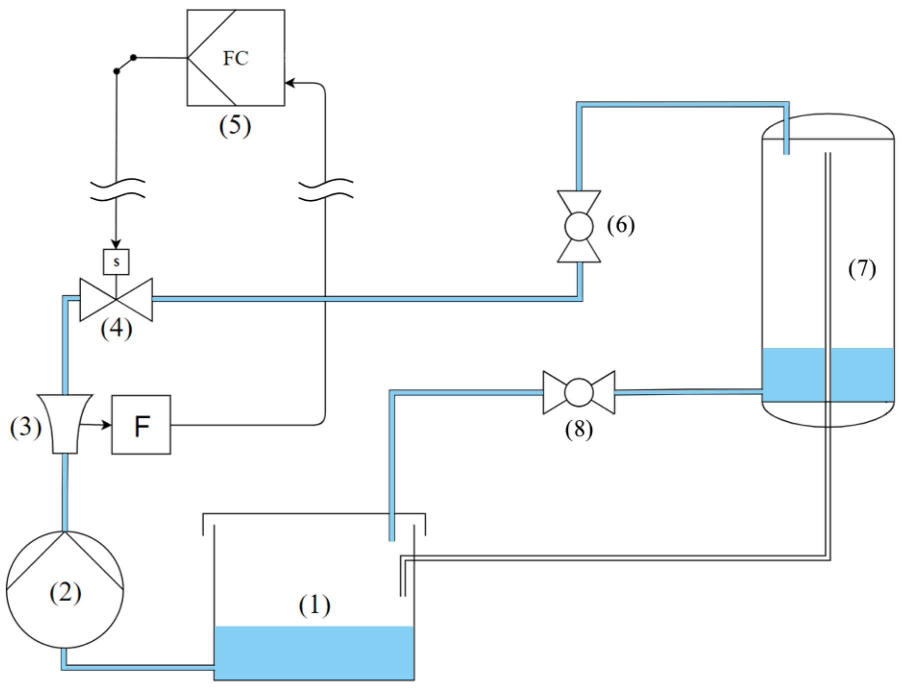


Figure 3.3 Proof-of-concept flow control system diagram

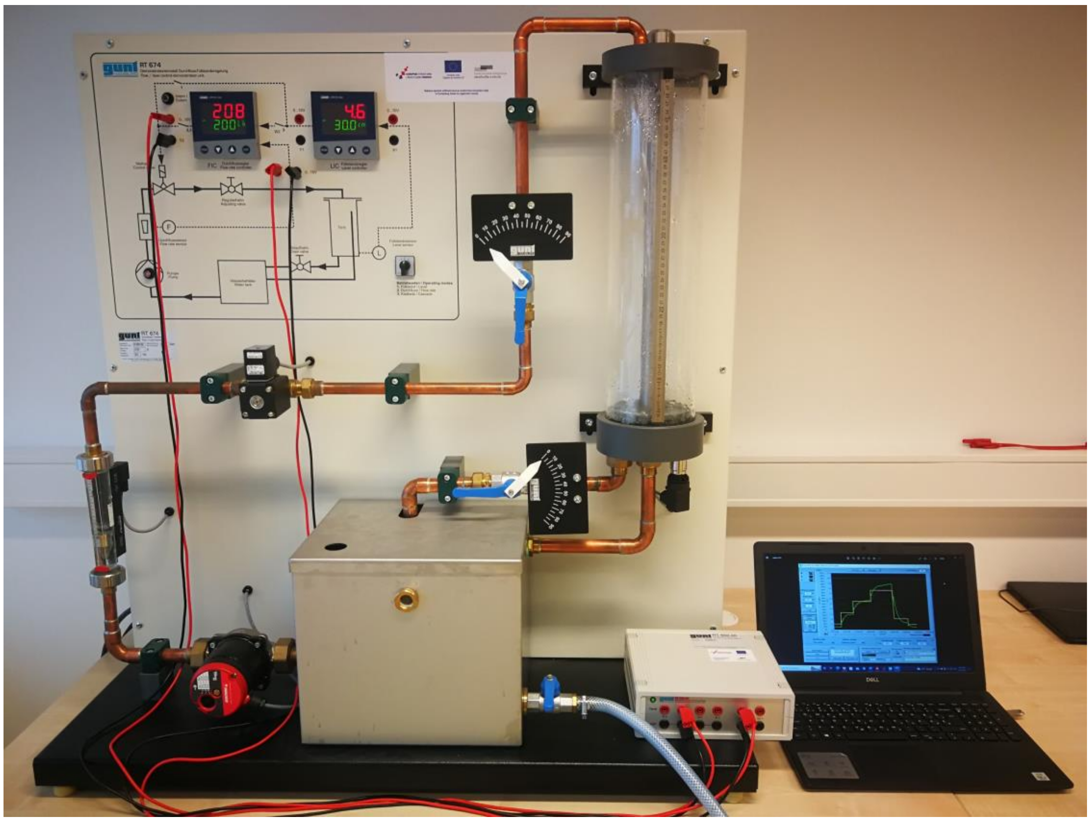


Figure 3.4 Proof-of-concept flow control system in a laboratory

3.2.2. Data acquisition in the proof-of-concept phase

Process variables and control signals were recorded through the data-acquisition interface at a fixed sampling rate of 2 samples per second according to the flowchart presented in Figure 3.5. The resulting time series were segmented into three independent datasets: the first comprising 1514 samples, the second 598 samples, and the third 266 samples. Datasets are non-overlapping and were prepared for subsequent identification, validation, and testing. All channels were time-stamped at acquisition to maintain synchronisation, and raw logs were archived before any pre-processing to ensure full reproducibility of the modelling workflow.

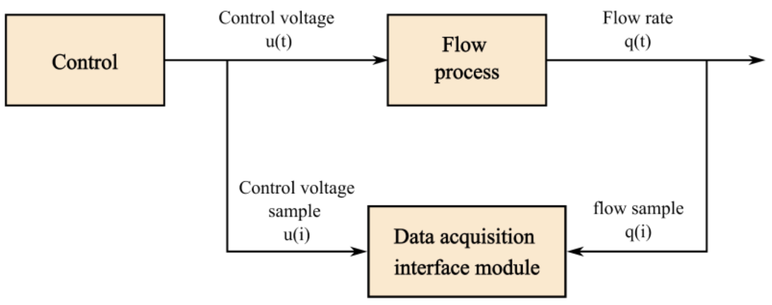


Figure 3.5 Data sampling flowchart using data acquisition equipment

3.3. TRANSFER FUNCTION MODEL DEVELOPMENT

The primary objective of this chapter is to design and evaluate a transfer function-based model of the real system. System identification was carried out in order to obtain reliable parameter estimates. Model candidates were generated by systematically varying the number of poles *i* and zeros *j* across all relevant combinations. This approach enabled a thorough evaluation of the transfer function configurations and guided the selection of the most accurate and robust representation of the system dynamics.

3.3.1. Transfer function-based model

To identify a suitable process model for the flow loop, a family of 56 candidate transfer-function structures was explored. Each candidate follows the rational form according to Equation (3.1).

$$G(s) = \frac{P_i(s)}{Q_j(s)} \tag{3.1}$$

where the numerator $P_i(s)$ and denominator $Q_j(s)$ are polynomials whose orders *i* and *j* were systematically varied to determine the best-performing structure.

Model parameters were identified using the instrumental variable (IV) approach as described in [187], with iterative Gauss-Newton refinement of the numerator and denominator coefficients. The iteration terminated when the tolerance on the least-squares update fell below 0.01.

The solenoid-valve control voltage sequence used for identification is shown in Figure 3.6. The input values span 0-10 V, chosen with regard to the valve's static characteristics shown in Figure 3.7 amplitudes were generally kept above 0.5 V, and the sequence was arranged to avoid the dead-zone observed between 8 and 8.5 V. Other excitation types were not used due to the limitations of manual actuation.

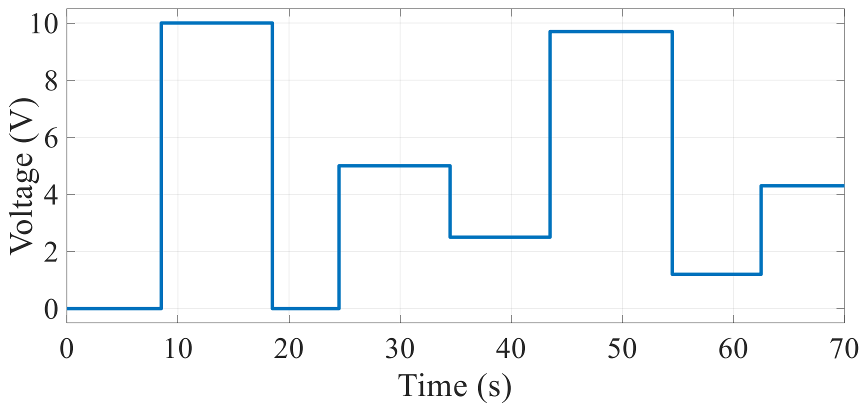


Figure 3.6 Solenoid valve control voltage sequence used for identification

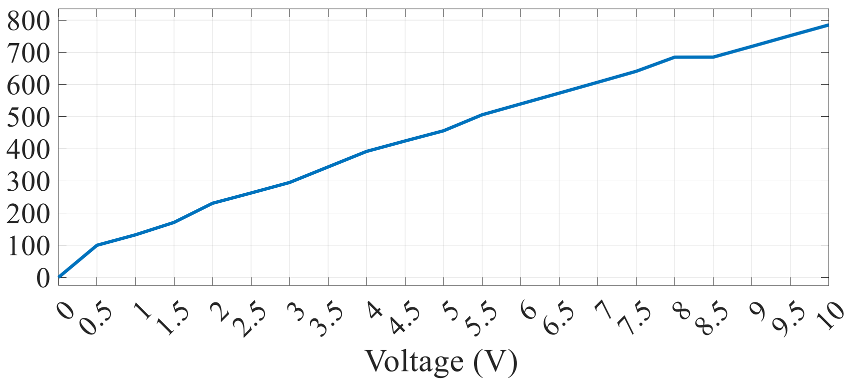


Figure 3.7 Solenoid valve static characteristic

3.3.2. TF model evaluation and validation

Model quality during identification and on an independent dataset was assessed using a goodness-of-fit measure calculated according to Equation (3.2) from the normalised root mean square error (NRMSE), calculated according to Equation (3.3). This evaluation procedure is consistent with MATLAB's System Identification Toolbox practice as explained in [188].

$$Goodness\ of\ fit\ (\%) = (1 - NRMSE) \cdot 100 \tag{3.2}$$

$$NRMSE = \sqrt{\frac{\sum_{k=1}^{n}(q_i - \hat{q}_i)^2}{\sum_{i=1}^{n}(q_i - \bar{q})^2}} \tag{3.3}$$

Where q_i represents the actual value of flow, $\hat{q}$ the predicted value of flow, $\bar{q}$ the mean of actual flow.

This choice enables direct comparability with MATLAB identification reports and provides an intuitive percentage scale for both identification and validation sets.

A representative comparison between measured flow and the simulated response of the TF1 model, with 13 poles (*i*=13) and 9 zeros (*j*=9) for this input, is shown in Figure 3.8.

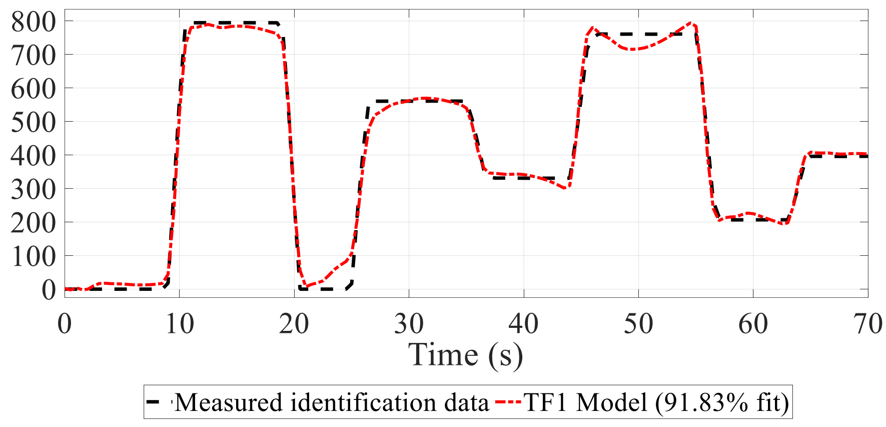


Figure 3.8 Comparison of measured and modelled (TF1) response during identification process

The influence of poles and zeros on fit is summarised in Figure 3.9. As expected, increasing the number of poles and zeros improves accuracy only up to a point. Further increases yield diminishing returns and may degrade validation due to overfitting. Note that model realisation requires that the number of poles must be equal or greater than number of zeros, i.e., $i \geq j$. Non-feasible combinations are indicated by the blank (non-coloured) area.

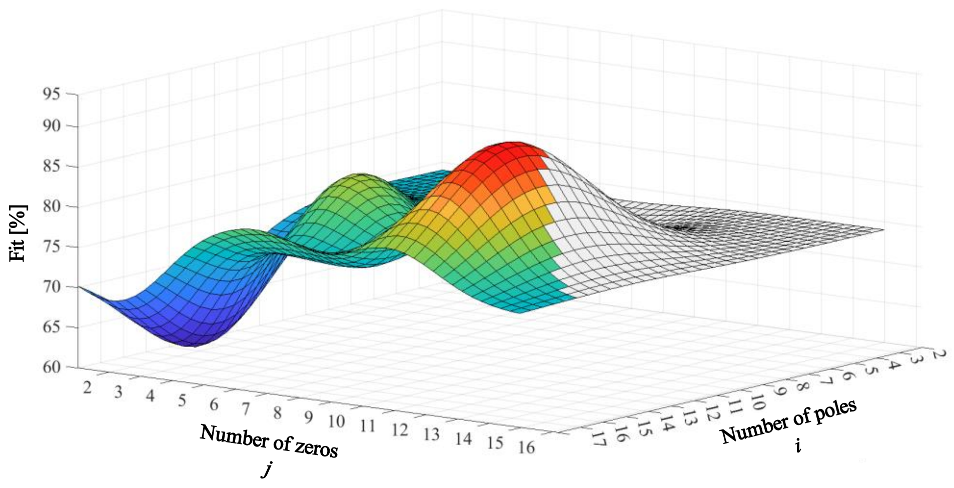


Figure 3.9 The influence of number of poles and zeros to goodness-of-fit measure

All candidate transfer functions were then validated on a separate dataset of 266 input-output samples. The validation input, i.e., solenoid control voltage sequence, is shown in Figure 3.10. Using the same goodness-of-fit (%) metric as in identification, TF1 again yielded the best validation performance, with a 91.2% fit. Based on these results, TF1 was selected as the baseline transfer-function model for subsequent comparative analysis and controller-tuning studies.

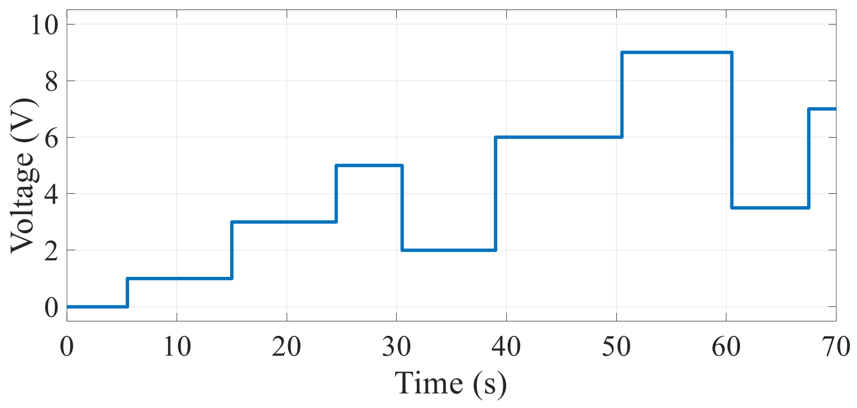


Figure 3.10 Solenoid valve control voltage sequence used for validation

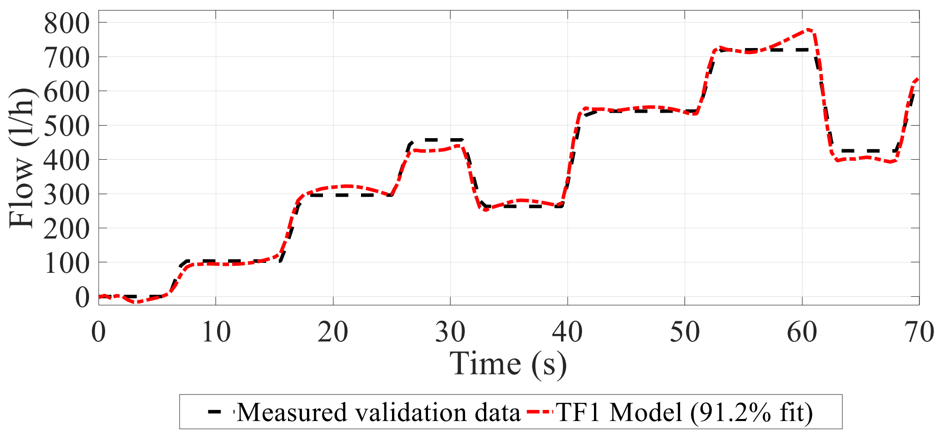


Figure 3.11 Comparison of measured and modelled (TF1) response during validation process

The top four models by goodness-of-fit (%) on the identification data and the worst model are listed in Table 3.1. The TF1 model with 13 poles (i=13) and 9 zeros (j=9) achieved the highest fit with goodness-of-fit value of 91.83% and 91.20% on identification and validation datasets, respectively.

Table 3.1 Goodness-of-fit measure on the identification and validation datasets

	GoF [%]	
	Identification	Validation
TF 1 Model	91.83	91.20
TF 2 Model	89.53	86.98
TF 3 Model	88.89	87.62
TF 4 Model	85.63	84.04
TF 5 Model	61.29	51.84

3.4. ARTIFICIAL NEURAL NETWORK MODEL DEVELOPMENT

The primary objective of this chapter is to design and evaluate an ANN-based NARX model of a real system. A series of experiments were performed to identify the optimal number of input delays (n and m), effectively determining the suitable number of inputs for the ANN NARX structure. In total, 2520 models were developed and tested. This extensive set of experiments was intended to enhance the generalization capability and robustness of the resulting models.

3.4.1. ANN-based NARX model

The nonlinear Autoregressive model with exogenous input (NARX) is a standard choice for approximating nonlinear dynamic systems as stated in [189]. In discrete time, the next output is expressed as a nonlinear function of past outputs and past inputs, with delay orders m and n, respectively, according to the Equation (3.4).

$$y(t+1) = f\big(y(t), \ldots, y(t-m+1);\ u(t), u(t-1), \ldots, u(t-n+1)\big) \tag{3.4}$$

Here, *u(t)* and *y(t)* denote the input and output at time step *t*, respectively. The number of input and output samples, i.e., delays are presented with *n* and *m*, respectively, where $n \geq 0$ and $m \geq 1$. The nonlinear function f is realised with a feedforward ANN, i.e., multilayer perceptron, using a single hidden layer with sigmoid activation, which provides good approximation capability of continuous function as stated in [186].

Same datasets were used as for transfer function modelling in [20] in order to ensure comparability. The NARX network is implemented in MATLAB as a feedforward ANN. Training was performed using Bayesian Regularization (BR) algorithm with backpropagation as the optimiser, starting from 100 hidden neurons. The network takes delayed voltage u_{t-n} and flow q_{t-m} timeseries as inputs and returns the current flow sample q_t according to Equation (3.5).

$$q_t = ANN(\, u_{t-n}, q_{t-m}) \tag{3.5}$$

To determine suitable delay orders (m,n), multiple training iterations were performed according to the flowchart shown in Figure 3.12. To reduce sensitivity to poor initialisations, i.e., to avoid local minima of the cost function during training, each network structure was trained using a multi-start strategy with 10 random initialisations, as recommended in [190]. Each training run was limited to 10000 epochs. From the 1514 sample dataset, data were split into training, validation, and test subsets using 70%, 15% and 15% ratios.

Since there are 42 possible combinations of delays (*m*,*n*), one training iteration produces 420 ANN-based NARX models, each stored in *net{}* cell in Matlab workspace, with the associated (*m*,*n*) delays recorded in *delays*(*net{}*) cell. The corresponding mean-square error (MSE) for each model is calculated according to Equation (3.6) during training and stored in *MSE*(*net{}*) cell.

Training was conducted on a Windows 10 workstation with Intel i7-9700K, 32 GB DDR4-3200 RAM, and 500 GB NVMe M.2 SSD. Building the complete set of 2520 models required 38 h 56 min of compute time.

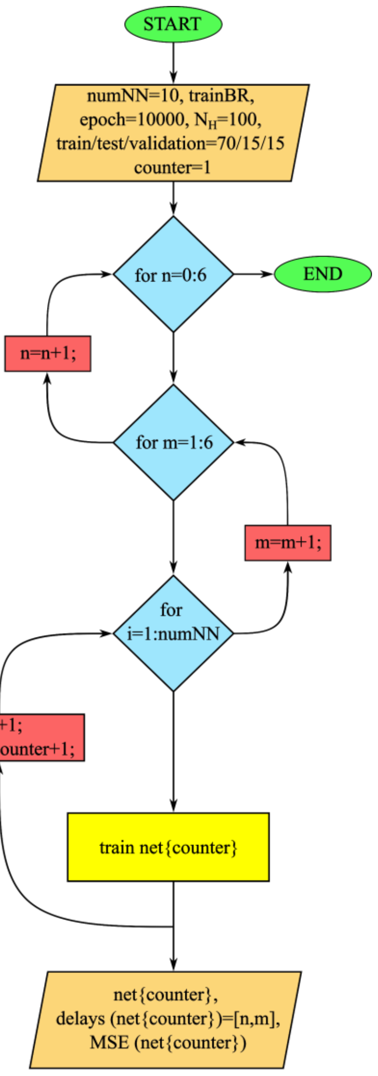


Figure 3.12 Flowchart of the NARX model training process proposed in research

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(q_i - \hat{q}_i)^2 \tag{3.6}$$

Where q_i represents the actual value of flow and $\hat{q}$ the predicted value of flow across n number of samples.

As noted in [191], when datasets are relatively small, k-fold cross-validation provides a robust basis for model assessment. In this approach, the data are partitioned into *k* folds and the ANN-based NARX is trained repeatedly so that each fold serves once for training process. This yields *k* independent estimates for each evaluation metric e.g., MSE, GoF, and best-fit, which are then averaged to produce a single, more reliable performance indicator than would be obtained from a single fold. Since there is no strict rule for choosing *k*, values in the 5-10 range are commonly adopted [192]. In this research, the timeseries dataset implies six distinct permutations of the prescribed splits, so a permuted 6-fold procedure was carried out to balance bias and variance.

To minimise bias in both testing and validation, networks were trained over all six permutations of the training, validation and test assignments while preserving the 70/15/15 ratio across the full record, as summarised by the flowchart shown in Figure 3.13. Therefore, for each training run, 70% of the 1514 samples were used for training, with the remaining 30% divided equally between validation and test datasets. This produces six non-overlapping permutations without repetition, listed in Table 3.2. Because each training iteration generates 420 ANN-based NARX models, combining these with the six dataset permutations results in a total of 2520 trained models.

For consistency with the transfer function-based models presented in Chapter 3.3.2., performance was summarised by the goodness-of-fit based on NRMSE, according to the Equation (3.3).

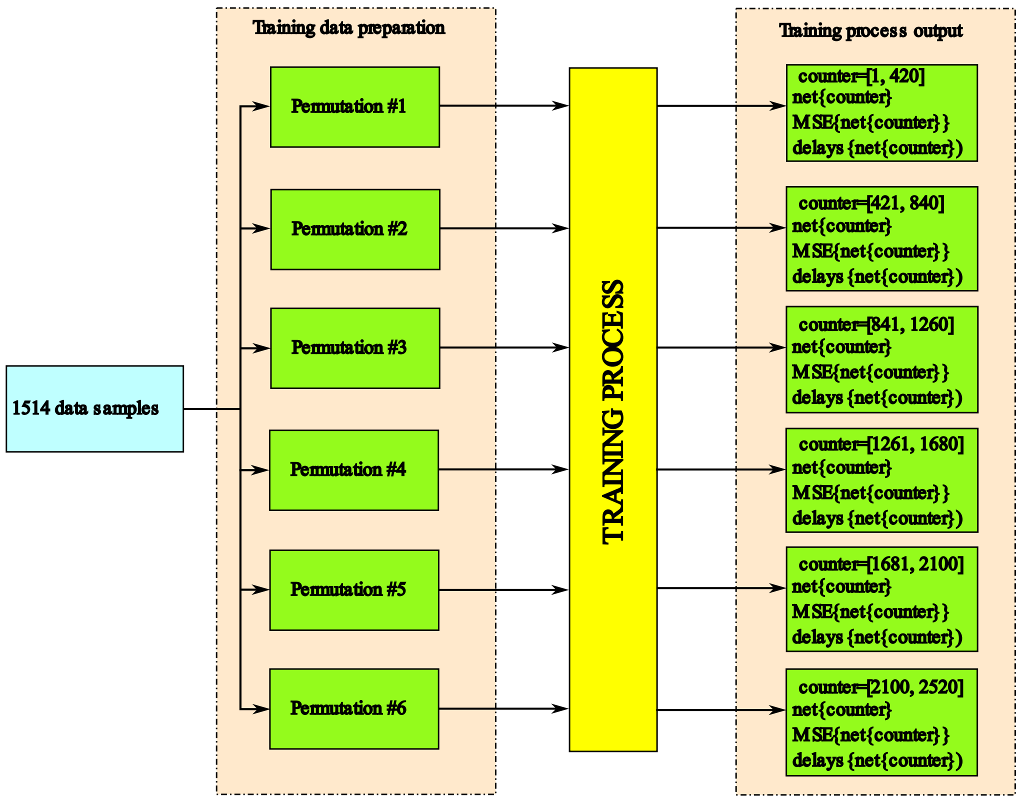

Training data preparation
Training process output
1514 data samples
Permutation #1
Permutation #2
Permutation #3
Permutation #4
Permutation #5
Permutation #6
TRAINING PROCESS
counter=[1, 420] net{counter} MSE{net{counter}} delays{net{counter}}
counter=[421, 840] net{counter} MSE{net{counter}} delays{net{counter}}
counter=[841, 1260] net{counter} MSE{net{counter}} delays{net{counter}}
counter=[1261, 1680] net{counter} MSE{net{counter}} delays{net{counter}}
counter=[1681, 2100] net{counter} MSE{net{counter}} delays{net{counter}}
counter=[2100, 2520] net{counter} MSE{net{counter}} delays{net{counter}}

Figure 3.13 Flowchart of the NARX model training process proposed in research

Table 3.2 Training, testing and validation dataset permutations within overall dataset consisting of 1514 data samples

Permutation number	**Training dataset [sample sequence]**	**Test dataset [sample sequence]**	**Validation dataset [sample sequence]**
1.	[1, 1060]	[1061, 1287]	[1288, 1514]
2.	[1, 1060]	[1288, 1514]	[1061, 1287]
3.	[228, 1287]	[1, 227]	[1288, 1514]
4.	[228, 1287]	[1288, 1514]	[1, 227]
5.	[455, 1514]	[1, 227]	[228, 454]
6.	[455, 1514]	[228, 454]	[1, 227]

3.4.2. ANN-based NARX model evaluation and validation

For model evaluation, each trained network was evaluated on the two time series datasets of 598 and 266 samples, according to the flowchart shown in Figure 3.14. These datasets were never used in training nor internal validation, thus verifying generalisation to unseen operating sequences.

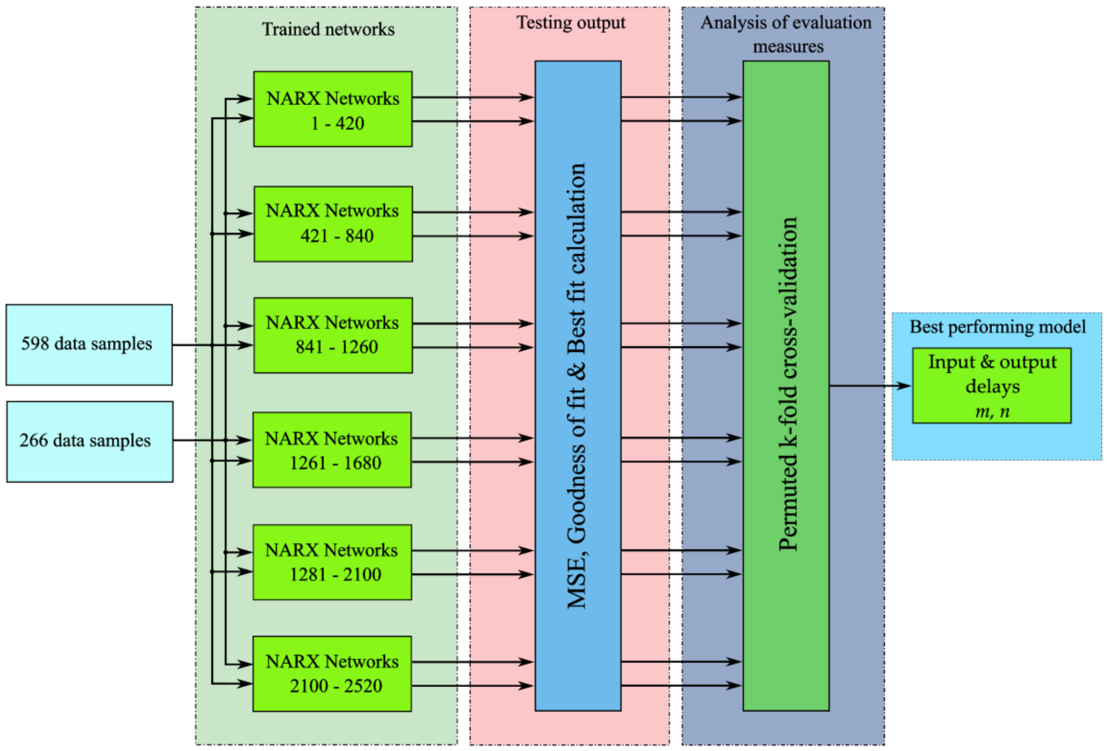

Trained networks
Testing output
Analysis of evaluation measures
598 data samples
266 data samples
NARX Networks 1 - 420
NARX Networks 421 - 840
NARX Networks 841 - 1260
NARX Networks 1261 - 1680
NARX Networks 1281 - 2100
NARX Networks 2100 - 2520
MSE, Goodness of fit & Best fit calculation
Permuted k-fold cross-validation
Best performing model
Input & output delays
m, n

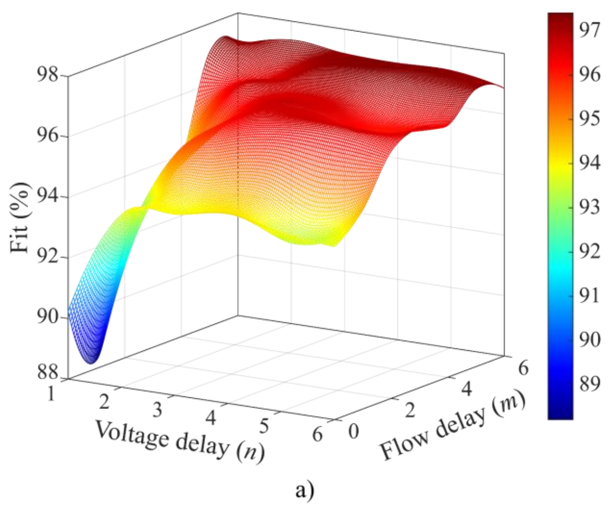
Figure 3.14 Flowchart of the NARX model evaluation process proposed in research

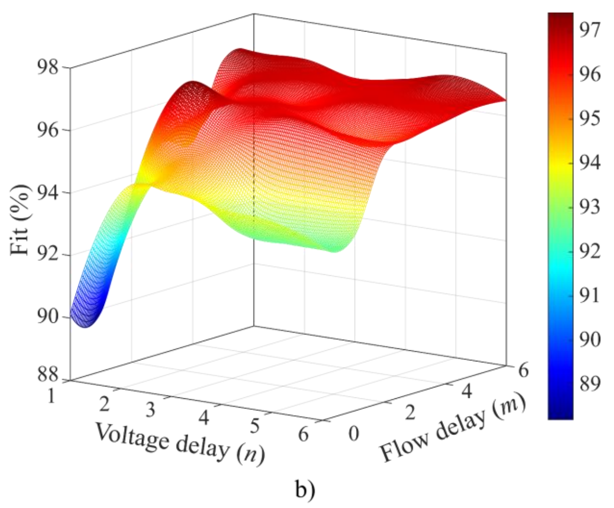
The influence of the voltage delays *n* and flow delays *m* on the ANN-based NARX model performance for the six cross-validation folds is presented in Figure 3.15. For small delay orders such as *n*=1 and *m*=0, the fit is lowest, ranging between 87% and 90%. Introducing moderate delays, such as *n*=2 and *m*=1 results in a substantial improvement, with the fit increasing to values above 94% in all folds. The highest fit values, reaching 98%, are achieved for voltage delays between *n*=3 and *n*=6 and flow delays between *m*=2 and *m*=4. Beyond these regions, further increases in delay orders yield only marginal improvements, typically less than 1%, indicating possible global maximum of a GoF. In all six folds, two turning points can be noted at *n*=3-4 and *m*=2-3, as well as at *n*=5-6 and *m*=2-4.

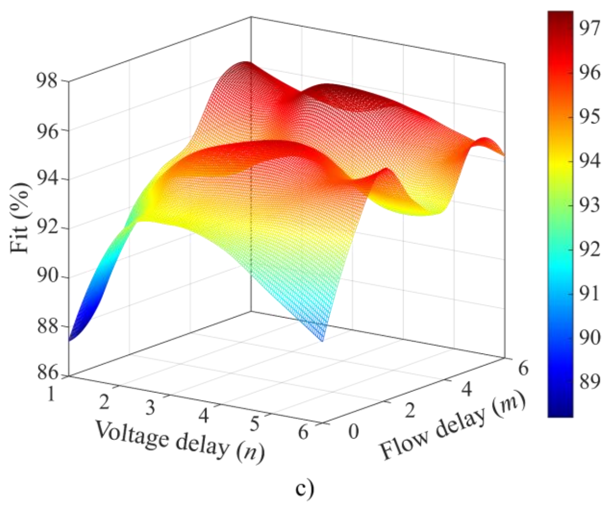
As in training process, in evaluation process a MSE was used to quantify the prediction error of the ANN-based NARX models. Additionally, a best fit evaluation measure, calculated according to Equation (3.7) was introduced.

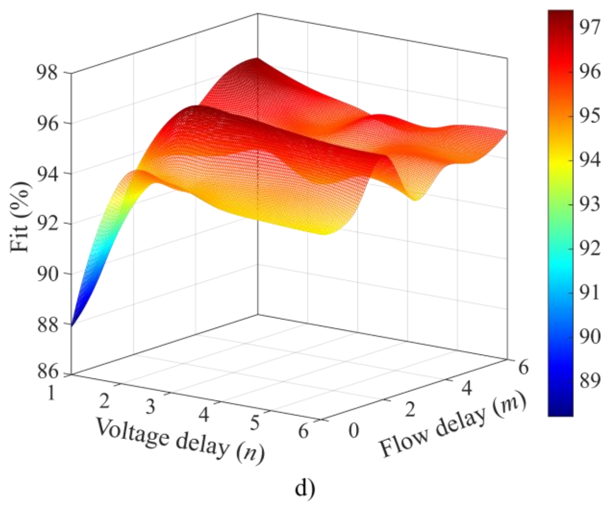
$$Best\ fit = \left(1 - \frac{\sum_{i=1}^{n}|q_i - \hat{q}_i|}{\sum_{i=1}^{n}|q_i - \bar{q}|}\right) \cdot 100\% \tag{3.7}$$

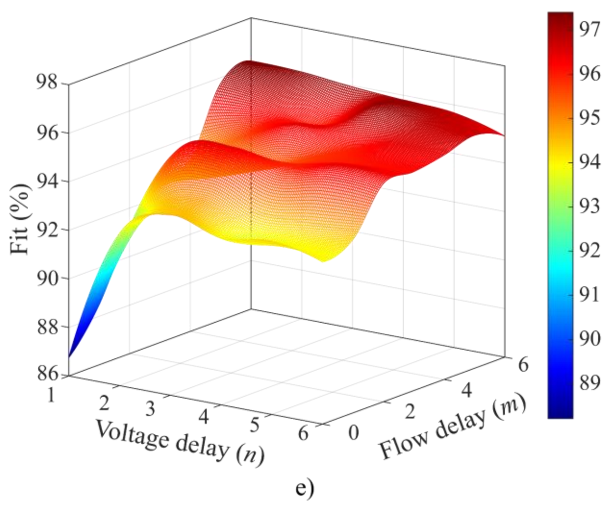
Where q_i represents the actual value of flow and $\hat{q}$ the predicted value of flow across *n* number of samples.











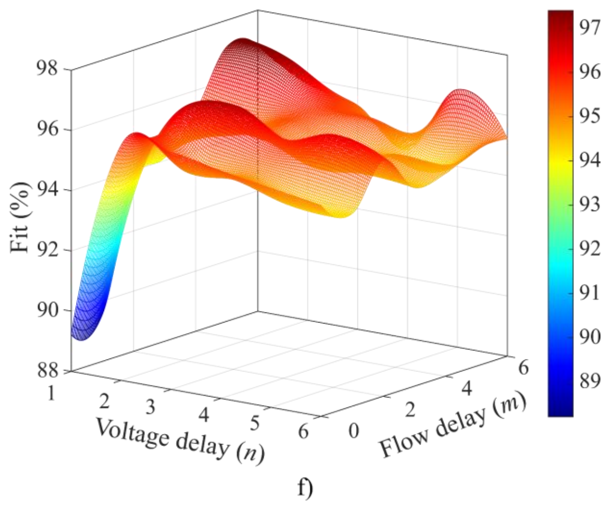


Figure 3.15 The influence of input delays to goodness-of-fit measure: (a) on 1st permutation of data; (b) on 2nd permutation of data; (c) on 3rd permutation of data; (d) on 4th permutation of data; (e) on 5th permutation of data; (f) on 6th permutation of data

As noted earlier, MSE, GoF, and best fit are the core metrics reported throughout this research. However, for ranking candidate networks to select a single model, one criterion must be decisive. Because MSE is also the training cost function, it was chosen as the primary ranking metric for performance evaluation.

The ten best-performing ANN-based NARX models on each of the two independent datasets are listed in Table 3.3 and Table 3.4, respectively. Rankings are based on MSE, while the corresponding GoF and best fit values are provided for completeness. The ANN NARX number column identifies the training iteration and the data-split permutation at which each best performing model was obtained. As seen in both tables, architectures with six flow delays and two or five voltage delays consistently appear among the top three solutions for both datasets. To determine the single best overall model, the average MSE across the two datasets was computed for these models and the resulting comparison is summarised in Table 3.5.

Table 3.3 Evaluation of ANN-based NARX models on test dataset 1

Flow delay (m)	**Voltage delay (n)**	**ANN NARX number (permutation)**	**MSE**	**Best fit**	**GoF**
6	1	366 (1st)	77.214	98.195	96.744
6	5	402 (2nd)	77.659	98.128	96.724
6	2	380 (2nd)	82.881	98.208	96.573
6	5	405 (2nd)	87.583	98.018	96.551
5	3	322 (2nd)	88.515	98.092	96.534
5	3	329 (2nd)	91.625	98.658	96.376
6	1	367 (1st)	92.666	98.569	96.355
5	2	313 (1st)	95.070	98.576	96.304
6	1	365 (2nd)	95.491	98.233	96.300
6	1	361 (2nd)	96.286	98.744	96.296

Table 3.4 Evaluation of ANN-based NARX models on test dataset 2

Flow delay (m)	Voltage delay (n)	ANN NARX number (permutation)	MSE	Best fit	GoF
6	2	371 (1st)	53.460	98.059	96.885
6	5	403 (2nd)	62.642	97.900	96.506
5	4	339 (2nd)	63.038	97.584	96.519
5	4	336 (2nd)	63.522	97.756	96.631
5	3	329 (1st)	66.465	97.744	96.502
6	5	408 (2nd)	66.664	98.038	96.575
5	2	315 (1st)	67.343	97.636	96.474
6	5	405 (2nd)	69.844	97.456	96.316
4	4	278 (1st)	70.027	97.980	96.943
6	5	402 (3rd)	70.335	97.794	96.295

Table 3.5 The overall best-performing ANN-based NARX models

Flow delay (m)	Voltage delay (n)	Average MSE
6	2	68.171
6	5	70.151

The structure of the best-performing model, i.e., model with 6 flow delays and 2 voltage delays is shown in Figure 3.16. As an input signal, current and two previous samples of voltage u_t are used. As a feedback flow signal q_t, six future samples are used. Hidden layer consists of two sets of weight matrices W for processing the signals and a bias vector b. These weighted inputs are summed and passed through a sigmoid activation function. The output layer applies additional set of weights W and biases b and a linear activation function to produce the flow sample q_t.

The analysis revealed that the flow time series, i.e., the flow delay m, exerts a dominant influence on ANN-based NARX model performance, as the ten best-performing models predominantly used five or six previous flow samples, indicating the optimal range for this parameter. In contrast, the voltage time series, i.e., the voltage delay n, demonstrated a weaker influence, with optimal values ranging between one and five delays, suggesting that recent voltage values are more relevant to model output than a longer historical series.

On the first independent test dataset, the configuration with six flow and one voltage delay achieved the best results. However, it did not rank among the top ten for the second dataset, where the model with six flow and two voltage delays performed best. Averaging results across both datasets led to the selection of the ANN NARX model with six flow and two voltage delays as the most reliable and balanced configuration.

The use of different data permutations confirmed the robustness of the ANN-based NARX models, as model performance remained stable regardless of training dataset arrangements.

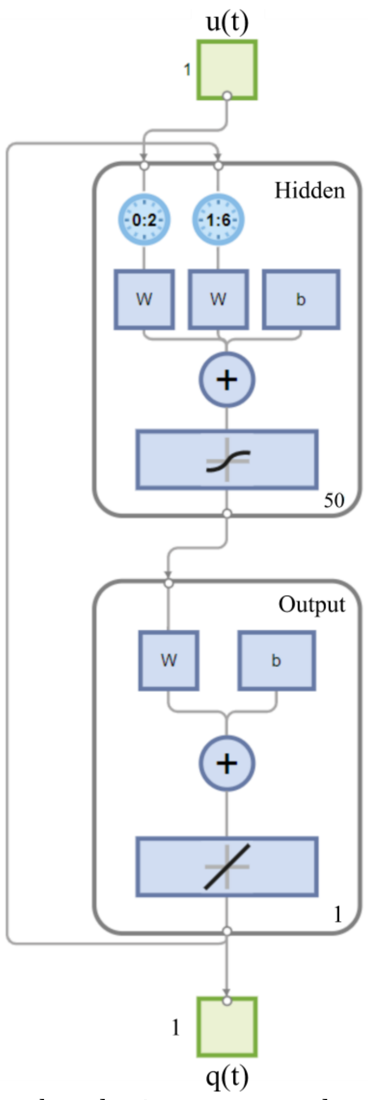


Figure 3.16 ANN-based NARX structure developed in Matlab

3.5. COMPARATIVE ANALYSIS OF TRANSFER FUNCTION-BASED AND ANN-BASED NARX MODELS

Based on the results presented in Chapter 3.3. and Chapter 3.4., the ANN-based NARX model configured with six flow delays and two voltage delays was selected as the top-performing model. This model was benchmarked against the previously developed transfer-function (TF) model using the GoF metric on the two independent datasets. The numerical comparison is summarised in Table 3.6, while the corresponding time-response graphs are shown in Figure 3.17 and Figure 3.18. A statistical test confirms the improvement with t-test yields p=0.016 (t =39.839, df=1), indicating that the presented ANN-based NARX model significantly outperforms the TF-based model under the same evaluation metrics and operating conditions.

Table 3.6 Comparison of the best-performing ANN-based NARX and TF-based models using GoF evaluation metric

Dataset	**ANN-based NARX model**	**TF-based model**
	GoF	**GoF**
Test dataset 1	97.85 %	91.83 %
Test dataset 2	97.53 %	91.20 %

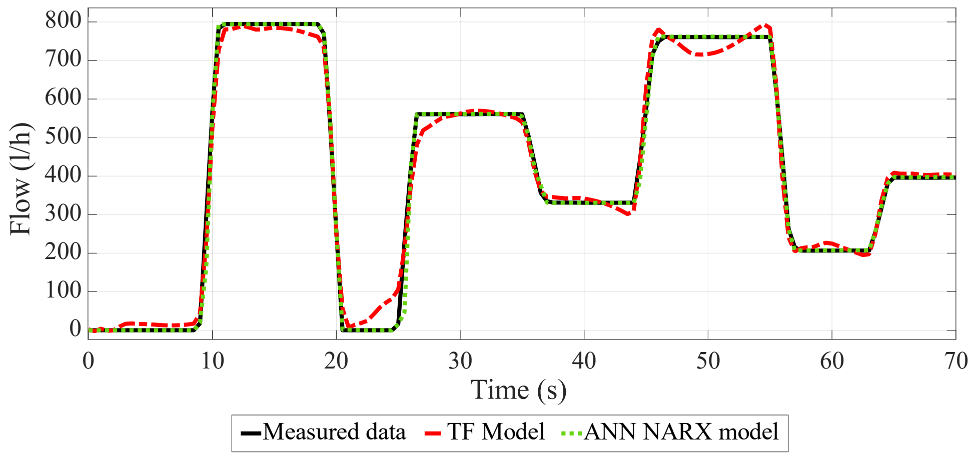


Figure 3.17 Time response of the of the best performing ANN-based NARX and TF-based models on test dataset 1

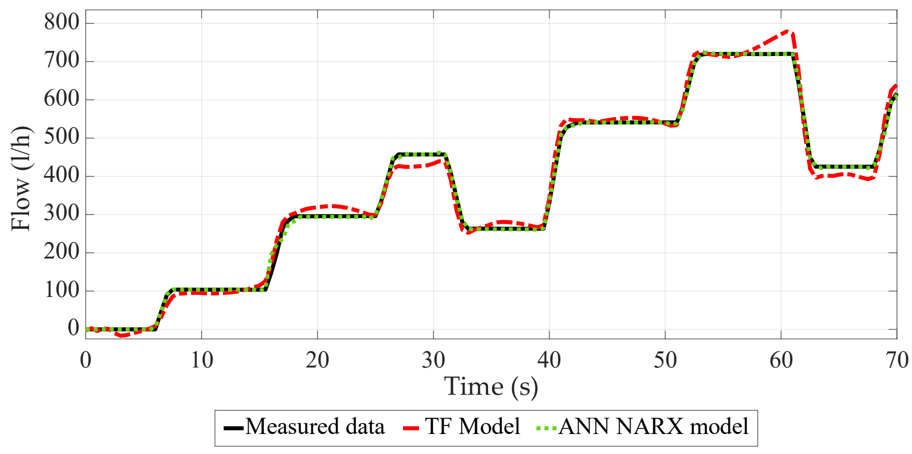


Figure 3.18 Time response of the of the best performing ANN-based NARX and TF-based models on test dataset 2

An error analysis was conducted for both test datasets to determine the conditions under which the model's performance deviates from the target outputs. As the outcomes for the two datasets are similar, only the analysis for test dataset 2 is presented. As illustrated in Figure 3.19, the ANN-based NARX model exhibits higher error levels during transitional phases, whereas its accuracy improves notably under steady-state conditions. The most notable error is between sample number thirty and forty, while the rest is negligible. However, the TF-based model exhibited higher error levels, even under steady-state conditions, as shown in Figure 3.20.

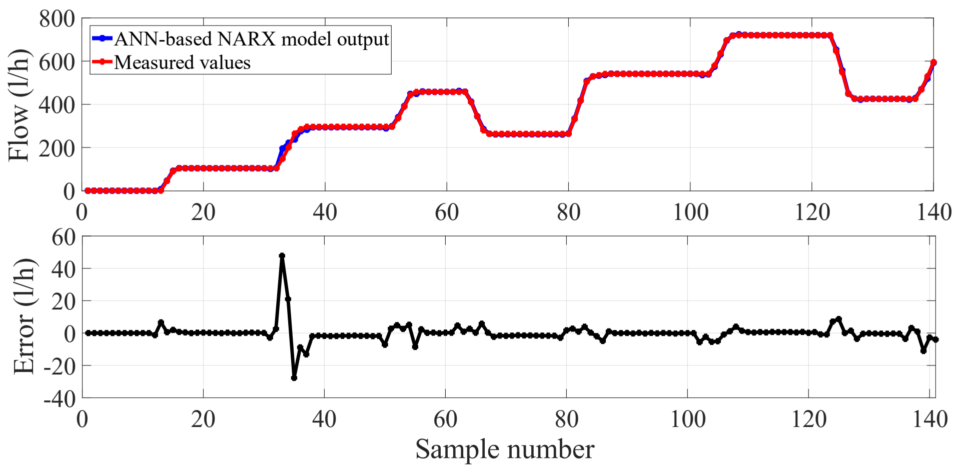


Figure 3.19 Error plot of the best performing ANN-based NARX model on test dataset 2

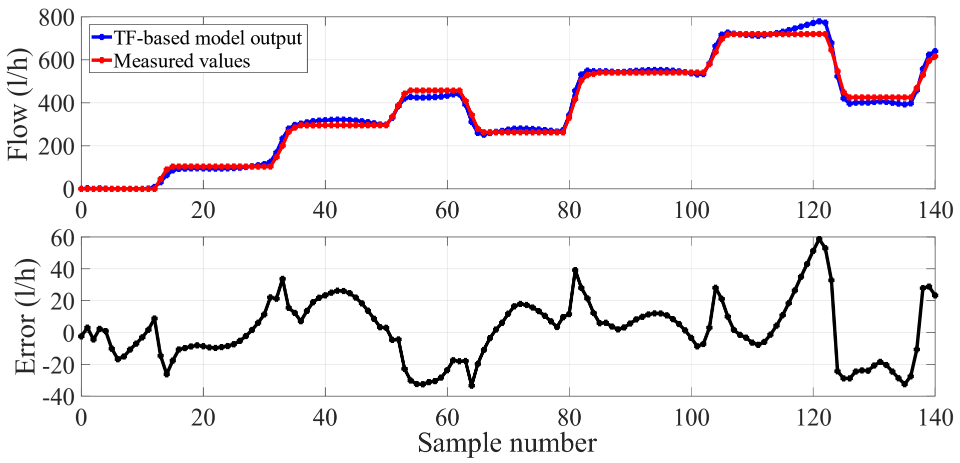


Figure 3.20 Error plot of the best performing TF-based model on test dataset 2

The primary objective of this chapter was to evaluate and compare an ANN-based NARX and TF-based models of a real system. When compared with the previously developed transfer function model, the ANN-based NARX approach demonstrated superior overall performance, with the best model configuration achieving approximately 6% higher GoF.

3.6. PID-BASED CONTROLLER TUNING

Closed-loop control systems, such as the flow-control feedback loop illustrated in Figure 3.21, forms the basis of most automation processes. In the configuration used in this research, the process variable, i.e., flow rate, is continuously measured and fed back to the controller, forming a feedback loop. In the negative-feedback structure, the controller receives the difference between the setpoint and the measured flow value, i.e., the control error. Based on this error, the controller outputs the appropriate voltage signal to the solenoid valve actuator, adjusting the process by driving the flow rate toward the desired value.

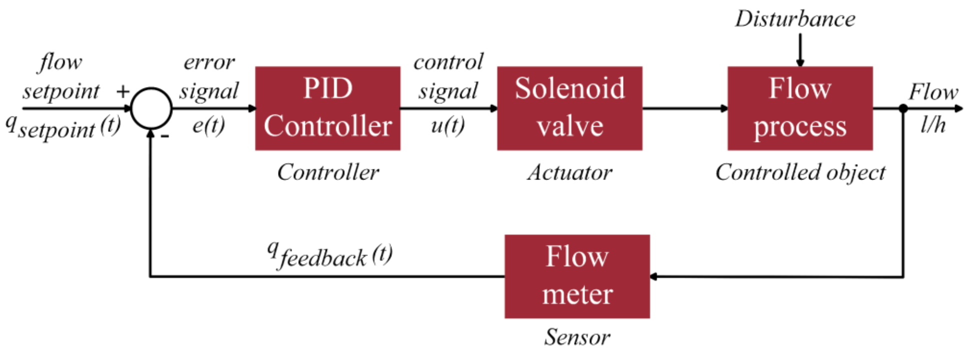


Figure 3.21 Flow process in a closed control loop with PID controller

Within such feedback systems, controller tuning is an essential part of control system design. The tuning methods and PID controller structure considered in this chapter form the basis for the comparative model-based and system-based evaluations presented in Chapter 3.7.

3.6.1. Overview of PID control structure

The PID controller used in this work is formulated in its ideal form, comprising proportional, integral, and derivative components acting on the control error. The proportional term provides immediate corrective action in response to the current error, the integral term eliminates steady-state offset by accumulating past error, and the derivative term offers anticipatory behaviour by reacting to the rate of change of the error.

The ideal PID controller structure, presented in Figure 3.22, is expressed by Equation (3.8) where $u(t)$ represents the solenoid valve voltage, K_p represents the proportional gain, T_i the integral time constant, T_d the derivative time constant, and $e(t)$ represents the control error. Determining appropriate values for these parameters is the central task addressed by the tuning methods described in the following subsection.

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right) \tag{3.8}$$

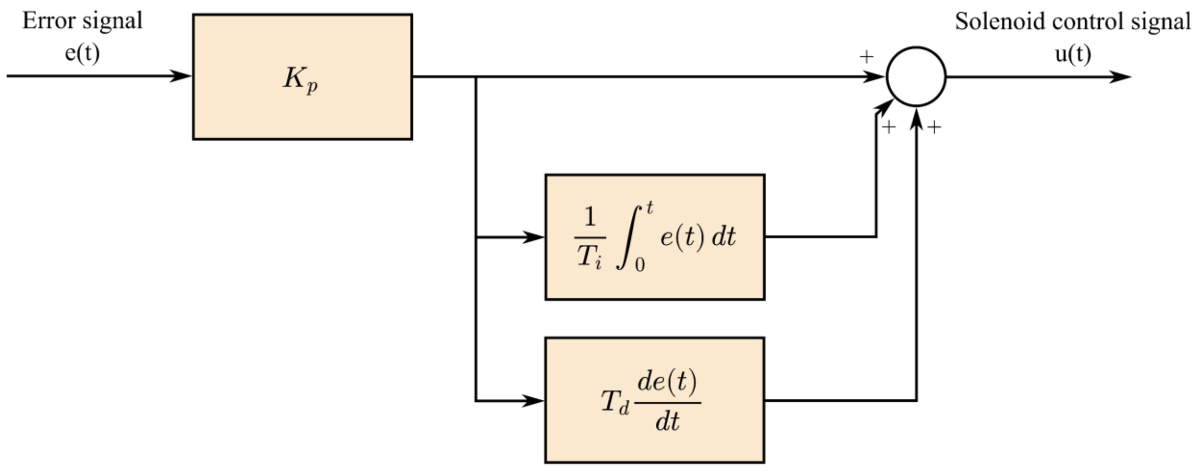


Figure 3.22 Ideal PID controller structure

3.6.2. Controller tuning methods

In this research, several classical tuning approaches were evaluated: the Ziegler-Nichols open-loop and closed-loop methods, the Chien-Hrones-Reswick method, and the Cohen-Coon method. These procedures rely on characteristic response data to compute the proportional, integral, and derivative components of the PID controller. Specifically, the Ziegler-Nichols closed-loop method requires the determination of the critical gain and associated oscillation period, whereas the remaining methods analyse the open-loop step response, i.e., process reaction curve (PRC).

General descriptions of open-loop step-response-based tuning using transfer function models can be found in [193], while closed-loop step-response-based procedures are presented in [194]. A wide range of alternative tuning techniques, beyond the scope of this research, also exists, including relay auto-tuning, pole-assignment approaches, self-tuning regulators, Newton-Raphson-based methods, and artificial-intelligence-based techniques as reviewed in [195]. Other model-based procedures are discussed in [196], whereas neural-network-based tuning methods are reviewed in [197]. Several flow-control-specific tuning approaches are also examined in [198].

3.6.2.1. Ziegler-Nichols closed-loop method

The Ziegler–Nichols closed-loop (ZN CL) tuning procedure is based on analysing the response of a system or a model to step function in closed control loop. The method requires disabling the integral and derivative actions and manually increasing the proportional gain until the system exhibits sustained oscillations. The proportional gain at which continuous and stable oscillations appear, as presented in Figure 3.23, is referred to as the critical gain K_{CR}, and the corresponding oscillation period is denoted as T_{CR}.

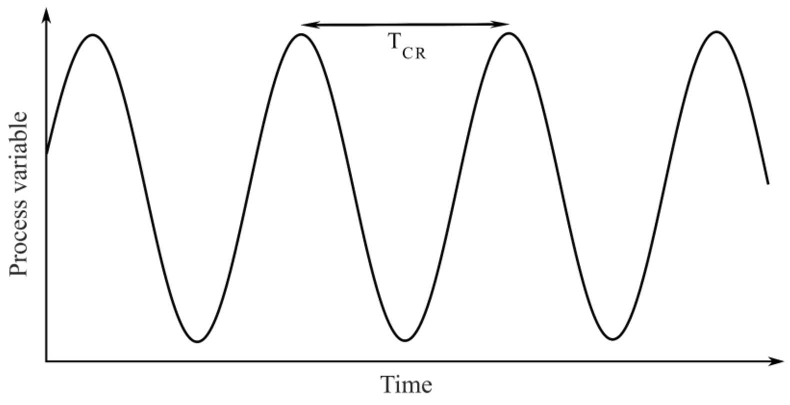


Figure 3.23 Continuous and stable oscillations at critical gain

3.6.2.2. Ziegler-Nichols open-loop method

The Ziegler-Nichols step-response (ZN SR) tuning method, proposed in [199], is based on analysing the open-loop reaction of the process to an input step change. A process reaction curve (PRC) is obtained by applying a step change Δy to the system or model input and observing the corresponding change in the process variable. The tangent method, explained in [200] and reformulated in [201] and [202], is then employed to extract the characteristic parameters of the curve by drawing a tangent line at the point of inflection of the PRC, thus enabling the extraction of process dead time T_D, time constant T_C, and the change of the process variable ΔPV. The PRC analysis using tangent method is presented in Figure 3.24.

Using the identified values of T_D, T_C, and ΔPV, the tuning constants α and τ were calculated according to Equations 3.9 and 3.10. These tuning constants are used as a basis for tuning the controllers according to the formulas presented in Table 3.7.

$$\alpha = \Delta PV \frac{T_D}{T_C} \tag{3.9}$$

$$\tau = \frac{T_D}{T_D + T_C} \tag{3.10}$$

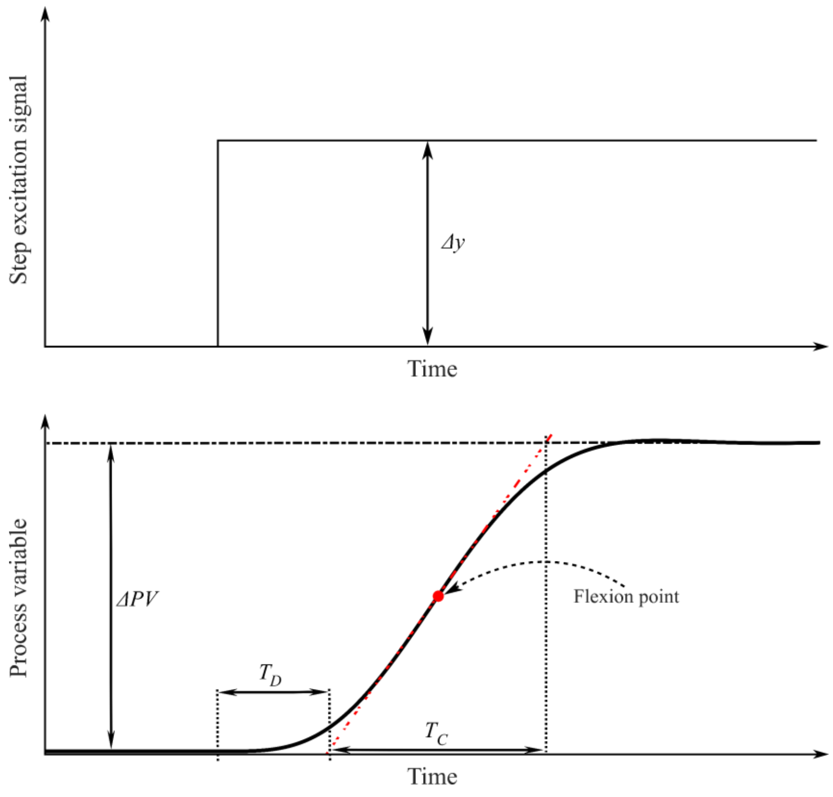


Figure 3.24 Tangent method analysis of the process reaction curve

3.6.2.3. Cohen-Coon method

As introduced in [203], the Cohen–Coon (CC) tuning method is based on PRC analysis similarly to the Ziegler–Nichols step-response method. Using the tangent method to extract PRC parameters, tuning constants are calculated according to the Equation (3.9) and (3.10). These constants are then used to compute the controller parameters, according to the tuning formulas summarised in Table 3.7.

3.6.2.4. Chien-Hrones-Reswick

Similarly, to the previously introduced open-loop tuning methods, the Chien-Hrones-Reswick (CHR) method, introduced in [204], relies on PRC analysis. This implies following the same step-response foundation as ZN SR and CC. Thus, the same measured PRC parameters and tuning constants as calculated according to Equation (3.9) and (3.10) are used to calculate the controller parameters according to Table 3.7.

Table 3.7 Controller tuning parameters calculation

Method	Controller	Kp	Ti	Td
ZN SR	PI	$0.9/\alpha$	$3T_D$	0
	PID	$1.2/\alpha$	$2T_D$	$0.5T_D$
ZN CL	PI	$0.45K_{CR}$	$0.83T_{CR}$	0
	PID	$0.59K_{CR}$	$0.5T_{CR}$	$0.125T_{CR}$
CC	PI	$(0.9/\alpha)\cdot(1+(0.92\tau)/(1-\tau))$	$T_D\cdot(3.3-3\tau)/(1+1.2\tau)$	0
	PID	$(1.35/\alpha)\cdot(1+(0.18\tau)/(1-\tau))$	$T_D\cdot(2.5-2\tau)/(1-0.39\tau)$	$T_D\cdot(0.37-0.37\tau)/(1-0.81\tau)$
CHR	PI	$0.35/\alpha$	$1.2T_C$	0
	PID	$0.6/\alpha$	T_C	$0.5T_D$

3.7. APPLICATION OF DEVELOPED MODELS FOR CONTROLLER TUNING IN FLOW CONTROL SYSTEM

This chapter presents the application of the developed transfer function-based and ANN-based NARX models for calculating controller parameters intended for use on the real flow control system. The primary aim is to evaluate how effectively each model supports the controller tuning process by calculating controller parameters that can be successfully used on the physical RT674 unit. To establish a meaningful benchmark, same tuning procedures, described in Chapter 3.6.2., were also carried out directly on the unit, enabling a comparison between controllers tuned from the models and those tuned experimentally on the unit. Once the parameters are calculated, they are evaluated on the physical system under identical testing conditions. This allows assessment of how well the nonlinear model supports controller tuning and how closely its recommendations translate to real system behaviour.

In this context, following subchapters are structured to clearly separate the contributions of each modelling approaches. Since a proportional controller alone cannot remove the steady-state error in this system, only PI and PID configurations were examined. Their performances were evaluated using the metrics introduced in the previous section, accompanied by a graphical comparison of the resulting responses.

3.7.1. Controller tuning performed on the physical RT 674 system

In order to establish a reference for evaluating model-based controllers, the tuning procedures were carried out directly on the physical RT 674 flow control unit. Unlike the model-based approaches in Sections 3.7.2 and 3.7.3, where the controller parameters were calculated using the transfer function or ANN-based NARX models and subsequently tested on the real system, the parameters in this subsection were obtained exclusively from the system's own responses.

The closed-loop control architecture remained unchanged, with the solenoid valve voltage acting as the manipulated variable and the measured flow serving as the process variable in the feedback loop. The system was operated under stable laboratory conditions without intentional external disturbances. According to the tuning methods previously described in Section 3.6, the parameters for each controller were calculated using either the system's open-loop step response or its closed-loop critical oscillation behaviour. No model-based estimates were used at any stage of this tuning process.

Once the PI and PID gains were determined, the controllers were implemented on the RT 674 unit and subjected to the same setpoint sequence used in the model-based tests. This ensured direct comparability between system-tuned and model-tuned controllers.

The RMSE values obtained using all four tuning methods tested on the physical system are presented in Table 3.8.

The system-based tuning configuration presented here serves as the most realistic benchmark for controller performance and forms the reference against which the model-based tuning approaches are compared in Section 3.7.4.

Table 3.8 Evaluation analysis of the system-based tuning of the controllers tested on the system

	RMSE	
	PI controller	**PID controller**
ZN CL	40.500	37.801
CC	46.128	48.686
ZN SR	59.163	54.824
CHR	126.154	118.843

The resulting time-domain responses for PI and PID controllers are shown in Figure 3.25 and 3.26 respectively. These results are displayed graphically because the ZN-CL tuning consistently yielded the best performance among all evaluated methods, both in terms of transient response and steady-state accuracy.

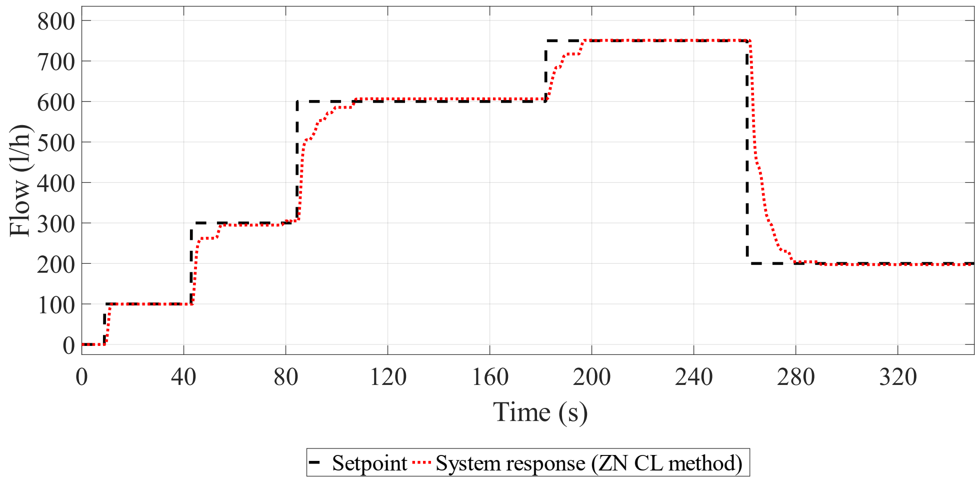


Figure 3.25 System response to setpoint sequence in closed loop with PI controller tuned directly on the system using ZN CL method

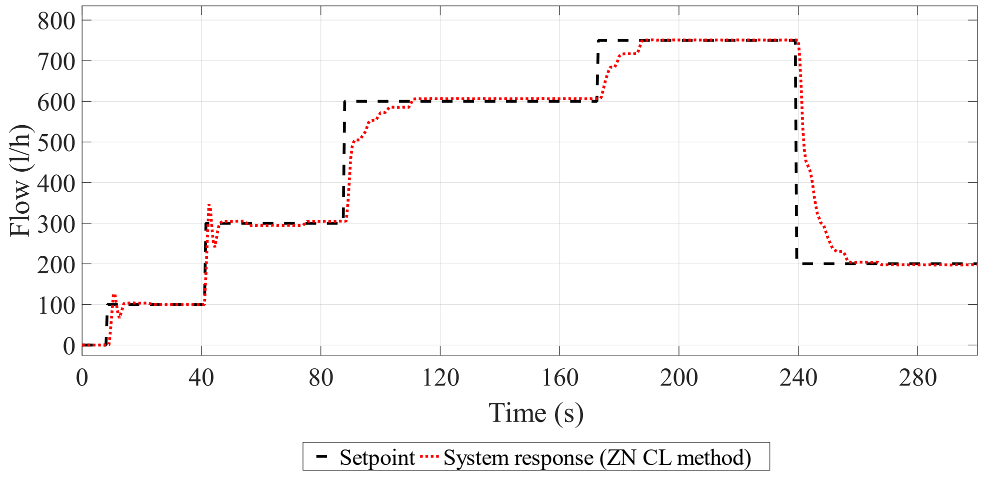


Figure 3.26 System response to setpoint sequence in closed loop with PID controller tuned directly on the system using ZN CL method

3.7.2. Controller tuning using the transfer function-based model

In the first step of the tuning procedure, the transfer function model developed in Chapter 3.3. is used as a DT for calculating controller parameters for the flow control system. By relying on the model obtained through system identification, standard model-based tuning methods can be applied in a systematic and reproducible manner. The purpose of this step is to examine how well the identified transfer function model can support the calculation of PI and PID controller parameters that, when implemented on the physical RT 674 system, provide satisfactory closed-loop performance.

To ensure a rigorous evaluation, the controller parameters derived from the transfer function model are subsequently tested both in simulation and on the real system. A closed-loop Simulink model was constructed using the identified transfer function and subjected to the same input sequences as those later applied to the physical plant. This allowed verification of the transfer function model itself and assessment of whether it provides a sufficiently accurate basis for tuning. Controller performance was analysed using the evaluation metrics introduced earlier, complemented by a graphical inspection of the time-domain responses.

Performing the tuning methods directly on the physical system enabled a direct comparison between controllers tuned experimentally on the plant and those derived from the transfer function model, providing insight into the model's suitability for practical controller design.

The Ziegler-Nichols step-response (ZN SR), Cohen-Coon, and Chien-Hrones-Reswick tuning procedure is based on analysing the open-loop reaction of the process to a step change, as explained earlier. The step-response analysis was carried out using the transfer function model of the system. Analysis of the PRC using tangent method enables the extraction of process dead time T_D, time constant T_C, and the change of the process variable ΔPV. By applying the input step change Δy at moment t=1 s, characteristic parameters of the PRC are obtained and presented in Table 3.9.

Table 3.9 Characteristic parameters of the PRC – Tangent method analysis results

Description	Symbol	Value
Input step amplitude	Δy	10 V
Change of PV	ΔPV	827.96 l/h
Dead time	T_D	0.714 s
Time constant	T_C	1.486 s
Tuning parameter α	α	397.822
Tuning parameter τ	τ	0.32455

Using MATLAB's margin() function, the gain margin, representing the gain value at which the system becomes marginally stable, was obtained and interpreted as the critical gain required by the ZN CL procedure. The corresponding model-based response yielded K_{CR} = 0.0265 with an oscillation period of T_{CR} = 2.537 s.

Controller parameters calculation flowchart is shown in Figure 3.27 and the resulting PI and PID parameters were calculated according to Table 3.10 and summarized in Table 3.11.

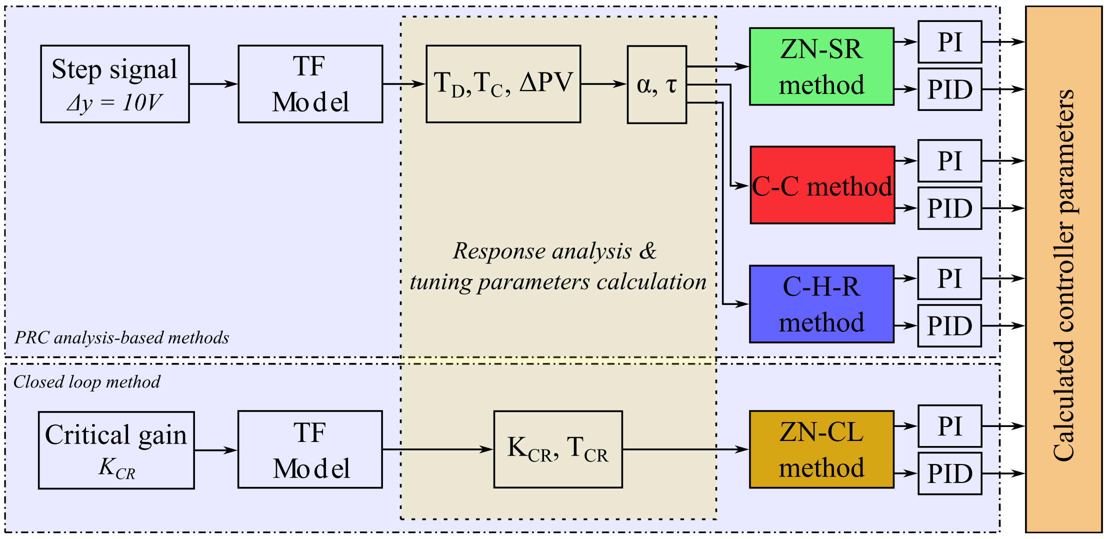


Figure 3.27 Controller parameters calculation flowchart – TF-based tuning

Table 3.10 Calculated controller tuning parameters – TF-based tuning

Method	Controller	Kp	Ti	Td
ZN SR	PI	0.002	2.142	0
	PID	0.003	1.428	0.357
ZN CL	PI	0.012	2.106	0
	PID	0.016	1.269	0.317
CC	PI	0.009	2.476	0
	PID	0.011	1.492	0.373
CHR	PI	0.003	1.195	0
	PID	0.004	1.513	0.242

The controller parameters obtained using the tuning procedures described in this section were evaluated on the physical flow control unit under setpoint regulation conditions.

The quantitative assessment of controller performance was carried out by computing the root mean square error (RMSE) using the R 4.2.2 statistical software. The ability of the controllers, tuned using the various methods described earlier, to track changes in the setpoint was evaluated experimentally on the physical system, with the corresponding results summarized in Table 3.11.

Table 3.11 Evaluation analysis of the TF-based tuning of the controllers tested on the system

	RMSE	
	PI controller	**PID controller**
ZN CL	59.03422	61.9233
CC	66.87513	68.46557
ZN SR	105.3716	76.89271
CHR	137.6533	135.4693

The closed-loop responses of the real system, obtained with the PI and PID controllers tuned using the transfer function model are presented in Figures 3.28 and 3.29, respectively. In all cases, the controllers were operated in standard setpoint regulation mode, adjusting the process output to track a sequence of predefined reference values in the absence of external disturbances.

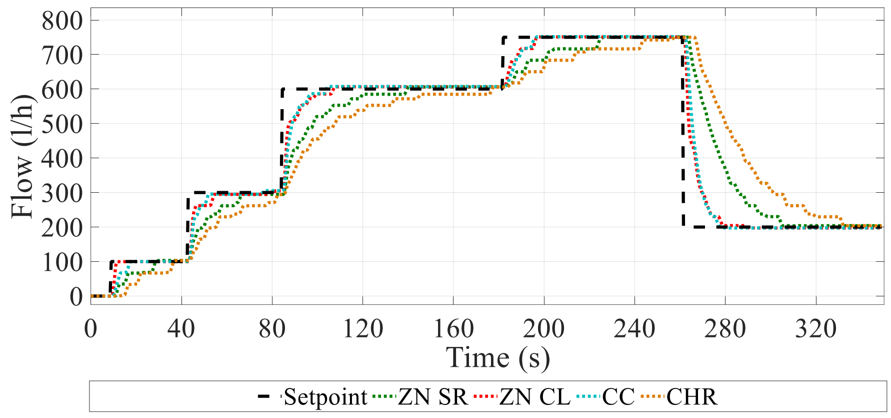


Figure 3.28 System response to setpoint sequence in closed loop with PI controllers tuned using transfer function model

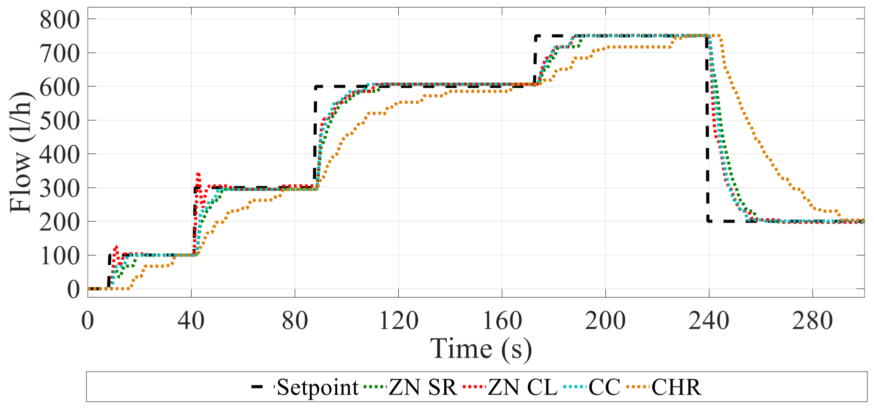


Figure 3.29 System response to setpoint sequence in closed loop with PID controllers tuned using transfer function model

3.7.3. Controller tuning using the ANN-based NARX model

Following the analysis of the transfer function-based tuning approach, this subsection examines the use of the ANN-based NARX model for deriving controller parameters for the flow control system. The ANN-NARX model, developed and validated in Chapter 3.4., captures a broader range of system nonlinearities and temporal dependencies, making it a promising candidate for improved tuning precision.

In this context, the ANN-NARX model is employed as a digital twin of the physical process to support the calculation of PI and PID controller gains. The same tuning procedures as used for the TF model are applied to the ANN-NARX model in order to ensure methodological consistency and to enable a direct comparison of their suitability for controller design. The controller parameters calculation flowchart is shown in Figure 3.30 and the resulting PI and PID parameters were calculated according to Table 3.7 and summarized in Table 3.12.

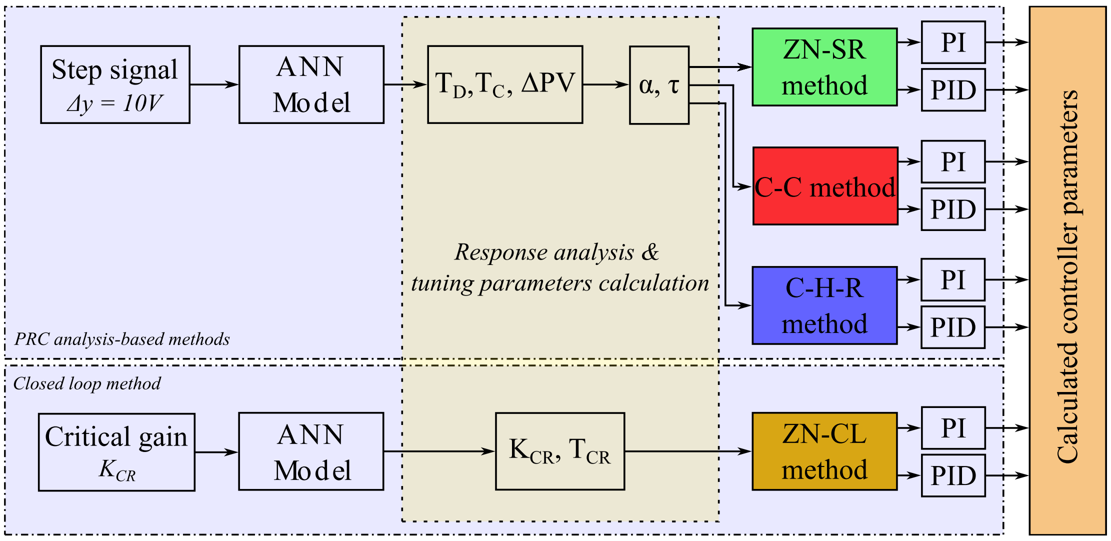


Figure 3.30 Controller parameters calculation flowchart – ANN-based tuning

Table 3.12 Calculated controller tuning parameters – ANN-based tuning

Method	Controller	Kp	Ti	Td
ZN SR	PI	0.003	1.668	0
	PID	0.003	1.112	0.278
ZN CL	PI	0.008	2.07	0
	PID	0.01	1.252	0.313
CC	PI	0.004	0.971	0
	PID	0.004	1.193	0.19
CHR	PI	0.001	1.52	0
	PID	0.002	1.267	0.278

The controller parameters derived using the tuning procedures described in this subsection were evaluated on the physical flow control unit under setpoint regulation conditions. Controller performance was quantified using the RMSE metric introduced in Chapter 2.4, and further examined through graphical inspection of the time-domain responses.

The quantitative evaluation of the controller performance was conducted by calculating the root mean square error (RMSE) using the R 4.2.2 statistical environment. The tracking capability of the controllers tuned using the ANN-based NARX model was assessed experimentally on the physical flow control system, and the resulting RMSE values are presented in Table 3.13.

Table 3.13 Evaluation analysis of the ANN NARX model-based tuning of the controllers tested on the system

	RMSE	
	PI controller	**PID controller**
ZN CL	49.59993	43.56430
CC	50.13484	65.65392
ZN SR	62.90623	68.33754
CHR	137.4912	123.8308

The closed-loop responses of the real system obtained using the PI and PID controllers tuned with the ANN-based NARX model are presented in Figures 3.31 and 3.32, respectively. As in the previous subsection, the controllers operated in standard setpoint regulation mode, adjusting the output to track a predefined sequence of setpoint values without the presence of external disturbances.

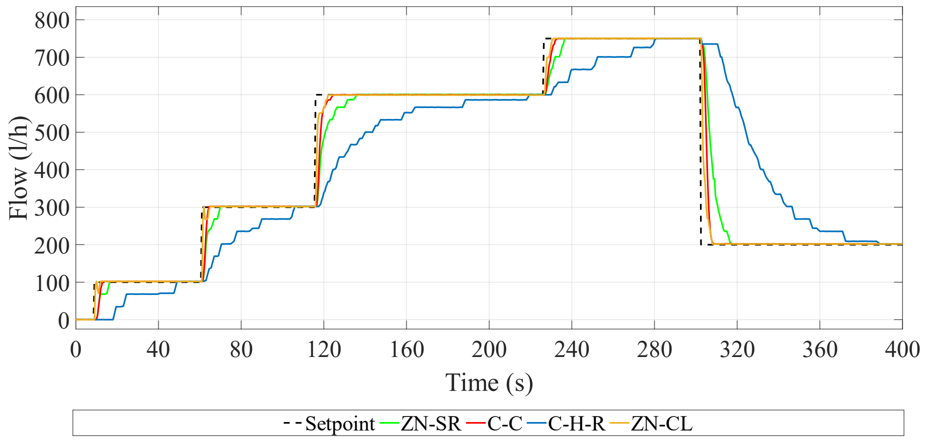


Figure 3.31 System response to setpoint sequence in closed loop with PI controllers tuned using ANN-based NARX model

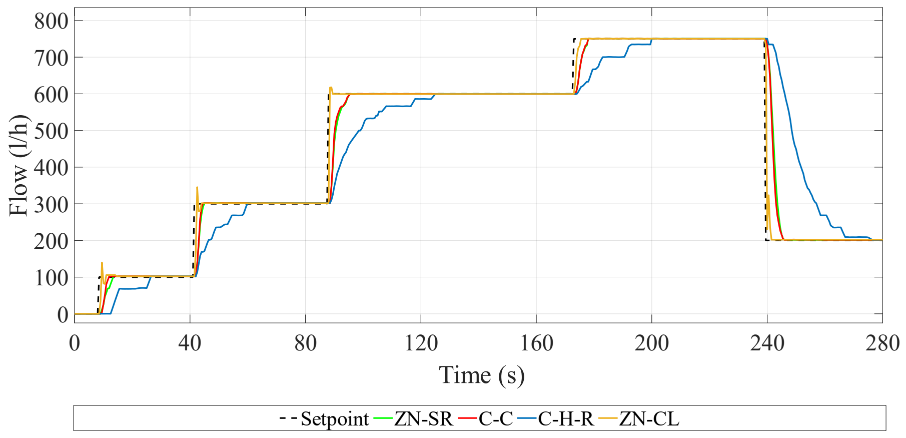


Figure 3.32 System response to setpoint sequence in closed loop with PID controllers tuned using ANN-based NARX model

3.7.4. Comparative evaluation of tuning performance

The tuning procedures described in the previous subsections produced four PI and four PID controllers. To enable a meaningful and systematic comparison, the controllers were evaluated in three cases:

I. Controllers tuned directly on the RT 674 system and tested on the system, as presented in Section 3.7.1.

II. Controllers tuned using the transfer function model and implemented on the RT 674 system as presented in Section 3.7.2.

III. Controllers tuned based on the ANN-based NARX model and implemented on the physical RT 674 system, as presented in Section 3.7.3.

A unified setpoint sequence, shown in Figure 3.10, was used across all scenarios to ensure consistency between model-based and system-based evaluations.

Quantitative performance evaluation was conducted using the root mean square error (RMSE) metric introduced in Section 2.4. RMSE values were calculated in R 4.2.2 statistical software, based on the deviation between the actual flow response and the setpoint value. These values were used to determine the most effective tuning method, evaluate the loss of accuracy when transferring controller parameters from the models to the real system, compare ANN-based tuning to system-based tuning, and compare ANN-based tuning to TF-based tuning. The results of these analyses are summarized in Table 3.14. and Table 3.15.

Table 3.14 RMSE values of the TF model-based, ANN model-based, and system-based tuned controllers

	System-based tuning		TF model-based tuning		ANN model-based tuning	
	I		II		III	
	PI	PID	PI	PID	PI	PID
ZN CL	40.50	37.80	59.03	61.92	49.6	43.56
CC	46.13	48.69	66.88	68.47	50.14	65.65
ZN SR	59.16	54.82	105.37	76.89	62.91	68.34
CHR	126.15	118.84	137.65	135.47	137.49	123.83

Table 3.15 Percentage differences of the RMSE values of the TF model-based, ANN model-based, and system-based tuned controllers tested on the system

	Δ (I-II) [%]		Δ (I-III) [%]		Δ (III-II) [%]	
	PI	PID	PI	PID	PI	PID
ZN CL	+31.39	+38.94	+18.35	+13.23	-15.98	-29.65
CC	+30.98	+28.91	+7.99	+25.84	-25.03	-4.11
ZN SR	+43.84	+28.68	+5.95	+19.78	-40.30	-11.13
CHR	+8.35	+12.26	+8.24	+4.03	-0.12	-8.59

In Table 3.14, controllers tuned using the transfer function model (case II) and the ANN-based NARX model (case III) are compared with controllers tuned directly on the physical RT 674 system (case I). Table 3.15 further summarizes the evaluation by presenting the percentage difference between system-based tuning and TF model-based tuning through Δ(I−II), as well as the difference between system-based tuning and ANN model-based tuning through Δ(I−III). Finally, Δ(III−II) reflects the overall difference between tuning based on the transfer function model and tuning performed based on the ANN model. These analyses were carried out for both PI and PID controllers. The positive values for the cases Δ(I−II) and Δ(I−III) indicate increase of the RMSE values compared to the tuning performed on the system, while the negative values for the case Δ(III−II) indicate a decrease of the RMSE values when transitioning from TF model-based to ANN model-based tuning.

The results in Table 3.14 indicate that the Ziegler–Nichols closed-loop method consistently provided the lowest RMSE values across all three tuning scenarios. When controller parameters obtained from the transfer function model (case II) were applied to the real system, PID controllers generally outperformed PI controllers. A similar pattern is observed for the ANN-based tuning (case III), although the relative performance of PI and PID controllers depends on the tuning method. For ZN CL and CHR, PID controllers resulted in lower RMSE values, whereas PI controllers performed better in the CC and ZN SR tuning methods.

The Δ(III−II) values in Table 3.15 demonstrate that controllers tuned using the ANN model generally achieve lower RMSE values than controllers tuned using the transfer function model, indicating a more accurate representation of system dynamics in the ANN-based approach. Importantly, among all tested configurations, the PID controller tuned using the ZN CL method based on the ANN model achieved RMSE values closest to those tuned directly on the system.

To further evaluate the capacity of the ANN model to support controller tuning, its performance was compared against the transfer function model. For PID controllers, ANN-based tuning outperformed TF-based tuning across all four methods, with improvements ranging from marginal to substantial. The ZN CL method yielded the most pronounced benefit, with ANN-based parameters providing considerably lower RMSE compared with those obtained from the transfer function model. For PI controllers, ANN-based tuning also improved performance for most methods, including substantial gains for ZN SR and CC. Only minor differences were observed for the CHR method. These results indicate that the ANN-based model better reflects the real system behaviour than the transfer function model, particularly in conditions characterized by nonlinearities and temporal dependencies.

The response plots illustrate these findings graphically for the ZN CL tuning approach using PI and PID controllers, in Figure 3.33 and 3.34, respectively.

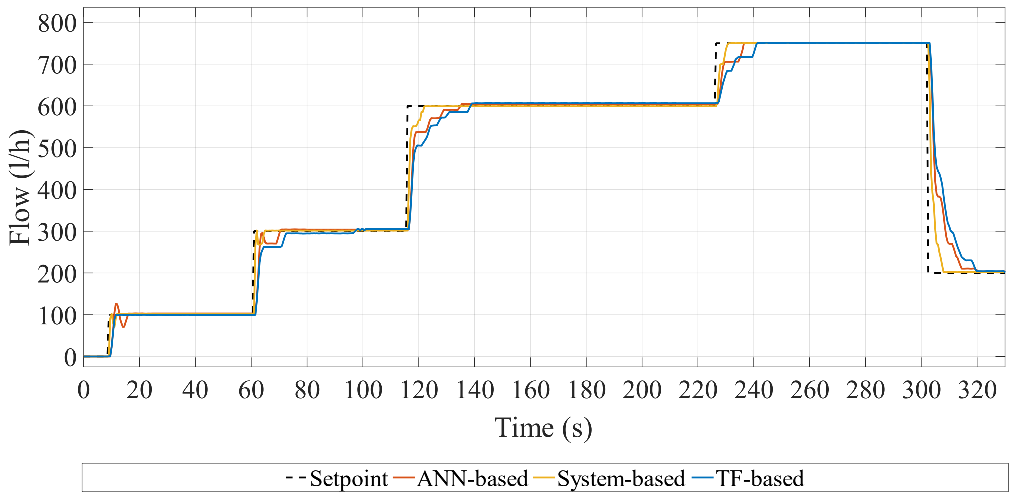


Figure 3.33 System response to the applied setpoint sequence for PI controllers tuned using the ZN CL method: TF model-based, ANN model-based, and system-based tuning

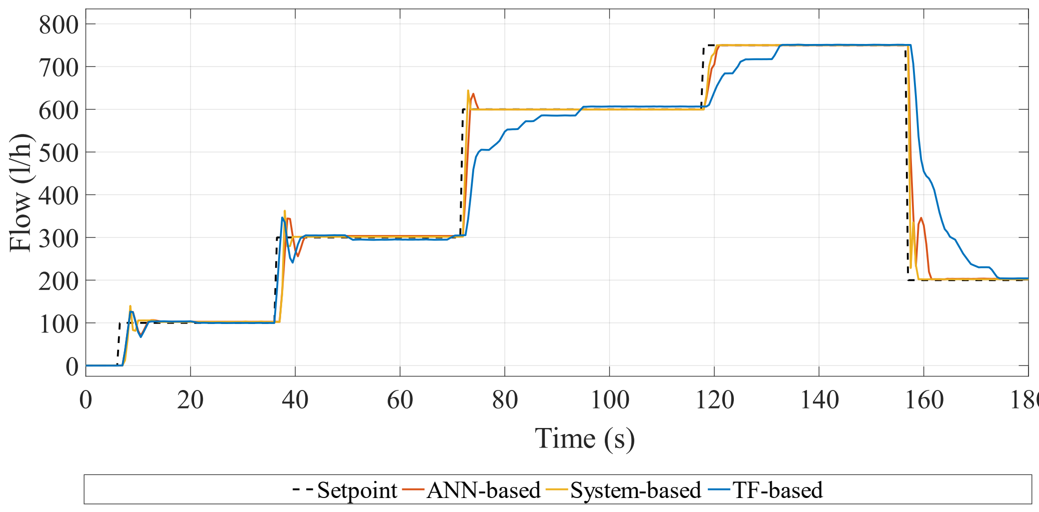


Figure 3.34 System response to the applied setpoint sequence for PID controllers tuned using the ZN CL method: TF model-based, ANN model-based, and system-based tuning

Overall, the comparative study demonstrates that the ANN-based NARX model supports controller tuning more effectively than the transfer function model, especially in terms of achieving consistent performance when parameters are transferred to the real system. Although controllers tuned directly on the physical system remain the most accurate reference, the ANN-based approach provides a meaningful balance between accuracy and practicality, enabling high-quality tuning without extensive experimentation on the real equipment.

3.8. ANN-BASED MODELLING APPROACH IN A MULTIVARIABLE CONTROL SYSTEM

In this chapter an ANN-based modelling approach presented in Section 3.4 is applied to a multivariable plant. The objective is to develop and validate an ANN-based model of the flow control system under multivariable operating conditions and to prepare the model for controller-tuning experiments in Section 3.9. The same ANN framework, training strategy, dataset partitioning practice, and evaluation metrics introduced previously are retained to ensure methodological consistency.

3.8.1. Description of multivariable control system GUNT RT 580

The GUNT RT 580 is a laboratory-scale multivariable process in which a flow and thermal subsystems are integrated on a same platform, thus enabling regulation of flow rate with a fluid temperature as a disturbance variable. From a modelling perspective, the unit constitutes a nonlinear multivariable dynamic system in which manipulated variables influence multiple process variables through shared physical components and interaction pathways. A schematic overview of the RT 580 process configuration is shown in Figure 3.35. The plant consists of two main tanks, the cold-water tank B1 and the heated tank B2 with stirring device R1. Tank B1 serves as the supply reservoir for the circulation circuits and is connected to a refrigeration loop, while tank B2 is equipped with an electric heater A1 and clear scale LI 2 and is used for experiments involving temperature variable.

In a hot-water circuit A, water is pumped from the hot-water tank B2 by a vertical multi-stage high-pressure centrifugal pump P1 and directed back toward the heated tank B2 via pipes with solenoid valves V01, V02, V03 and V04, and manual valves V05, V06, V07 and V10. The flow is regulated by the electro-pneumatically actuated control valve V08 via flow controller FIC. The liquid level in tank B2 is measured by a hydrostatic level sensor LIRC 1 and regulated through a level controller LIC.

In a cold-water circuit B, water is pumped to plate heat exchanger W1 from the cold-water tank B1 by a circulating wet rotor pump P2 through pipes including manual valves V14 and V09. The flow is measured using flow rotameter FI 2.

The thermal dynamics of the system are governed by the electric heater A1 located in tank B2 and by the plate heat exchanger W1, which thermally couples the hot-water circuit A and cold-water circuit B. Temperature sensors provide measurements at multiple locations, including the heat exchanger outlet in cold-water circuit B, measured by TIRC 1, temperature after the heat exchanger in hot-water circuit A, measured by TIR 2, temperature in hot-water tank B2, measured by TIRC 3, and temperature in cold-water tank B1, measured by TIR 4. Temperature regulation is performed using a temperature controller TIC that acts indirectly through the hydraulic variables by modifying flow conditions and directly, acting on a heater A1 duty-cycle.

The cold-water circuit B is additionally connected to a refrigeration loop via heat exchanger W2, supplied by pump P3 through manual valve V12. This loop allows active cooling of the cold-water tank B1 using compressor V1 to circulate the refrigerant through condenser and expansion valve V13. As a result, the refrigeration loop influences thermal conditions of the entire system. Variations in cooling intensity or cold-water circulation affect not only the temperature profile but also the heat transfer rate in W1, thereby impacting the temperature dynamics in tank B2. In this research the refrigeration loop is treated as an external thermal disturbance and maintained at fixed operating conditions during data acquisition experiments. The hydraulic circuit also includes auxiliary components such as the drain valve

V11 and the inline strainer V15, which are used for system drainage and protection of the piping and pumps. The GUNT RT 580 system and its corresponding equipment in a laboratory are shown in Figure 3.36.

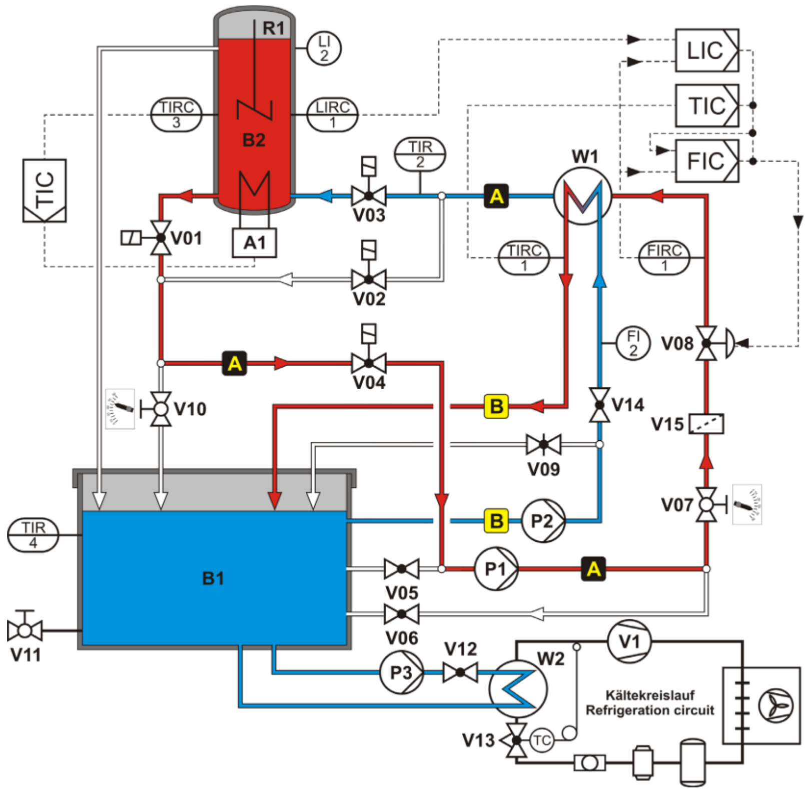


Figure 3.35 A schematic overview of the RT 580 process [205]

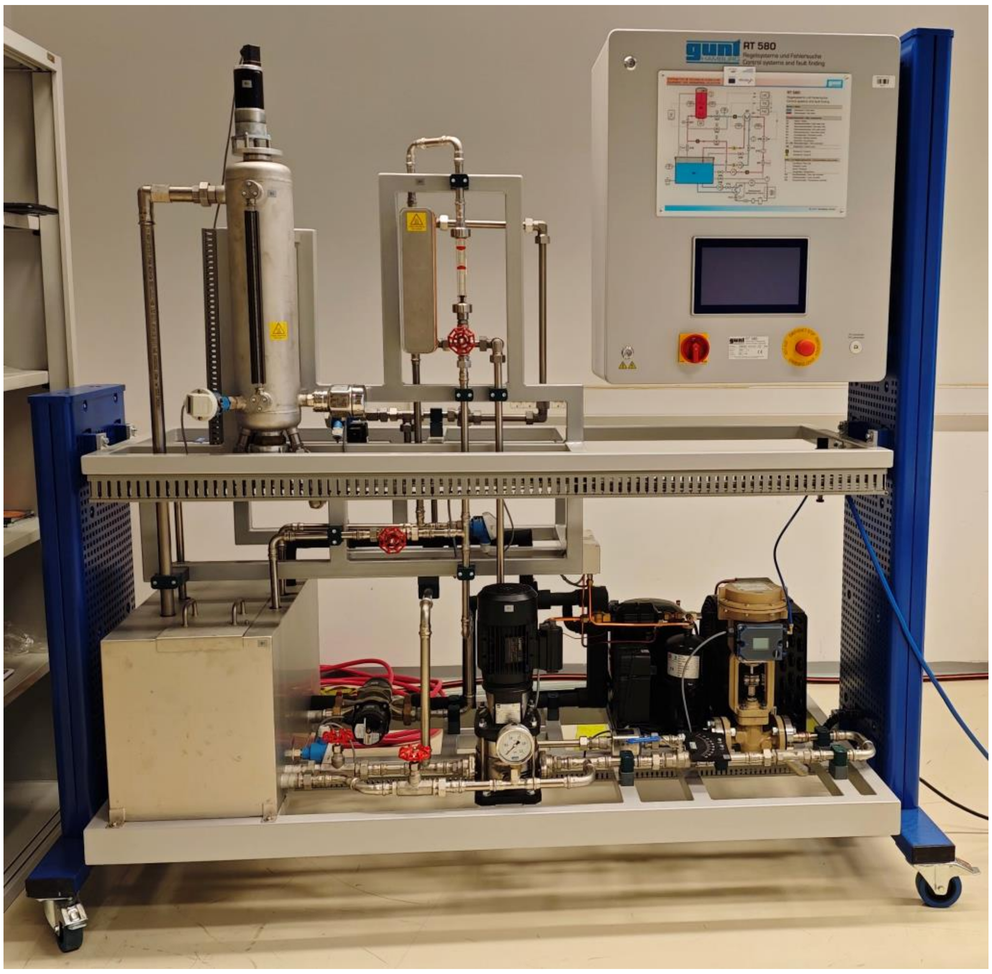


Figure 3.36 GUNT RT 580 multivariable process in a laboratory

3.8.2. Data acquisition

Process variables and control signals of the GUNT RT 580 system were recorded through the data-acquisition interface at a fixed sampling rate of 2 samples per second, following the acquisition procedure used in the proof-of-concept phase described in Section 3.2.2. The acquisition workflow is illustrated by the flowchart shown in Figure 3.37.

The resulting multivariable time series were segmented into two independent non-overlapping datasets, comprising 2000, and 200 samples, respectively. These datasets were prepared for subsequent ANN-based model development, evaluation, and validation.

All measured variables were time-stamped at acquisition to maintain synchronisation between manipulated variables and process variables across both flow and thermal subsystems.

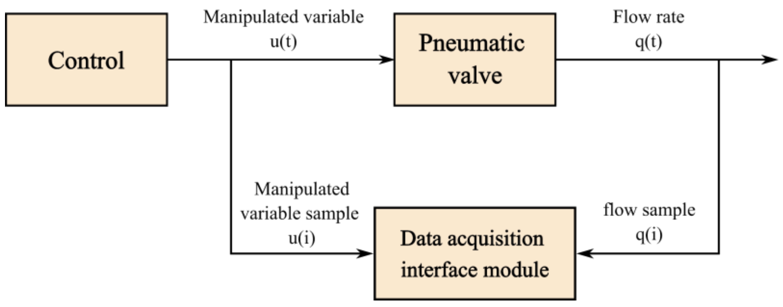

Control
Manipulated variable u(t)
Pneumatic valve
Flow rate q(t)
Manipulated variable sample u(i)
Data acquisition interface module
flow sample q(i)

Figure 3.37 Data sampling flowchart using data acquisition equipment

3.8.3. Model development

The primary objective of this section is to design and evaluate an ANN-based NARX model of the multivariable system. A series of modelling experiments was performed to identify suitable input and output delay orders (m, n) for the ANN-based NARX model, therefore determining the appropriate number of delayed samples for both manipulated and process variable. In total, 2520 ANN-based NARX models were developed and tested.

The ANN-based NARX model, introduced in Section 3.4, was adopted for modelling the nonlinear dynamic behaviour of the multivariable system described in Section 3.8.1. In discrete time, the model represents the next output as a nonlinear function of delayed output and input samples, according to the formulation given by Equation (3.4).

Here, $u(t)$ denotes the vector of manipulated variable and $y(t)$ denotes the vector of measured process variable at time step t. The delay orders n and m define the number of past input and output samples included in the regressor vector, with $n \geq 0$ and $m \geq 1$. The nonlinear function $F(\cdot)$ is realised using a feedforward Artificial Neural Network in the form of a multilayer perceptron with a single hidden layer and sigmoid activation functions, as described in Section 3.4.

The model was implemented in MATLAB as a feedforward ANN. Training was performed using the Bayesian Regularization (BR) algorithm with backpropagation as the optimiser, employing the same training configuration as in the proof-of-concept study. The network inputs consist of delayed time series of the manipulated variable and process variable of the multivariable system, while the network outputs correspond to the current samples of the flow as a process variable, according to the Equation (3.5).

To determine suitable delay orders (m, n) for the multivariable system, multiple training iterations were performed following the same workflow as shown in Figure 3.12. To reduce sensitivity to random initialisations and to avoid convergence to local minima of the cost function during training, each network structure was trained using a multi-start strategy with 10 random initialisations. Each training run was limited to 10000 epochs. From the available dataset, samples were divided into training, validation, and test subsets using the same 70%, 15%, and 15% split ratios adopted in the proof-of-concept modelling phase.

For each candidate delay configuration, a set of ANN-based NARX models was developed and stored in the MATLAB workspace, together with the corresponding delay orders. The mean-square error (MSE), defined by Equation (3.6), was calculated for each trained network during the training process and stored for subsequent ranking and comparison of candidate models.

Training was conducted on a Windows 10 workstation equipped with an Intel i7-9700K processor, 32 GB DDR4-3200 RAM, and a 500 GB NVMe M.2 solid-state drive. Training the complete set of 2520 models required 46 hours and 15 minutes of computation time.

As mentioned earlier and noted in [191], when datasets are relatively small, k-fold cross-validation provides a robust basis for model assessment. In this approach, the data are partitioned into *k* folds and the ANN-based NARX model is trained repeatedly such that each fold serves once as part of the training process. This yields *k* independent estimates for each evaluation metric, such as MSE, GoF, and best fit, which are then averaged to produce a single, more reliable performance indicator than would be obtained from a single fold. Since there is no strict rule for choosing *k*, values in the range of 5-10 are commonly adopted [192]. In this research, the time-series dataset acquired on the multivariable system implies six distinct permutations of the data splits. Therefore, a permuted 6-fold procedure was carried out to balance bias and variance.

To minimise bias in both testing and validation, the networks were trained over all six permutations of the training, validation, and test assignments while preserving the 70/15/15 split ratio across the full dataset, as summarised by the flowchart shown in Figure 3.38. Accordingly, for each training run, 70% of the available samples were used for training, while the remaining 30% were divided equally between validation and test datasets. This procedure results in six non-overlapping permutations without repetition, which are listed in Table 3.16. Because each training iteration generates 420 ANN-based NARX models, combining these with the six dataset permutations results in a total of 2520 trained ANN-based NARX models.

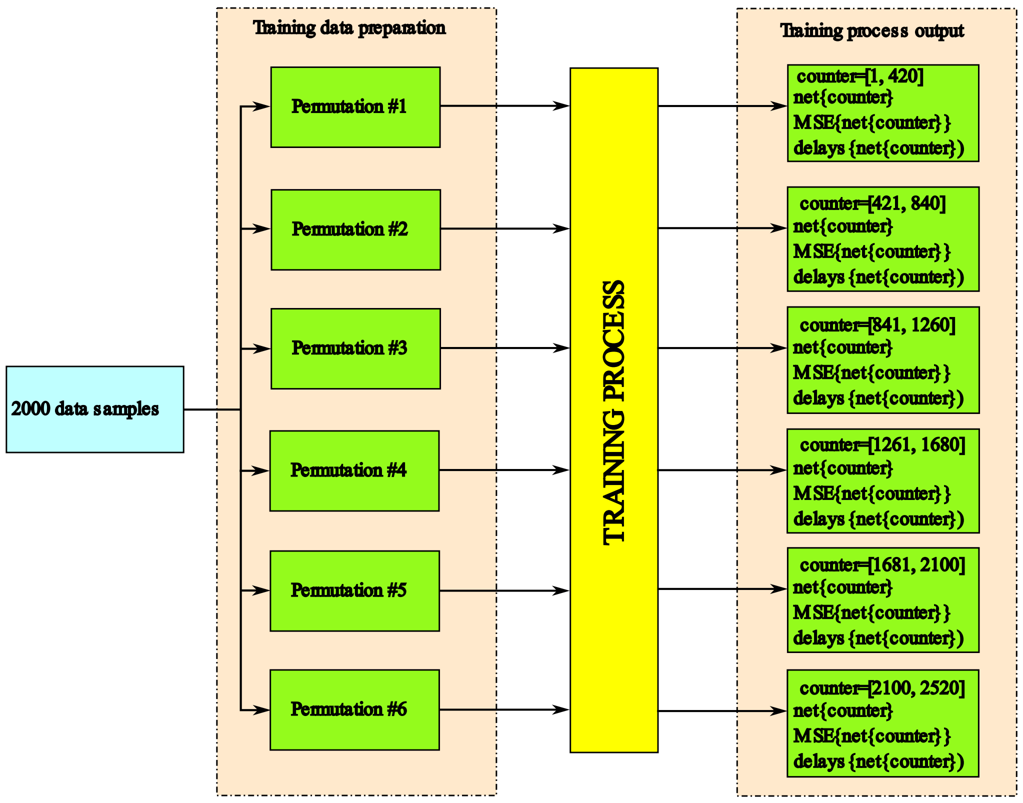

Training data preparation
Training process output
2000 data samples
Permutation #1
Permutation #2
Permutation #3
Permutation #4
Permutation #5
Permutation #6
TRAINING PROCESS
counter=[1, 420] net{counter} MSE{net{counter}} delays {net{counter})
counter=[421, 840] net{counter} MSE{net{counter}} delays {net{counter})
counter=[841, 1260] net{counter} MSE{net{counter}} delays {net{counter})
counter=[1261, 1680] net{counter} MSE{net{counter}} delays {net{counter})
counter=[1681, 2100] net{counter} MSE{net{counter}} delays {net{counter})
counter=[2100, 2520] net{counter} MSE{net{counter}} delays {net{counter})

Figure 3.38 Flowchart of the NARX model training process proposed in research

Table 3.16 Training, testing and validation dataset permutations within overall dataset consisting of 2000 data samples

Permutation number	Training dataset [sample sequence]	Test dataset [sample sequence]	Validation dataset [sample sequence]
1.	[1, 1400]	[1401, 1700]	[1701, 2000]
2.	[1, 1400]	[1701, 2000]	[1401, 1700]
3.	[301, 1700]	[1, 300]	[1701, 2000]
4.	[301, 1700]	[1701, 2000]	[1, 300]
5.	[601, 2000]	[1, 300]	[301, 600]
6.	[601, 2000]	[301, 600]	[1, 300]

3.8.4. Model evaluation and validation

For model evaluation, each trained network was evaluated on a time-series test dataset consisting of 200 samples, according to the flowchart shown in Figure 3.39. This dataset was never used in training nor internal validation, thus verifying generalisation to unseen operating data.

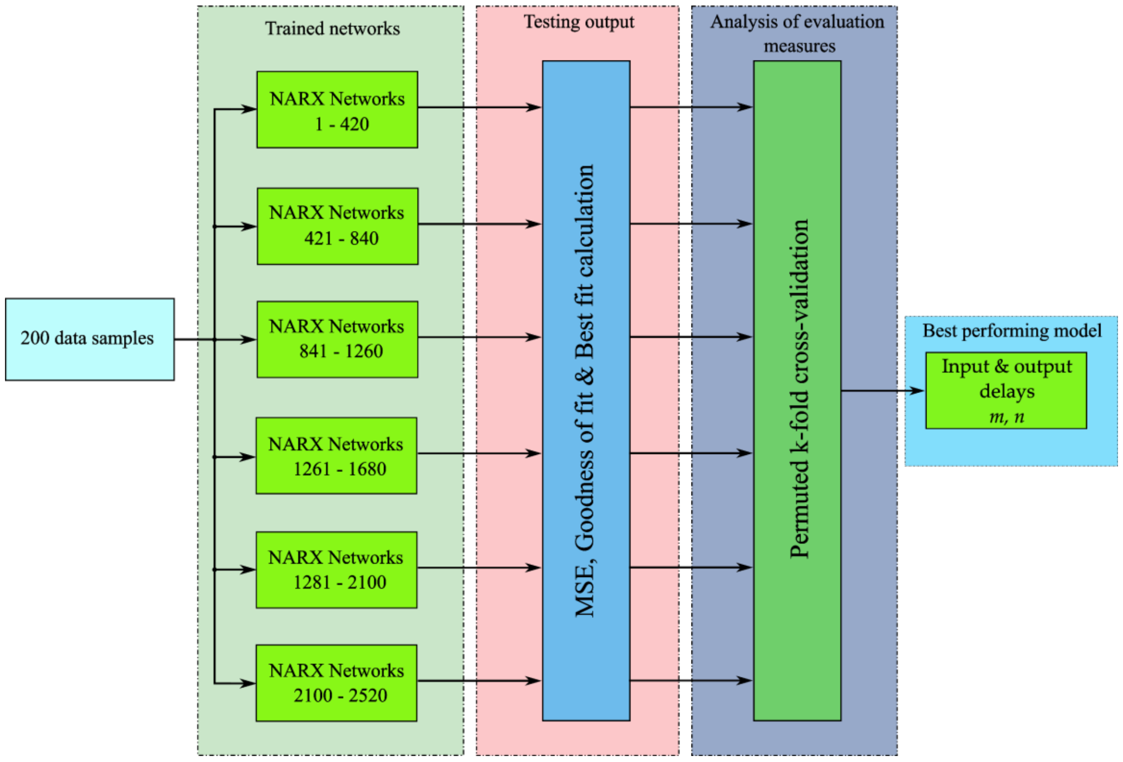

Trained networks
Testing output
Analysis of evaluation measures
200 data samples
NARX Networks 1 - 420
NARX Networks 421 - 840
NARX Networks 841 - 1260
NARX Networks 1261 - 1680
NARX Networks 1281 - 2100
NARX Networks 2100 - 2520
MSE, Goodness of fit & Best fit calculation
Permuted k-fold cross-validation
Best performing model
Input & output delays
m, n

Figure 3.39 Flowchart of the NARX model evaluation process using independent test dataset

Model evaluation was carried out using the same performance metrics employed during training, as defined earlier in this chapter. In line with the modelling framework adopted throughout this research, the mean-square error (MSE) was used as the primary criterion for quantifying prediction accuracy, while goodness-of-fit (GoF) and best-fit values were reported as complementary indicators.

Although multiple metrics were considered, a single decisive criterion was required for ranking candidate networks and selecting a representative model. Since MSE also serves as the training cost function, it was retained as the primary ranking metric for comparative performance assessment.

The ten best-performing ANN-based NARX models on independent test dataset are summarised in Table 3.17. The models are ranked according to MSE, with the corresponding GoF and best-fit values included for completeness. The ANN NARX number column indicates the training iteration and dataset permutation from which each model was obtained.

The time-series response of the best performing model, i.e., model number 1071 with 6 flow delays and 3 manipulated variable delays is shown in Figure 3.40.

Table 3.17 Evaluation of ANN-based NARX models on test dataset

Flow delay (m)	Manipulated variable delay (n)	ANN NARX number (permutation)	MSE	Best fit	GoF
6	3	1071 (3rd)	197.155	98.765	97.232
4	2	1000 (3rd)	204.114	98.882	97.212
5	0	468 (2nd)	204.513	98.518	97.195
3	1	926 (3rd)	204.993	98.624	97.220
5	1	521 (2nd)	205.300	98.525	97.190
4	1	520 (2nd)	205.931	98.593	97.200
4	0	459 (2nd)	206.426	98.480	97.196
5	1	528 (2nd)	206.877	98.550	97.180
6	1	533 (2nd)	210.931	98.526	97.137
4	1	933 (3rd)	211.553	98.616	97.162

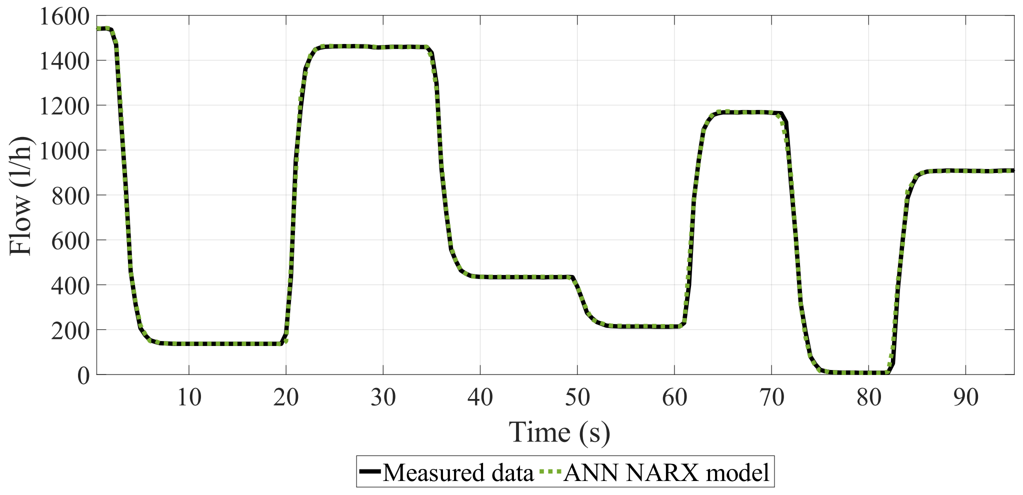


Figure 3.40 Time response of the of the best performing ANN-based NARX model on test dataset

The structure of the best-performing model is shown in Figure 3.41. As an input signal, current and two previous samples of manipulated variable u_t are used. As a feedback flow signal q_t, six previous samples are used. Hidden layer consists of two sets of weight matrices W for processing the signals and a bias vector b. These weighted inputs are summed and passed through a sigmoid activation function. The output layer applies additional set of weights W and biases b and a linear activation function to produce the flow sample q_t.

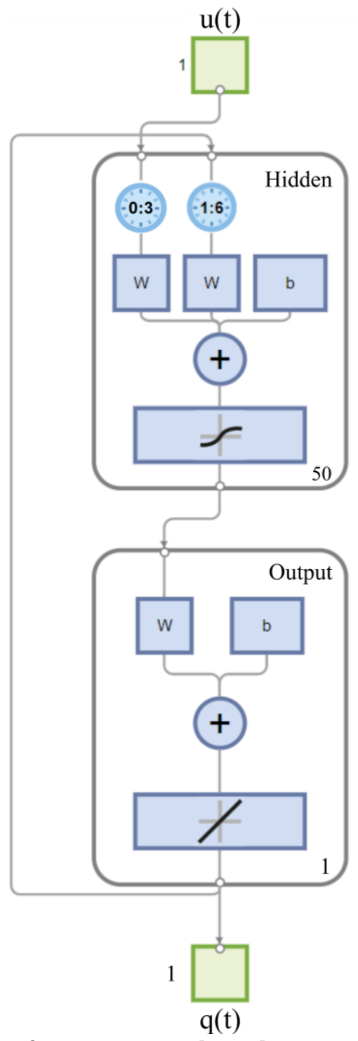


Figure 3.41 Best performing ANN-based NARX model architecture

3.9. APPLICATION OF DEVELOPED MODELS FOR CONTROLLER TUNING ON A COMPLEX FLOW CONTROL SYSTEM

This section presents the application of the developed ANN-based NARX model for controller tuning on the GUNT RT 580 multivariable flow control system. The objective is to assess how effectively the data-driven model supports controller design for a complex plant characterized by coupled dynamics. In contrast to the proof-of-concept system, the RT 580 exhibits interaction effects, particularly due to heat exchange capabilities of the system.

Controller performance is evaluated under two fixed thermal operating conditions, with the fluid temperature maintained at 15 °C and 50 °C, respectively. These conditions represent nominal and elevated operating regimes and allow systematic assessment of tuning robustness with respect to thermal state. For each operating condition, controller parameters were evaluated using identical setpoint sequences to ensure comparability across tuning approaches.

3.9.1. Controller tuning performed on the physical RT 580 system

To establish a reference for evaluating model-based controller tuning, the tuning procedures were also carried out directly on the physical RT 580 flow control system. Therefore, controller parameters in this case were obtained exclusively from the measured responses, without reliance on any identified model.

Tuning was performed according to the procedures described in Section 3.6, using either the system's open-loop step response or its closed-loop critical oscillation behaviour. The resulting characteristic parameters obtained from the ANN-based PRC analysis are summarized in Table 3.18.

Table 3.18 Characteristic parameters of the system-based PRC – Tangent method analysis results

Description	Symbol	Value
Input step amplitude	Δy	100 %
Change of PV	ΔPV	1547.40 l/h
Dead time	T_D	1.007 s
Time constant	T_C	1.723 s
Tuning parameter α	α	904.371
Tuning parameter τ	τ	0.3688

By experimenting on a system with integral and derivative action switched off in the PID controller, the system becomes marginally stable at critical gain K_{CR} = 0.01184 with an oscillation period of T_{CR} = 3.21 s. The resulting PI and PID controller parameters are presented in Table 3.19.

Table 3.19 Calculated controller tuning parameters – system-based tuning

Method	Controller	Kp	Ti	Td
ZN SR	PI	0.0010	3.021	0
	PID	0.0014	2.014	0.504
ZN CL	PI	0.0053	2.664	0
	PID	0.0069	1.605	0.401
CC	PI	0.0015	1.531	0
	PID	0.0016	2.073	0.335
CHR	PI	0.0004	2.068	0
	PID	0.0007	1.723	0.504

Following parameter identification, the PID controllers were implemented on the RT 580 system and tested using the same setpoint sequence employed in the model-based evaluations. The PI controllers were not presented since they underperformed compared to the PID controllers similarly to the results presented in the proof-of-concept phase. The closed-loop responses of the physical RT 580 system obtained using PID controllers tuned directly on the system is presented in Figure 3.42 and 3.43 for the two temperature conditions. In all cases, the controllers operated in standard setpoint regulation mode, adjusting the flow rate to track reference values.

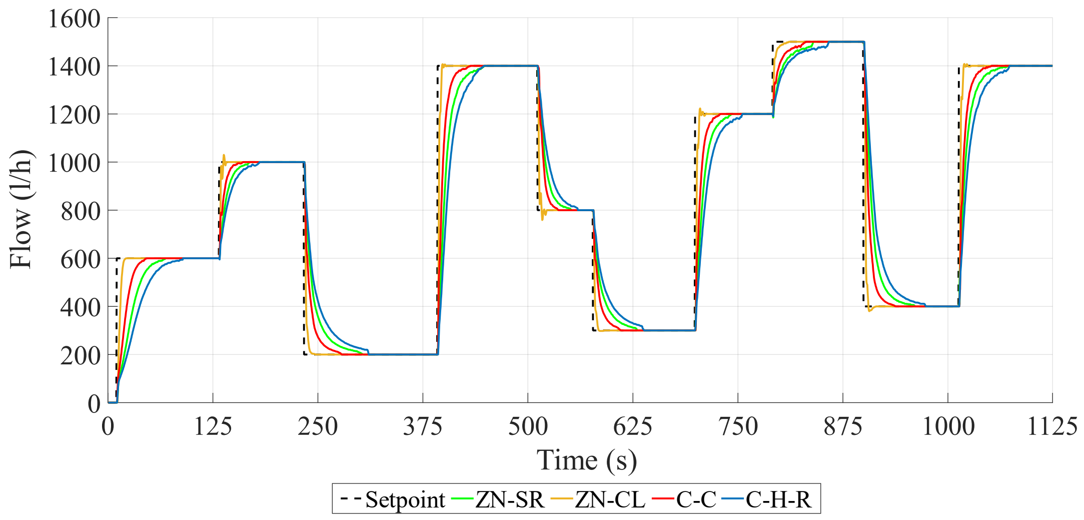


Figure 3.42 System response to the applied setpoint sequence for PID controllers tuned using four tuning methods and system-based tuning – fluid temperature 15°C

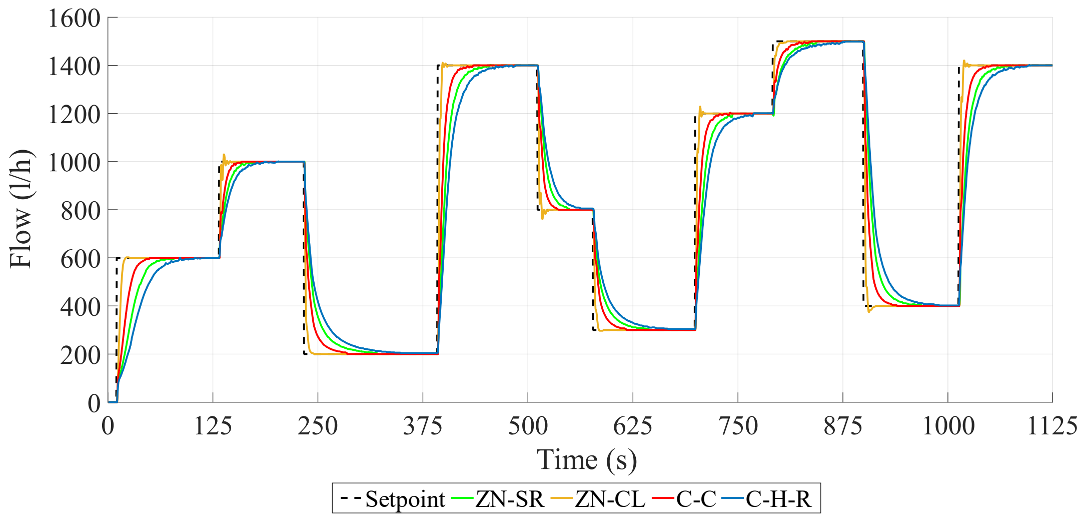


Figure 3.43 System response to the applied setpoint sequence for PID controllers tuned using four tuning methods and system-based tuning – fluid temperature 50°C

This ensured direct comparability between system-based and model-based tuning approaches under same operating conditions. RMSE values were calculated for both temperature levels and are summarized in Table 3.20.

Table 3.20 Evaluation results of the system-based tuning of the PID controllers tested on the system at two temperature conditions

	RMSE	
	T = 15°C	**T = 50°C**
ZN CL	113.78	111.10
CC	143.50	138.25
ZN SR	175.22	177.50
CHR	205.53	202.76

3.9.2. Controller tuning using the ANN model

Following the identification and validation of the ANN-based NARX model in Chapter 3.8, this subsection examines its application for calculating PI and PID controller parameters for the RT 580 flow control system. In this context, the ANN-based NARX model is employed as a digital twin of the physical process.

The same tuning procedures previously introduced in Section 3.6 are applied to the ANN model to ensure methodological consistency. The calculated PID controller parameters were evaluated using the same setpoint sequences on the physical RT 580 system under corresponding temperature conditions. This approach enables assessment of the ANN-based NARX model to determine how model-derived controller parameters translate to the real plant.

The controller parameter calculation procedure follows the workflow previously introduced in Chapter 3.7 and is therefore not repeated here. The step-response analysis was performed by exciting the selected best-performing ANN-based NARX model with a step change in the manipulated variable.

Analysis of the PRC using the tangent method enables the extraction of the characteristic process parameters, namely the process dead time T_D, time constant T_C, and the change of the process variable ΔPV. The tangent was constructed at the inflection point of the simulated step response, following the same procedure as described earlier for the transfer function-based model. From the identified values of T_D, T_C, and ΔPV, the tuning parameters α and τ were calculated according to Equations (3.9) and (3.10).

The resulting characteristic parameters obtained from the ANN-based PRC analysis are summarized in Table 3.21. These values form the basis for the calculation of PI and PID controller parameters using the ZN SR, Cohen-Coon, and Chien-Hrones-Reswick tuning formulas presented in Table 3.7.

Using MATLAB’s margin() function, the gain margin, representing the gain value at which the system becomes marginally stable, was obtained and interpreted as the critical gain required by the ZN CL procedure. The corresponding model-based response yielded K_{CR} = 0.010064 with an oscillation period of T_{CR} = 3.049 s. The resulting PI and PID controller parameters are presented in Table 3.22.

Table 3.21 Characteristic parameters of the ANN NARX model-based PRC – Tangent method analysis results

Description	Symbol	Value
Input step amplitude	Δy	100 %
Change of PV	ΔPV	1546.55 l/h
Dead time	T_D	1.029 s
Time constant	T_C	1.682 s
Tuning parameter α	α	946.136
Tuning parameter τ	τ	0.3796

Table 3.22 Calculated controller tuning parameters – ANN-based tuning

Method	Controller	Kp	Ti	Td
ZN SR	PI	0.0009	3.087	0
	PID	0.0012	2.058	0.515
ZN CL	PI	0.0045	2.530	0
	PID	0.0059	1.525	0.381
CC	PI	0.0015	1.528	0
	PID	0.0016	2.103	0.341
CHR	PI	0.0004	2.018	0
	PID	0.0006	1.682	0.515

The closed-loop responses of the physical RT 580 system obtained using PID controllers tuned via the ANN-based NARX model is presented in Figure 3.44 and 3.45 for the two temperature conditions. In all cases, the controllers operated in standard setpoint regulation mode, adjusting the flow rate to track predefined reference values.

Quantitative performance evaluation was conducted using the RMSE metric introduced in Chapter 2.4. RMSE values were calculated for both temperature levels and are summarized in Table 3.23. This provides insight into the accuracy of the ANN-based tuning approach and its robustness with respect to temperature-induced nonlinearities.

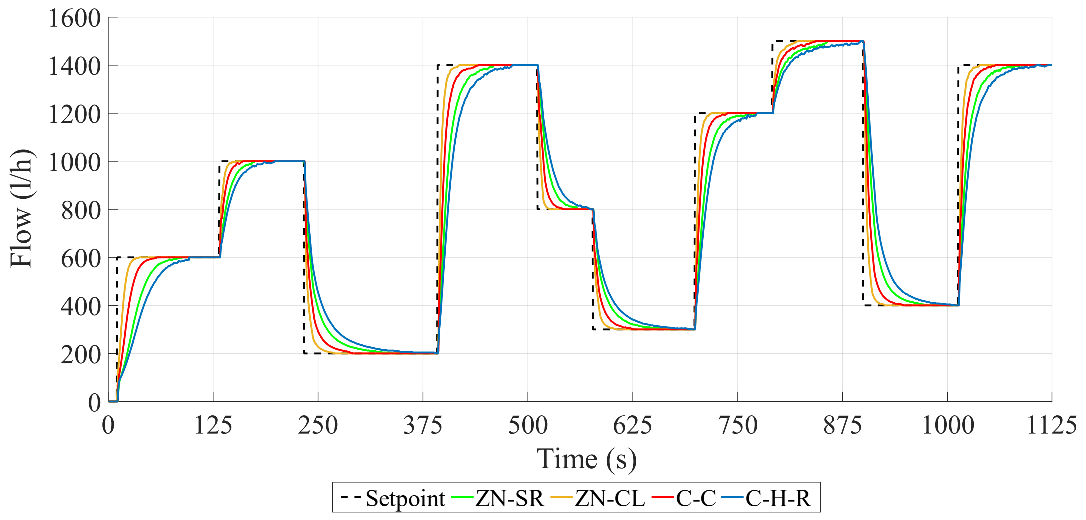


Figure 3.44 System response to the applied setpoint sequence for PID controllers tuned using four tuning methods based on ANN NARX model – fluid temperature 15°C

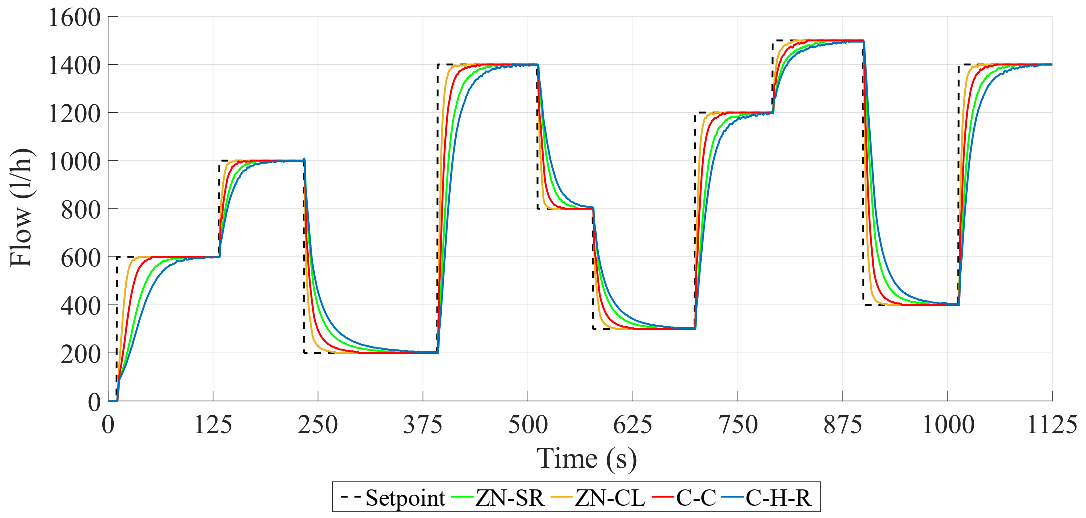


Figure 3.45 System response to the applied setpoint sequence for PID controllers tuned using four tuning methods based on ANN NARX model – fluid temperature 50°C

Table 3.23 RMSE values of the ANN NARX model-based tuning of the PID controllers tested on the system at two temperature conditions

	RMSE	
	T = 15°C	**T = 50°C**
ZN CL	121.00	116.35
CC	151.93	148.76
ZN SR	193.43	192.67
CHR	223.02	221.50

3.9.3. Comparative evaluation of tuning performance

The tuning procedures described in the previous subsections resulted in four PID controllers for each tuning approach and each temperature condition. To enable systematic comparison, controller performance was evaluated in two distinct cases:

I. Controllers tuned directly on the RT 580 system and tested on the RT 580 system.

II. Controllers tuned using the ANN-based NARX model and tested on the RT 580 system.

All evaluations were performed using the same setpoint sequence and repeated independently at 15 °C and 50 °C, allowing assessment of tuning robustness with respect to thermal operating point. Controller performance was quantified using the RMSE metric, calculated from the deviation between measured flow response and setpoint value.

A unified setpoint sequence was used across all scenarios to ensure consistency between model-based and system-based evaluations. This experimental design enabled assessment of both the performance of each tuning method and how the model-derived parameters translate to the system.

Quantitative performance evaluation was conducted using the root mean square error (RMSE) metric introduced in Section 2.4. These values were used to determine the most effective tuning method, evaluate the loss of accuracy when transferring controller parameters from the models to the real system, compare ANN-based tuning to system-based tuning, and to evaluate the robustness of tuned controllers in two temperature operating conditions. The results of these analyses are summarized in Table 3.24. and Table 3.25.

Table 3.24 RMSE values of the ANN model-based, and system-based tuned controllers tested on the system at two fluid temperature conditions

	System-based tuning		ANN model-based tuning	
	I		II	
	T = 15°C	T = 50°C	T = 15°C	T = 50°C
ZN CL	113.78	111.10	121.00	116.35
CC	143.50	138.28	151.93	148.76
ZN SR	175.22	177.49	193.43	192.67
CHR	205.53	202.76	223.02	221.50

Table 3.25 Percentage differences of the RMSE values of the ANN model-based and system-based tuned controllers tested on the system at two fluid temperature conditions

	Δ ($I_{15°C}$ - $I_{50°C}$) [%]	Δ ($II_{15°C}$ - $II_{50°C}$) [%]	Δ ($I_{15°C}$ - $II_{15°C}$) [%]	Δ ($I_{50°C}$ – $II_{50°C}$) [%]
ZN CL	-2.35	-3.85	+6.35	+8.55
CC	-3.66	-2.08	+5.87	+7.60
ZN SR	+1.30	-0.39	+10.39	+8.55
CHR	-1.35	-0.68	+8.51	+9.24

In Table 3.24, the performance of controllers tuned using the ANN-based NARX model (case II) is compared with controllers tuned directly on the physical system (case I), where both sets of parameters are evaluated on the real system under two fluid temperature conditions: T = 15°C and T = 50°C. Table 3.25 complements this comparison by reporting the percentage differences between cold and hot conditions for each tuning approach, i.e., $\Delta(II_{15°C} - II_{50°C})$ for ANN model-based tuning and $\Delta(I_{15°C} - I_{50°C})$ for system-based tuning, as well as the relative difference between tuning approaches at each temperature, given by $\Delta(I_{15°C} - II_{15°C})$ and $\Delta(I_{50°C} - II_{50°C})$. Positive values for $\Delta(I - II)$ indicate an increase of RMSE when moving from system-based tuning to an ANN model-based tuning, i.e., system-based tuning yields lower RMSE, while positive values for $\Delta(II_{15°C} - II_{50°C})$ and $\Delta(I_{15°C} - I_{50°C})$ indicate higher RMSE compared to the values at lower temperature.

The results in Table 3.24 show that, across all tuning methods, system-based tuning (case I) consistently achieves lower RMSE than ANN model-based tuning (case II) at both temperatures. The lowest RMSE values are obtained with the ZN CL method in all scenarios, followed by the CC method, while ZN SR and CHR yield notably larger errors. This is preserved in both temperature conditions, indicating that the relative ranking of tuning approaches is same with respect to temperature changes.

The temperature influence is summarized by $\Delta(II_{15°C} - II_{50°C})$ and $\Delta(I_{15°C} - I_{50°C})$ in Table 3.25. For ANN model-based tuning, RMSE decreases slightly when temperature increases from 15°C to 50°C for all methods, with the largest improvement occurring for ZN CL and the smallest for ZN SR. A similar trend is observed for system-based tuning, where RMSE generally decreases at 50°C for ZN CL, CC, and CHR. The ZN SR method is the only exception: $\Delta(I_{15°C} - I_{50°C})$ is positive, indicating a small degradation at higher temperature, consistent with the slightly larger RMSE at 50°C compared to 15°C. Overall, the magnitudes of these temperature-driven changes remain low, within 4%, thus indicating that controller performance is not greatly sensitive to the tested temperature range.

The differences between ANN model-based and system-based tuning at the same temperature are described by columns $\Delta(I_{15°C} - II_{15°C})$ and $\Delta(I_{50°C} - II_{50°C})$. These values quantify the deviation between the two tuning approaches when tested under same thermal conditions. The smallest differences occur for the ZN CL method, indicating that ANN-based tuning produces controller parameters closest to the system-tuned reference for this method. In contrast, the largest deviations are observed for ZN SR and CHR, implying that these tuning procedures are more sensitive to modelling inaccuracies and therefore benefit more strongly from direct tuning on the physical system. Overall, the comparisons confirm that system-based tuning remains the most accurate reference, while ANN model-based tuning provides consistently close performance.

The response plots illustrate these findings graphically for the ZN CL tuning approach using PID controllers at 15°C and 50°C in Figure 3.46 and Figure 3.47, respectively.

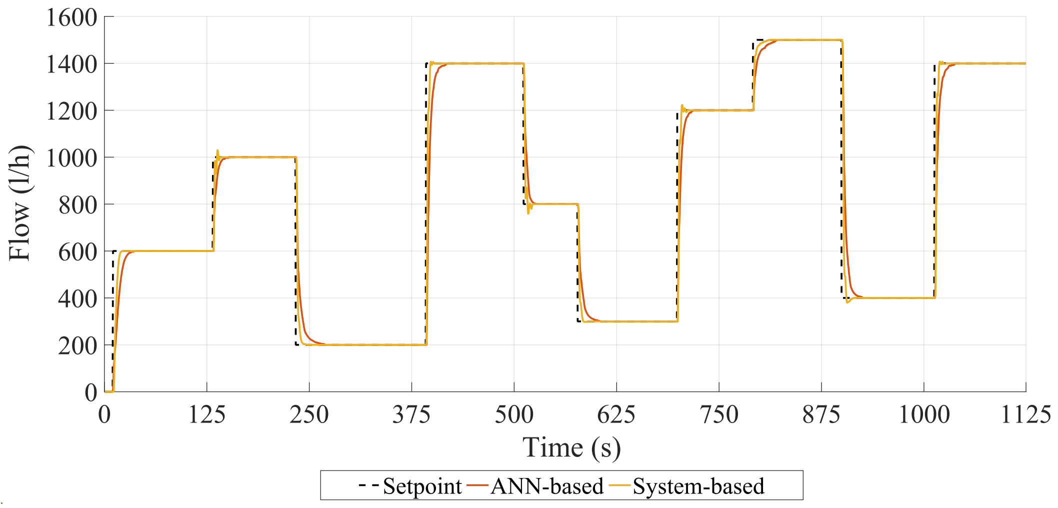


Figure 3.46 System response to the applied setpoint sequence for PID controllers tuned using the ZN CL method: ANN model-based, and system-based tuning, tested at T=15°C

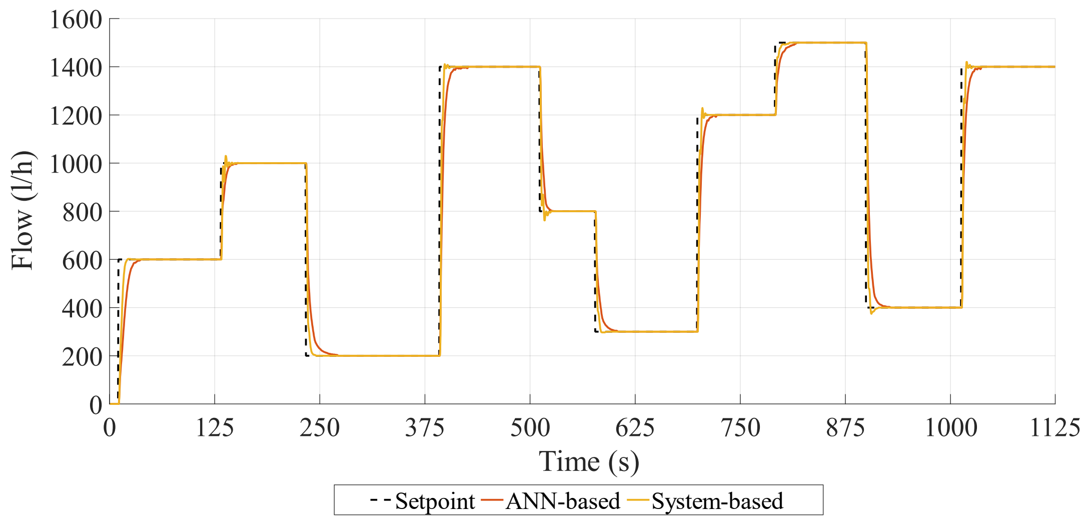


Figure 3.47 System response to the applied setpoint sequence for PID controllers tuned using the ZN CL method: ANN model-based, and system-based tuning, tested at T=50°C

3.10. DISCUSSION

This research investigated the applicability of data-driven modelling approaches as a foundation for Ship's Digital Twin (SDT) development, with the specific objective of supporting controller tuning without relying exclusively on direct experimentation on physical systems. Two modelling approaches were considered: a transfer function (TF) model and an ANN-based NARX model. Their suitability was investigated through controller tuning and experimental testing on the proof-of-concept flow control system GUNT RT 674 and further examined on the more complex GUNT RT 580 flow control unit, including performance evaluation under two temperature operating conditions.

The proof-of-concept evaluation on the RT 674 system highlights an important practical limitation of transfer-function-based modelling for controller tuning: even when a TF model provides a reasonable representation of system dynamics, controller parameters tuned on such a model may not transfer reliably to the real process due to nonlinearities, unmodelled dynamics, valve characteristics, and disturbances present in physical operation. This is clearly reflected in Table 3.14, where TF model-based tuning (case II) produced consistently higher RMSE values than both ANN model-based tuning (case III) and direct system-based tuning (case I).

For the ZN CL method, the TF model-based RMSE values were 59.03 (PI) and 61.92 (PID), while ANN model-based tuning improved values to 49.60 (PI) and 43.56 (PID). The lowest values were obtained with system-based tuning, 40.50 (PI) and 37.80 (PID). A similar trend can be noted for the ZN SR method, where TF model-based tuning resulted in 105.37 (PI) and 76.89 (PID), while ANN model-based tuning reduced the RMSE to 62.91 (PI) and 68.34 (PID), and system-based tuning achieved the lowest values of 59.16 (PI) and 54.82 (PID). These results demonstrate that ANN-based tuning significantly reduces the performance gap relative to TF-based tuning and produces controller behaviour closer to the system-based reference.

The percentage difference results in Table 3.15 further quantify these observations. The differences Δ(I−II) confirm that TF model-based tuning yields substantially larger RMSE values compared to system-based tuning, with increases up to +43.84% for ZN SR (PI) and +38.94% for ZN CL (PID). In contrast, Δ(I−III) values show that ANN model-based tuning is notably closer to system-based tuning, with smaller increases such as +5.95% for ZN SR (PI), +13.23% for ZN CL (PID), and +4.03% for CHR (PID). Finally, the Δ(III−II) values provide a direct comparison between ANN model-based and TF model-based tuning. In all tested cases, the values are negative, confirming improved performance when switching from TF-based tuning to ANN-based tuning. The strongest improvement occurs for ZN SR (PI) with -40.30%, and for ZN CL (PID) with -29.65%, indicating that the ANN-based model significantly improves tuning reliability for these cases.

Across the tuning methods, the Ziegler-Nichols closed-loop (ZN CL) method consistently yielded the lowest RMSE values in all three tuning scenarios, confirming it as the most reliable tuning strategy for both PI and PID control on the RT 674 system.

The response plots illustrate these findings graphically for the ZN CL tuning approach using PI and PID controllers, in Figure 3.33 and Figure 3.34, respectively. These figures show that ANN model-based tuned controllers track the setpoint more similarly to system-based tuning than TF model-based tuning, supporting the conclusion that the ANN-based model better represents the system dynamics relevant for control design.

The RT 580 evaluation extended the methodology to a larger and more complex system configuration and further analysed performance under temperature variation. Table 3.24 shows that system-based tuning remains the most accurate reference at both temperatures, producing lower RMSE values than ANN model-based tuning across all methods. For ZN CL method, ANN model-based RMSE values decreased from 121.00 (15°C) to 116.35 (50°C), while system-based tuning achieved lower values of 113.78 (15°C) and 111.10 (50°C). Similarly, for CC the ANN model-based RMSE was 151.93 (15°C) and 148.76 (50°C), compared with 143.50 (15°C) and 138.25 (50°C) for system-based tuning. The ZN SR method resulted in RMSE value of 175.22 (15°C) and 177.49 (50°C) for system-based tuning and 193.43 (15°C) and 192.67 (50°C) for ANN model-based tuning. For CHR, ANN model-based tuning resulted in 223.02

(15°C) and 221.50 (50°C), while system-based tuning achieved 205.53 (15°C) and 202.76 (50°C).

The effect of temperature is further quantified in Table 3.25. Within ANN model-based tuning, RMSE decreased in the hot condition for all tuning methods, with $\Delta(II_{15°C} - II_{50°C})$ ranging from -0.39% (ZN SR) to -3.85% (ZN CL). For system-based tuning, RMSE also decreased in the hot condition for most methods: -2.35% (ZN CL), -3.66% (CC), and -1.35% (CHR). The only exception is ZN SR, where RMSE slightly increased (+1.30%). Overall, these values indicate that temperature influences performance, but the changes remain within 4%, implying stable controller behaviour in both temperature conditions.

The approach differences at the same temperature show that system-based tuning continues to outperform ANN model-based tuning, with $\Delta(I_{15°C} - II_{15°C})$ and $\Delta(I_{50°C} - II_{50°C})$ ranging from +5.87% to +10.39% at 15°C and from +7.60% to +9.24% at 50°C.

Taken together, the two experimental phases demonstrate that ANN-based NARX modelling provides a stronger basis for digital-twin-supported controller tuning than transfer-function modelling. The proof-of-concept results show that ANN-based tuning substantially reduces the performance degradation observed when transferring TF-derived parameters to the real system, with improvements reaching 40.30% in RMSE reduction compared to TF-based tuning for ZN SR. The RT 580 validation further confirms that ANN-based tuning remains effective under operating-point changes such as temperature variation and achieves stable performance across both 15°C and 50°C conditions. Although system-based tuning remains the most accurate reference, ANN-based tuning offers comparable accuracy without extensive experimentation on the real system.

3.11. SUMMARY AND CONCLUDING REMARKS

This dissertation presented the development and experimental validation of modelling approaches intended to support a Ship’s Digital Twin (SDT) concept for controller tuning in flow-control processes. The main objective was to evaluate whether controller parameters obtained through model-based tuning, specifically using a transfer function (TF) model and an ANN-based NARX model, can be successfully transferred to real systems while maintaining acceptable performance. The evaluation was conducted in two stages: a proof-of-concept study using the RT 674 flow control unit and a validation study on the more complex RT 580 system, including assessment under two different fluid temperature conditions.

In the proof-of-concept phase on RT 674, controllers tuned using the TF model, ANN model, and directly on the physical system were compared using RMSE as a performance metric. The results demonstrated that system-based tuning consistently provided the lowest RMSE values and therefore represents the most accurate reference for controller performance. However, ANN model-based tuning achieved results significantly closer to the system-tuned controllers compared to TF model-based tuning. The advantage of ANN-based tuning is clearly reflected in the percentage differences. Across all tuning scenarios, the Ziegler-Nichols closed-loop method (ZN CL) produced the lowest RMSE values, confirming its effectiveness in both model-based and system-based implementations. The response plots for the ZN CL method further supported these conclusions by showing that ANN-based tuned controllers follow the setpoint more similarly to system-based tuning than TF-based tuned controllers.

The second phase extended the ANN-based methodology to the RT 580 flow control system and evaluated performance under two thermal operating points. The results again confirmed system-based tuning as the most accurate reference, while ANN model-based tuning achieved consistently close performance under both temperature conditions. Temperature sensitivity was shown to be modest in magnitude for most tuning methods: ANN model-based tuning exhibited small RMSE reductions at higher temperature, while system-based tuning showed similar trends. These findings indicate that both tuning approaches remain stable across the tested temperature range and that ANN-based tuning retains practical applicability even when operating conditions change.

Based on the results presented in this dissertation, the following conclusions can be highlighted:

1. ANN-based NARX modelling supports controller tuning more effectively than transfer-function modelling, producing controller parameters that transfer to real systems with smaller performance degradation.
2. The Ziegler-Nichols closed-loop method (ZN CL) consistently yielded the best performance across both systems, tuning approaches, and operating conditions.
3. Temperature variation between 15°C and 50°C influences RMSE values, but the effect is generally small, indicating stable closed-loop operation under both thermal conditions.
4. ANN model-based tuning provides a practical compromise between tuning accuracy and experimental effort, enabling tuning comparable to system-based tuning without extensive testing on the physical system.

In conclusion, this dissertation confirms that ANN-based NARX models are a suitable modelling foundation for SDT-supported control systems design in flow applications. While direct tuning on physical equipment remains superior in absolute performance, the ANN-based approach offers meaningful accuracy with reduced dependence on real-system experimentation. This makes proposed ANN-based digital twin development approach useful for supporting control systems design in maritime systems where onboard testing time, cost, and operational constraints limit the feasibility of experimental tuning.

4. DISSERTATION OUTLINE PROPOSAL

This chapter presents the proposed outline of the doctoral dissertation. The purpose of this chapter is to clearly define the structure of the dissertation and to establish a logical and coherent connection between its individual sections. The outlined structure reflects the research objectives, methodological approach, and experimental workflow adopted in this study.

The proposed content covers all major chapters and subchapters of the dissertation, highlighting the role and scope of each section. The structure is designed to guide the reader progressively from the theoretical background and problem formulation, through system modelling and experimental validation, to controller design and comparative performance evaluation.

4.1. EXPLANATION OF THE STRUCTURE OF THE DOCTORAL DISSERTATION

The doctoral dissertation is structured into six main chapters, each comprising several subchapters that collectively ensure a logical progression from problem definition and theoretical foundations to experimental validation and controller performance evaluation.

Chapter 1 – Introduction

Chapter 1 defines the research problem and establishes the scientific context of the dissertation. Subchapter 1.1 introduces the subject of research and formulates the core problem addressed in the study. Subchapter 1.2 presents the research hypotheses derived from identified gaps in existing literature. Subchapter 1.3 defines the research purpose and objectives, outlining the intended scientific contributions and practical relevance of the work.

Chapter 2 – Literature Review

Chapter 2 provides a comprehensive review of existing research relevant to the dissertation. Subchapter 2.1 examines digital twins in the maritime context, including conceptual foundations, communication and interoperability aspects, practical applications, and identified research gaps. Subchapter 2.2 focuses on Artificial Neural Networks, covering ANN architectures, activation functions, training algorithms, and their applications in ship systems, followed by a critical analysis of current research limitations. Subchapter 2.3 reviews system identification and modelling approaches, including data-driven, transfer-function-based, ANN-based, and hybrid models. Subchapter 2.4 introduces performance evaluation metrics used throughout the dissertation, while Subchapter 2.5 summarizes the key findings of the literature review and justifies the chosen research methodology.

Chapter 3 – Research Methodology and Model Development

Chapter 3 constitutes the core methodological and experimental part of the dissertation. Subchapter 3.1 introduces the Ship's Digital Twin (SDT) development framework adopted in the research. Subchapter 3.2 describes the proof-of-concept experimental setup, including the GUNT RT 674 flow control system and the associated data acquisition procedures.

Subchapters 3.3 and 3.4 present the development of transfer-function-based and ANN-based NARX models, respectively, including detailed descriptions of modelling approaches, training procedures, and validation methods. Subchapter 3.5 provides a comparative analysis of the two modelling approaches.

Subchapter 3.6 introduces PID-based controller tuning, describing the controller structure and tuning methods. Subchapter 3.7 applies the developed models to controller tuning on the proof-of-concept system, including model-based and system-based tuning and comparative performance evaluation.

Subchapter 3.8 extends the ANN-based modelling approach to a multivariable system, describing the GUNT RT 580 experimental setup, data acquisition, model development, and validation. Subchapter 3.9 applies the developed models for controller tuning on the complex RT 580 system, including ANN-based tuning, system-based tuning, and comparative evaluation of tuning performance.

Finally, Subchapters 3.10 and 3.11 provide a discussion of the obtained results and summarize the main findings and conclusions of the research methodology chapter.

Chapter 4 – Dissertation Outline Proposal

Chapter 4 presents the proposed structure of the doctoral dissertation. Subchapter 4.1 explains the role and content of each chapter and subchapter, clarifying the logical organization of the dissertation and its alignment with the research objectives.

Chapter 5 – Scientific Contribution

Chapter 5 outlines the scientific contributions of the dissertation. It highlights the methodological, theoretical, and practical advancements expected from the research, particularly in the areas of ANN-based modelling, digital twins, and controller tuning for complex flow control systems.

Chapter 6 – Literature

Chapter 6 contains the complete list of bibliographic references cited throughout the dissertation, formatted in accordance with the prescribed citation standards.

5. SCIENTIFIC CONTRIBUTION

The scientific contribution of this dissertation is focused on advancing the application of Digital Twin (DT) concepts in control engineering and maritime systems, particularly in cases where control system design must be performed under practical constraints. Although the DT concept is increasingly used across engineering disciplines, it still lacks a universally standardized definition, especially in practical and domain-specific implementations. As a consequence, the development and application of DTs in industry is often inconsistent, with no clear and repeatable guidelines that define what constitutes a DT and how it should be constructed for a specific purpose. This limitation is especially relevant in the maritime domain, where PID controller tuning remains necessary for numerous onboard flow processes such as oil, gas, and water transport, yet direct experimental tuning may be difficult to conduct on in-service ship systems due to safety requirements, limited accessibility, cost, and operational restrictions. This dissertation addresses these challenges by proposing and validating a structured methodology for DT development based on real measured data, aimed specifically at enabling controller tuning that can be transferred to the physical system with acceptable performance.

The primary scientific contribution of this work is the development and experimental validation of a DT-based controller tuning approach. Instead of treating the DT only as a simulation model, the dissertation evaluates the DT through a control-oriented application: whether controller parameters obtained through DT-based tuning provide stable and effective closed-loop behaviour when implemented on the real system. This approach contributes new insight compared to many existing studies that focus primarily on prediction accuracy or model fit, because it demonstrates that a model can appear satisfactory during open-loop validation yet remain inadequate for practical controller tuning. The research hypothesis formulated in this dissertation states that a DT developed using real system measurements enables remote controller tuning with performance comparable to experimental tuning, providing a practical solution for systems where direct tuning is challenging, such as in-service ships. In addition, auxiliary hypotheses define the need to identify an alternative modelling approach that overcomes TF limitations and to verify its applicability on a system of higher complexity. The presented results confirm these hypotheses by demonstrating that a suitable data-driven modelling strategy can be used not only in a proof-of-concept phase but also when transferred to a more complex system and variable operating conditions.

A key contribution of this dissertation is the systematic comparison between transfer function (TF) modelling and ANN-based modelling in the context of system identification for DT-supported controller tuning. TF identification remains widely used due to its simplicity and interpretability. However, the results of this research demonstrate a significant limitation: TF models can provide reasonable validation results but tend to be inefficient for controller tuning purposes when tuning parameters must be transferred to the physical system. This finding has scientific relevance as it highlights the difference between identification success and control usefulness, and it provides experimental evidence that the model evaluation criteria should be defined according to the intended DT application. In response to this limitation, the dissertation proposes a data-driven modelling framework and validates ANN-based NARX modelling as a more suitable alternative for practical DT control applications. The ANN-based approach better captures nonlinear and time-dependent behaviour, resulting in tuning parameters that are closer to those obtained through direct system tuning and therefore more appropriate for remote tuning support.

An additional methodological contribution of this dissertation is that it addresses the lack of consistent DT development guidelines by proposing a systematic DT implementation procedure in the form of a four-step iterative framework. The framework provides a structured path from real-system measurements to model construction, validation, controller tuning, and refinement, enabling repeatable DT development for control-oriented applications. This contribution is significant as it translates the broad and often ambiguous DT concept into an implementation methodology that can be used and extended in future research and practical engineering work. Furthermore, the implementation of the proposed DT framework is performed in MATLAB and Simulink, which supports reproducibility and makes the methodology accessible for both academic research and industrial practice.

In terms of professional and practical impact, the dissertation contributes by demonstrating the feasibility of remote tuning support for systems where direct tuning experiments are challenging. The proposed methodology can reduce the need for extensive experimental tuning campaigns, reduce downtime during commissioning and maintenance, and improve safety by limiting the amount of testing performed directly on operational equipment. This is particularly relevant for in-service maritime systems, where tuning tasks must often be performed under time pressure and operational constraints. The validated methodology also provides a foundation for broader application to complex ship systems characterized by nonlinear behaviour and coupled dynamics, and therefore supports the future development of DT-based tools for control design within maritime engineering field.

Finally, the dissertation provides a basis for further scientific work. Future research should focus on extending the proposed DT methodology to more complex ship-relevant systems such as thermal control loops, cooling networks, fuel conditioning processes, expanding the operating environment to include a wider range of disturbances and environmental conditions, and investigating online DT updating to account for system aging, wear, and long-term drift. In addition, the DT-based tuning workflow can be strengthened through the integration of neural network-based tuning approaches and adaptive control strategies.

Overall, this dissertation contributes scientifically by developing and validating a DT-based controller tuning approach, comparing TF and ANN-based models for system identification, identifying TF modelling limitations in practical control applications, proposing a structured data-driven DT development framework based on a four-step iterative procedure, demonstrating an ANN-based NARX model implemented in MATLAB and Simulink for reproducible remote tuning, and establishing a foundation for application to more complex maritime systems and future research in ship’s digital twin development.

6. LITERATURE

[1] Í. A. Fonseca and H. M. Gaspar, “Challenges when creating a cohesive digital twin ship: a data modelling perspective,” 2020. doi: 10.1080/09377255.2020.1815140.

[2] J. Šoda, M. Majić, I. Vujović, and B. Sorić, “An Overview on a Future Trends and Smart Technologies in Maritime,” in *8th International Maritime Science Conference (IMSC 2019),* Kotor: University of Montenegro, Faculty of Maritime Studies in Kotor ; University of Split, Faculty of Maritime Studies in Split, 2019, pp. 647–653.

[3] J. Lee, B. Bagheri, and H. A. Kao, “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems,” *Manuf. Lett.*, vol. 3, pp. 18–23, 2015, doi: 10.1016/j.mfglet.2014.12.001.

[4] J. Wan, A. Canedo, and M. A. Al Faruque, “Functional Model-Based Design Methodology for Automotive Cyber-Physical Systems,” *IEEE Syst. J.*, vol. 11, no. 4, pp. 2028–2039, 2015, doi: 10.1109/jsyst.2014.2387487.

[5] K. M. Alam, A. Sopena, and A. El Saddik, “Design and Development of a Cloud Based Cyber-Physical Architecture for the Internet-of-Things,” *Proceedings - 2015 IEEE International Symposium on Multimedia, ISM 2015*, pp. 459–464, 2016, doi: 10.1109/ISM.2015.96.

[6] K. M. Alam and A. El Saddik, “C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems,” *IEEE Access*, vol. 5, pp. 2050–2062, 2017, doi: 10.1109/ACCESS.2017.2657006.

[7] R. Saracco, “Digital Twins: Bridging Physical Space and Cyberspace,” *Computer (Long Beach. Calif).*, vol. 52, no. 12, pp. 58–64, 2019, doi: 10.1109/MC.2019.2942803.

[8] P. Matić, I. Golub Medvešek, and T. Perić, “System Identification in Difficult Operating Conditions Using Artificial Neural Networks,” *Transactions on Maritime Science*, vol. 4, no. 2, pp. 105–112, Oct. 2015, doi: 10.7225/toms.v04.n02.001.

[9] L. Ljung, “System Identification Toolbox for use with MATLAB Network inference View project Parameter Varying Systems Identification View project,” 2014. [Online]. Available: www.mathworks.com

[10] C. Unsalan, D. Barkana, and H. Gurhan, “Constructing Transfer Function of a System,” in *Embedded Digital Control with Microcontrollers*, Wiley, 2021, pp. 131–149. doi: 10.1002/9781119576600.ch6.

[11] K. J. Keesman, *System Identification*, no. 9780857295217. in Advanced Textbooks in Control and Signal Processing. London: Springer London, 2011. doi: 10.1007/978-0-85729-522-4.

[12] L. Mudronja, P. Matić, and M. Katalinić, "Data-Based Modelling of Significant Wave Height in the Adriatic Sea," *Transactions on Maritime Science*, vol. 6, no. 1, pp. 5–13, Apr. 2017, doi: 10.7225/toms.v06.n01.001.

[13] B. N. Bond *et al.*, "Compact Modeling of Nonlinear Analog Circuits Using System Identification via Semidefinite Programming and Incremental Stability Certification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 8, pp. 1149–1162, Aug. 2010, doi: 10.1109/TCAD.2010.2049155.

[14] J. B. M. Santos and P. R. Barros, "Time domain identification for First-Order plus time-delay systems using frequency domain information," in *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, Nov. 2011, pp. 775–780. doi: 10.1109/IECON.2011.6119408.

[15] A. Tepljakov, E. Petlenkov, and J. Belikov, "FOMCON: Fractional-Order Modeling and Control Toolbox for MATLAB,"

[16] R. Malti, S. Victor, and A. Oustaloup, "Advances in system identification using fractional models," *J. Comput. Nonlinear Dyn.*, vol. 3, no. 2, Apr. 2008, doi: 10.1115/1.2833910.

[17] Y. Bao, J. M. Velni, and M. Shahbakhti, "Epistemic Uncertainty Quantification in State-Space LPV Model Identification Using Bayesian Neural Networks," *IEEE Control Syst. Lett.*, vol. 5, no. 2, pp. 719–724, Apr. 2021, doi: 10.1109/LCSYS.2020.3005429.

[18] X. Liu, T. Zhang, and X. Liu, "Robust identification method for LPV ARX systems and its application to a mechanical unit," *IEEE Access*, vol. 7, pp. 164418–164428, 2019, doi: 10.1109/ACCESS.2019.2952891.

[19] X. Liu, C. Wang, and G. Han, "LPV Time-Delay System Identification and Its Application to the Centralized Heat-Supply System," *IEEE Trans. Instrum. Meas.*, vol. 71, 2022, doi: 10.1109/TIM.2022.3155778.

[20] N. Assani, P. Matic, and D. Kezic, "Flow control process identification using Matlab's System Identification Toolbox," in *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*, IEEE, May 2022, pp. 1228–1232. doi: 10.1109/CoDIT55151.2022.9803906.

[21] N. Assani, P. Matić, and D. Kezić, "Evaluating the Performance of the Well-Known Controller Tuning Methods for the Flow Control Using the Process Model," in *9th 2023*

International Conference on Control, Decision and Information Technologies, CoDIT 2023, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 433–438. doi: 10.1109/CoDIT58514.2023.10284155.

[22] N. Assani, P. Matić, D. Kezić, and N. Pleić, “Modeling Fluid Flow in Ship Systems for Controller Tuning Using an Artificial Neural Network,” *J. Mar. Sci. Eng.*, vol. 12, no. 8, Aug. 2024, doi: 10.3390/jmse12081318.

[23] N. Assani, P. Matić, and M. Katalinić, “Ship’s Digital Twin—A Review of Modelling Challenges and Applications,” Jun. 01, 2022, *MDPI*. doi: 10.3390/app12126039.

[24] J. Grieves Michael and Vickers, “Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems,” in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, S. and A. A. Kahlen Franz-Josef and Flumerfelt, Ed., Cham: Springer International Publishing, 2017, pp. 85–113. doi: 10.1007/978-3-319-38756-7_4.

[25] N. Assani and P. Matic, “Evaluating the ANN Model Performance for PID Controller Tuning in Flow Process Control: A Comparative Study,” *IEEE Access*, vol. 13, pp. 88499–88508, 2025, doi: 10.1109/ACCESS.2025.3571222.

[26] E. H. Glaessgen and D. S. Stargel, “The digital twin paradigm for future NASA and U.S. Air force vehicles,” *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pp. 1–14, 2012, doi: 10.2514/6.2012-1818.

[27] M. Grieves, “Digital Twin: Manufacturing Excellence through Virtual Factory Replication,” *White paper*, vol. 1, pp. 1–7, 2014, [Online]. Available: https://www.researchgate.net/publication/275211047

[28] S. Boschert and R. Rosen, “Digital twin-the simulation aspect,” in *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and Their Designers*, 2016. doi: 10.1007/978-3-319-32156-1_5.

[29] Q. Qi *et al.*, “Enabling technologies and tools for digital twin,” *J. Manuf. Syst.*, vol. 58, no. August, pp. 3–21, 2021, doi: 10.1016/j.jmsy.2019.10.001.

[30] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, “Digital Twin in Industry: State-of-the-Art,” *IEEE Trans. Industr. Inform.*, vol. 15, no. 4, pp. 2405–2415, Apr. 2019, doi: 10.1109/TII.2018.2873186.

[31] A. Rasheed, O. San, and T. Kvamsdal, “Digital Twin: Values, Challenges and Enablers From a Modeling Perspective,” *IEEE Access*, vol. 8, pp. 21980–22012, 2020, doi: 10.1109/ACCESS.2020.2970143.

[32] W. Hofmann and F. Branding, “Implementation of an IoT- And cloud-based digital twin for real-time decision support in port operations,” *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 2104–2109, 2019, doi: 10.1016/j.ifacol.2019.11.516.

[33] G. N. Schroeder, C. Steinmetz, R. N. Rodrigues, A. Rettberg, and C. E. Pereira, “Digital Twin connectivity topologies,” *IFAC-PapersOnLine*, vol. 54, no. 1, pp. 737–742, 2021, doi: 10.1016/j.ifacol.2021.08.086.

[34] R. Minerva, G. M. Lee, and N. Crespi, “Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models,” *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1785–1824, 2020, doi: 10.1109/JPROC.2020.2998530.

[35] M. Liu, S. Fang, H. Dong, and C. Xu, “Review of digital twin about concepts, technologies, and industrial applications,” *J. Manuf. Syst.*, vol. 58, no. October 2019, pp. 346–361, 2021, doi: 10.1016/j.jmsy.2020.06.017.

[36] Modelica Association, “Functional Mock-up Interface.” Accessed: Oct. 17, 2025. [Online]. Available: https://fmi-standard.org/

[37] F. Perabo, D. Park, M. K. Zadeh, O. Smogeli, and L. Jamt, “Digital Twin Modelling of Ship Power and Propulsion Systems: Application of the Open Simulation Platform (OSP),” in *IEEE International Symposium on Industrial Electronics*, 2020. doi: 10.1109/ISIE45063.2020.9152218.

[38] N. Assani, P. Matić, and M. Katalinić, “Ship’s Digital Twin—A Review of Modelling Challenges and Applications,” Jun. 01, 2022, *MDPI*. doi: 10.3390/app12126039.

[39] E. VanDerHorn and S. Mahadevan, “Digital Twin: Generalization, characterization and implementation,” *Decis. Support Syst.*, vol. 145, p. 113524, Jun. 2021, doi: 10.1016/J.DSS.2021.113524.

[40] Y. Lu and X. Xu, “A digital twin reference model for smart manufacturing,” *Proceedings of International Conference on Computers and Industrial Engineering, CIE*, vol. 2018-Decem, no. May, 2018.

[41] K. Y. H. Lim, P. Zheng, and C. H. Chen, “A state-of-the-art survey of Digital Twin: techniques, engineering product lifecycle management and business innovation perspectives,” *J. Intell. Manuf.*, vol. 31, no. 6, pp. 1313–1337, 2020, doi: 10.1007/s10845-019-01512-w.

[42] P. Major, G. Li, H. Zhang, and H. P. Hildre, “Real-time digital twin of research vessel for remote monitoring,” *Proceedings - European Council for Modelling and Simulation, ECMS*, vol. 35, no. 1, pp. 159–164, 2021, doi: 10.7148/2021-0159.

[43] Q. Liu, B. Liu, G. Wang, and C. Zhang, “A comparative study on digital twin models,” *AIP Conf. Proc.*, vol. 2073, no. February, 2019, doi: 10.1063/1.5090745.

[44] A. Coraddu, L. Oneto, F. Baldi, F. Cipollini, M. Atlar, and S. Savio, “Data-driven ship digital twin for estimating the speed loss caused by the marine fouling,” *Ocean Engineering*, vol. 186, 2019, doi: 10.1016/j.oceaneng.2019.05.045.

[45] V. Arrichiello and P. Gualeni, “Systems engineering and digital twin: a vision for the future of cruise ships design, production and operations,” *International Journal on Interactive Design and Manufacturing*, vol. 14, no. 1, pp. 115–122, 2020, doi: 10.1007/s12008-019-00621-3.

[46] R. E. Galeev, A. V. Soloviev, and Y. S. Fedosenko, “Modeling and visualization of the forecast trajectory for the decision support system for controlling the movement of a river displacement vessel,” *J. Phys. Conf. Ser.*, vol. 2131, no. 3, 2021, doi: 10.1088/1742-6596/2131/3/032033.

[47] A. Danielsen-Haces, “Digital Twin Development,” 2018.

[48] T. Hulkkonen, T. Manderbacka, and K. Sugimoto, “Digital Twin for Monitoring Remaining Fatigue Life of Critical Hull Structures,” *18th Conference on Computer Applications and Information Technology in the Maritime Industries (COMPIT2019)*, no. April, pp. 415–427, 2019, [Online]. Available: https://www.researchgate.net/publication/332223195

[49] A. V. Grinek, I. P. Boychuk, A. M. Fishenko, N. V. Savosteenko, and O. N. Gerasimenko, “Investigation of the operation of a ship’s synchronous generator based on a numerical model,” *J. Phys. Conf. Ser.*, vol. 2061, no. 1, 2021, doi: 10.1088/1742-6596/2061/1/012004.

[50] Í. A. Fonseca and H. M. Gaspar, “Fundamentals of digital twins applied to a plastic toy boat and a ship scale model,” *Proceedings - European Council for Modelling and Simulation, ECMS*, vol. 34, no. 1, pp. 207–213, 2020, doi: 10.7148/2020-0207.

[51] B. L. Tofte, O. Vennemann, F. Mitchell, N. Millington, and L. McGuire, “How digital technology and standardisation can improve offshore operations,” *Proceedings of the Annual Offshore Technology Conference*, vol. 2019-May, pp. 1–11, 2019, doi: 10.4043/29225-ms.

[52] M. Liu, Q. Zhou, X. Wang, C. Yu, and M. Kang, “Voyage performance evaluation based on a digital twin model,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 929, no. 1, 2020, doi: 10.1088/1757-899X/929/1/012027.

[53] A. Bekker, “Exploring the blue skies potential of digital twin technology for a polar supply and research vessel,” *Marine Design XIII*, vol. 1, pp. 135–146, 2018.

[54] D. Radan, *Integrated Control of Marine Electrical Power Systems*. 2008.

[55] M. Ibrion, N. Paltrinieri, and A. R. Nejad, “Learning from failures in cruise ship industry: The blackout of Viking Sky in Hustadvika, Norway,” *Eng. Fail. Anal.*, vol. 125, no. February, p. 105355, 2021, doi: 10.1016/j.engfailanal.2021.105355.

[56] S. S. Johansen and A. R. Nejad, “On digital twin condition monitoring approach for drivetrains in marine applications,” *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*, vol. 10, pp. 1–10, 2019, doi: 10.1115/omae2019-95152.

[57] K. N. Anyfantis, “An abstract approach toward the structural digital twin of ship hulls: A numerical study applied to a box girder geometry,” *Proceedings of the Institution of Mechanical Engineers Part M: Journal of Engineering for the Maritime Environment*, vol. 235, no. 3, pp. 718–736, 2021, doi: 10.1177/1475090221989188.

[58] C. Dufour, Z. Soghomonian, and W. Li, “Hardware-in-the-Loop Testing of Modern On-Board Power Systems Using Digital Twins,” *SPEEDAM 2018 - Proceedings: International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, pp. 118–123, 2018, doi: 10.1109/SPEEDAM.2018.8445302.

[59] O. Bondarenko and T. Fukuda, “Development of a diesel engine’s digital twin for predicting propulsion system dynamics,” *Energy*, vol. 196, 2020, doi: 10.1016/j.energy.2020.117126.

[60] M. Manngård, W. Lund, and J. Björkqvist, “Using Digital Twin Technology to Ensure Data Quality in Transport Systems,” *Researchgate.Net*, no. March, 2020.

[61] A. Wunderlich, K. Booth, and E. Santi, “Hybrid Analytical and Data-Driven Modeling Techniques for Digital Twin Applications,” in *2021 IEEE Electric Ship Technologies Symposium (ESTS)*, IEEE, Aug. 2021, pp. 1–7. doi: 10.1109/ESTS49166.2021.9512364.

[62] M. Katalinić and J. Parunov, “Uncertainties of estimating extreme significantwave height for engineering applications depending on the approach and fitting technique-adriatic sea case study,” *J. Mar. Sci. Eng.*, vol. 8, no. 4, Apr. 2020, doi: 10.3390/JMSE8040259.

[63] Y. Ichimura, D. Dalaklis, M. Kitada, and A. Christodoulou, “Shipping in the era of digitalization: Mapping the future strategic plans of major maritime commercial actors,” *Digital Business*, vol. 2, no. 1, p. 100022, Mar. 2022, doi: 10.1016/j.digbus.2022.100022.

[64] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943, doi: 10.1007/BF02478259.

[65] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain.,” *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958, doi: 10.1037/h0042519.

[66] D. M. Pu, D. Q. Gao, and Y. B. Yuan, “A primal analysis system of brain neurons data,” *Scientific World Journal*, vol. 2014, 2014, doi: 10.1155/2014/348526.

[67] P. Matić, O. Bego, and M. Maleš, “Complex Hydrological System Inflow Prediction using Artificial Neural Network,” *Tehnicki vjesnik - Technical Gazette*, vol. 29, no. 1, pp. 172–177, Feb. 2022, doi: 10.17559/TV-20200721133924.

[68] S. Haykin, *Neural Networks and Learning Machines*, Third edit. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[69] A. Sherstinsky, “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network,” *Physica D*, vol. 404, p. 132306, Mar. 2020, doi: 10.1016/j.physd.2019.132306.

[70] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, IEEE, Aug. 2017, pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.

[71] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” pp. 1–11, Nov. 2015, doi: https://doi.org/10.48550/arXiv.1511.08458.

[72] K.-L. Du and M. N. S. Swamy, “Radial Basis Function Networks,” in *Neural Networks in a Softcomputing Framework*, London: Springer-Verlag, 2006, pp. 251–294. doi: 10.1007/1-84628-303-5_6.

[73] T. Kohonen, “Self-organized formation of topologically correct feature maps,” *Biol. Cybern.*, vol. 43, no. 1, pp. 59–69, 1982, doi: 10.1007/BF00337288.

[74] D. F. Specht, “A general regression neural network,” *IEEE Trans. Neural Netw.*, vol. 2, no. 6, pp. 568–576, 1991, doi: 10.1109/72.97934.

[75] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, “A comprehensive survey on support vector machine classification: Applications, challenges and trends,” *Neurocomputing*, vol. 408, pp. 189–215, Sep. 2020, doi: 10.1016/j.neucom.2019.10.118.

[76] N. Cristianini and B. Schölkopf, “Support Vector Machines and Kernel Methods: The New Generation of Learning Machines,” *AI Mag.*, vol. 23, no. 3, pp. 31–41, 2002, doi: 10.5555/765580.765585.

[77] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[78] T. Luong, H. Pham, and C. D. Manning, “Effective Approaches to Attention-based Neural Machine Translation,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2015, pp. 1412–1421. doi: 10.18653/v1/D15-1166.

[79] E. L. Thorndike, “The Law of Effect,” *Am. J. Psychol.*, vol. 39, no. 1/4, pp. 212–222, Dec. 1927, doi: 10.2307/1415413.

[80] S. Sharma, S. Sharma, and A. Athaiya, “ACTIVATION FUNCTIONS IN NEURAL NETWORKS,” *International Journal of Engineering Applied Sciences and Technology*, vol. 04, no. 12, pp. 310–316, May 2020, doi: 10.33564/IJEAST.2020.v04i12.054.

[81] K. Levenberg, “A Method for the Solution of Certain Non-Linear Problems in Least Squares,” *Q. Appl. Math.*, vol. 2, no. 2, pp. 164–168, 1944, [Online]. Available: https://www.jstor.org/stable/43633451

[82] M. Hudson, B. Martin, T. Hagan, and H. B. Demuth, “Deep Learning Toolbox^{TM} User’s Guide,” 2022. [Online]. Available: www.mathworks.com

[83] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “DBSCAN Revisited, Revisited,” *ACM Transactions on Database Systems*, vol. 42, no. 3, pp. 1–21, Aug. 2017, doi: 10.1145/3068335.

[84] F. J. Pineda, “Recurrent Backpropagation and the Dynamical Approach to Adaptive Neural Computation,” *Neural Comput.*, vol. 1, no. 2, pp. 161–172, Jun. 1989, doi: 10.1162/neco.1989.1.2.161.

[85] N. Ketkar, *Deep Learning with Python*. Berkeley, CA: Apress, 2017. doi: 10.1007/978-1-4842-2766-4.

[86] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, “Mini-batch gradient descent: Faster convergence under data sparsity,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, Dec. 2017, pp. 2880–2887. doi: 10.1109/CDC.2017.8264077.

[87] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, Cham: IEEE, 2017, pp. 1942–1948. doi: 10.1109/ICNN.1995.488968.

[88] S. Katoch, S. S. Chauhan, and V. Kumar, “A review on genetic algorithm: past, present, and future,” *Multimed. Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, Feb. 2021, doi: 10.1007/s11042-020-10139-6.

[89] P. Judea, *Heuristics: Intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., 1984. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0306457385901001

[90] M. Mohammed, M. B. Khan, and E. B. M. Bashier, *Machine Learning*. CRC Press, 2016. doi: 10.1201/9781315371658.

[91] J. Ghosh and A. Nag, *An Overview of Radial Basis Function Networks*, no. x. 2001. doi: 10.1007/978-3-7908-1826-0_1.

[92] F. M. A. Acosta, “Radial basis function and related models: An overview,” *Signal Processing*, vol. 45, no. 1, pp. 37–58, Jul. 1995, doi: 10.1016/0165-1684(95)00041-B.

[93] S. Ruder, “An overview of gradient descent optimization algorithms,” pp. 1–14, Sep. 2016, doi: doi.org/10.48550/arXiv.1609.04747.

[94] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980

[95] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: Theory and applications,” *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, Dec. 2006, doi: 10.1016/j.neucom.2005.12.126.

[96] M. Müller, “Dynamic Time Warping,” in *Information Retrieval for Music and Motion*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 69–84. doi: 10.1007/978-3-540-74048-3_4.

[97] A. Cutler, D. R. Cutler, and J. R. Stevens, “Random Forests,” in *Ensemble Machine Learning*, C. Zhang and Y. Ma, Eds., Boston, MA: Springer US, 2012, pp. 157–175. doi: 10.1007/978-1-4419-9326-7_5.

[98] E. Garyfallidis, M. Brett, M. M. Correia, G. B. Williams, and I. Nimmo-Smith, “QuickBundles, a Method for Tractography Simplification,” *Front. Neurosci.*, vol. 6, no. December, pp. 1–13, 2012, doi: 10.3389/fnins.2012.00175.

[99] N. Kumyaito and K. Tamee, “Trajectory clustering by gps tracking dataset using quickbundles,” *ICIC Express Letters, Part B: Applications*, vol. 11, no. 10, pp. 921–928, 2020, doi: 10.24507/icicelb.11.10.921.

[100] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational Inference: A Review for Statisticians,” *J. Am. Stat. Assoc.*, vol. 112, no. 518, pp. 859–877, Apr. 2017, doi: 10.1080/01621459.2017.1285773.

[101] A. Borji and L. Itti, “State-of-the-Art in Visual Attention Modeling,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 185–207, Jan. 2013, doi: 10.1109/TPAMI.2012.89.

[102] A. Navada, A. N. Ansari, S. Patil, and B. A. Sonkamble, “Overview of use of decision tree algorithms in machine learning,” in *2011 IEEE Control and System Graduate Research Colloquium*, IEEE, Jun. 2011, pp. 37–42. doi: 10.1109/ICSGRC.2011.5991826.

[103] C. Aytekin, “Neural Networks are Decision Trees,” Oct. 2022, doi: https://doi.org/10.48550/arXiv.2210.05189.

[104] C. Dao-Duc, H. Xiaohui, and O. Morère, “Maritime Vessel Images Classification Using Deep Convolutional Neural Networks,” in *Proceedings of the Sixth International Symposium on Information and Communication Technology*, New York, NY, USA: ACM, Dec. 2015, pp. 276–281. doi: 10.1145/2833258.2833266.

[105] M. Petković, I. Vujović, N. Kaštelan, and J. Šoda, “Every Vessel Counts: Neural Network Based Maritime Traffic Counting System,” *Sensors*, vol. 23, no. 15, p. 6777, Jul. 2023, doi: 10.3390/s23156777.

[106] T. Zhang and X. Zhang, “High-Speed Ship Detection in SAR Images Based on a Grid Convolutional Neural Network,” *Remote Sens. (Basel).*, vol. 11, no. 10, p. 1206, May 2019, doi: 10.3390/rs11101206.

[107] B. Hou *et al.*, “A Neural Network Based on Consistency Learning and Adversarial Learning for Semisupervised Synthetic Aperture Radar Ship Detection,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, no. c, pp. 1–16, 2022, doi: 10.1109/TGRS.2022.3142017.

[108] Zhang, Zhang, Shi, and Wei, “Depthwise Separable Convolution Neural Network for High-Speed SAR Ship Detection,” *Remote Sens. (Basel).*, vol. 11, no. 21, p. 2483, Oct. 2019, doi: 10.3390/rs11212483.

[109] Z. Shao, L. Wang, Z. Wang, W. Du, and W. Wu, “Saliency-Aware Convolution Neural Network for Ship Detection in Surveillance Video,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 3, pp. 781–794, Mar. 2020, doi: 10.1109/TCSVT.2019.2897980.

[110] X. Chen *et al.*, “Ship Type Recognition via a Coarse-to-Fine Cascaded Convolution Neural Network,” *Journal of Navigation,* vol. 73, no. 4, pp. 813–832, Jul. 2020, doi: 10.1017/S0373463319000900.

[111] Z. Kong, Y. Cui, W. Xiong, F. Yang, Z. Xiong, and P. Xu, “Ship Target Identification via Bayesian-Transformer Neural Network,” *J. Mar. Sci. Eng.*, vol. 10, no. 5, p. 577, Apr. 2022, doi: 10.3390/jmse10050577.

[112] S. Liu *et al.*, “Multi-Scale Ship Detection Algorithm Based on a Lightweight Neural Network for Spaceborne SAR Images,” *Remote Sens. (Basel).*, vol. 14, no. 5, p. 1149, Feb. 2022, doi: 10.3390/rs14051149.

[113] N. K. Mishra, A. Kumar, and K. Choudhury, “Deep Convolutional Neural Network based Ship Images Classification,” *Def. Sci. J.*, vol. 71, no. 2, pp. 200–208, Mar. 2021, doi: 10.14429/dsj.71.16236.

[114] A. Khellal, H. Ma, and Q. Fei, “Convolutional Neural Network Based on Extreme Learning Machine for Maritime Ships Recognition in Infrared Images,” *Sensors*, vol. 18, no. 5, p. 1490, May 2018, doi: 10.3390/s18051490.

[115] Y. Ren, J. Yang, Q. Zhang, and Z. Guo, “Ship recognition based on Hu invariant moments and convolutional neural network for video surveillance,” *Multimed. Tools Appl.*, vol. 80, no. 1, pp. 1343–1373, Jan. 2021, doi: 10.1007/s11042-020-09574-2.

[116] Z. Huang, B. Sui, J. Wen, and G. Jiang, “An Intelligent Ship Image/Video Detection and Classification Method with Improved Regressive Deep Convolutional Neural Network,” *Complexity*, vol. 2020, pp. 1–11, Apr. 2020, doi: 10.1155/2020/1520872.

[117] J. Zhao, Z. Zhang, W. Yu, and T.-K. Truong, “A Cascade Coupled Convolutional Neural Network Guided Visual Attention Method for Ship Detection From SAR Images,” *IEEE Access*, vol. 6, pp. 50693–50708, 2018, doi: 10.1109/ACCESS.2018.2869289.

[118] C. Chen, C. He, C. Hu, H. Pei, and L. Jiao, “A Deep Neural Network Based on an Attention Mechanism for SAR Ship Detection in Multiscale and Complex Scenarios,” *IEEE Access*, vol. 7, pp. 104848–104863, 2019, doi: 10.1109/ACCESS.2019.2930939.

[119] F. Wu, Z. Zhou, B. Wang, and J. Ma, “Inshore Ship Detection Based on Convolutional Neural Network in Optical Satellite Images,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 11, no. 11, pp. 4005–4015, Nov. 2018, doi: 10.1109/JSTARS.2018.2873190.

[120] C. Chen, C. He, C. Hu, H. Pei, and L. Jiao, “MSARN: A Deep Neural Network Based on an Adaptive Recalibration Mechanism for Multiscale and Arbitrary-Oriented SAR Ship Detection,” *IEEE Access*, vol. 7, pp. 159262–159283, 2019, doi: 10.1109/ACCESS.2019.2951030.

[121] A. Daranda, “A NEURAL NETWORK APPROACH TO PREDICT MARINE TRAFFIC,” Vilnius, Oct. 2016.

[122] X. Jin, J. Xiong, D. Gu, C. Yi, and Y. Jiang, “Research on Ship Route Planning Method Based on Neural Network Wave Data Forecast,” *IOP Conf. Ser. Earth Environ. Sci.*, vol. 638, no. 1, p. 012033, Feb. 2021, doi: 10.1088/1755-1315/638/1/012033.

[123] H. Zhang, X. Zhang, and R. Bu, “Radial Basis Function Neural Network Sliding Mode Control for Ship Path Following Based on Position Prediction,” *J. Mar. Sci. Eng.*, vol. 9, no. 10, p. 1055, Sep. 2021, doi: 10.3390/jmse9101055.

[124] F. Zhu, “Ship Short-Term Trajectory Prediction Based on RNN,” *J. Phys. Conf. Ser.*, vol. 2025, no. 1, p. 012023, Sep. 2021, doi: 10.1088/1742-6596/2025/1/012023.

[125] X. Chen, Y. Liu, K. Achuthan, and X. Zhang, “A ship movement classification based on Automatic Identification System (AIS) data using Convolutional Neural Network,” *Ocean Engineering*, vol. 218, no. October, p. 108182, Dec. 2020, doi: 10.1016/j.oceaneng.2020.108182.

[126] T. Yang, X. Wang, and Z. Liu, “Ship Type Recognition Based on Ship Navigating Trajectory and Convolutional Neural Network,” *J. Mar. Sci. Eng.*, vol. 10, no. 1, p. 84, Jan. 2022, doi: 10.3390/jmse10010084.

[127] T. Guo and L. Xie, “Research on Ship Trajectory Classification Based on a Deep Convolutional Neural Network,” *J. Mar. Sci. Eng.*, vol. 10, no. 5, p. 568, Apr. 2022, doi: 10.3390/jmse10050568.

[128] Y. Suo, W. Chen, C. Claramunt, and S. Yang, “A Ship Trajectory Prediction Framework Based on a Recurrent Neural Network,” *Sensors*, vol. 20, no. 18, p. 5133, Sep. 2020, doi: 10.3390/s20185133.

[129] J.-Y. Kim, J.-H. Lee, J.-H. Oh, and J.-S. Oh, “A Comparative Study on Energy Consumption Forecast Methods for Electric Propulsion Ship,” *J. Mar. Sci. Eng.*, vol. 10, no. 1, p. 32, Dec. 2021, doi: 10.3390/jmse10010032.

[130] D. Nguyen, R. Vadaine, G. Hajduch, R. Garello, and R. Fablet, “GeoTrackNet —A Maritime Anomaly Detector Using Probabilistic Neural Network Representation of AIS Tracks and A Contrario Detection,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5655–5667, Jun. 2022, doi: 10.1109/TITS.2021.3055614.

[131] T. Niksa-Rynkiewicz, N. Szewczuk-Krypa, A. Witkowska, K. Cpałka, M. Zalasiński, and A. Cader, “Monitoring Regenerative Heat Exchanger in Steam Power Plant by Making Use of the Recurrent Neural Network,” *Journal of Artificial Intelligence and*

Soft Computing Research, vol. 11, no. 2, pp. 143–155, Apr. 2021, doi: 10.2478/jaiscr-2021-0009.

[132] Y. Raptodimos and I. Lazakis, “Using artificial neural network-self-organising map for data clustering of marine engine condition monitoring applications,” *Ships and Offshore Structures*, vol. 13, no. 6, pp. 649–656, Aug. 2018, doi: 10.1080/17445302.2018.1443694.

[133] Y. Jiang, G. Lan, and Z. Zhang, “Ship engine detection based on wavelet neural network and FPGA image scanning,” *Alexandria Engineering Journal*, vol. 60, no. 5, pp. 4287–4297, Oct. 2021, doi: 10.1016/j.aej.2021.02.028.

[134] S. Liang, J. Yang, Y. Wang, and M. Wang, “Fuzzy neural network in condition maintenance for marine electric propulsion system,” in *2014 IEEE Conference and Expo Transportation Electrification Asia-Pacific (ITEC Asia-Pacific)*, IEEE, Aug. 2014, pp. 1–5. doi: 10.1109/ITEC-AP.2014.6941252.

[135] A. V. Le *et al.*, “Reinforcement learning-based optimal complete water-blasting for autonomous ship hull corrosion cleaning system,” *Ocean Engineering*, vol. 220, no. June, p. 108477, Jan. 2021, doi: 10.1016/j.oceaneng.2020.108477.

[136] P. Karvelis, G. Georgoulas, V. Kappatos, and C. Stylios, “Deep machine learning for structural health monitoring on ship hulls using acoustic emission method,” *Ships and Offshore Structures*, vol. 16, no. 4, pp. 440–448, Apr. 2021, doi: 10.1080/17445302.2020.1735844.

[137] Y. Raptodimos and I. Lazakis, “An ANN approach for predicting the performance of machinery equipment,” in *International Conference on Maritime Safety and Operations*, Glasgow, UK, 2016, pp. 95–101.

[138] Y. Zhou, P. Yang, P. Fang, and N. Chen, “Research on Fault Diagnosis of Ship Power Station Based on Random Forest,” in *2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, IEEE, Aug. 2021, pp. 163–167. doi: 10.1109/AEECA52519.2021.9574252.

[139] C. Yu, L. Qi, J. Sun, C. Jiang, J. Su, and W. Shu, “Fault Diagnosis Technology for Ship Electrical Power System,” *Energies (Basel).*, vol. 15, no. 4, p. 1287, Feb. 2022, doi: 10.3390/en15041287.

[140] Y. Raptodimos and I. Lazakis, “Application of NARX neural network for predicting marine engine performance parameters,” *Ships and Offshore Structures*, vol. 15, no. 4, pp. 443–452, Apr. 2020, doi: 10.1080/17445302.2019.1661619.

[141] L. Moreira, R. Vettor, and C. Guedes Soares, “Neural Network Approach for Predicting Ship Speed and Fuel Consumption,” *J. Mar. Sci. Eng.*, vol. 9, no. 2, p. 119, Jan. 2021, doi: 10.3390/jmse9020119.

[142] T. Kawai, Y. Kawamura, T. Okada, T. Mitsuyuki, and X. Chen, “Sea state estimation using monitoring data by convolutional neural network (CNN),” *J. Mar. Sci. Technol.*, vol. 26, no. 3, pp. 947–962, Sep. 2021, doi: 10.1007/s00773-020-00785-8.

[143] Yao Zhang, G. E. Hearn, and P. Sen, “A neural network approach to ship track-keeping control,” *IEEE Journal of Oceanic Engineering*, vol. 21, no. 4, pp. 513–527, 1996, doi: 10.1109/48.544061.

[144] T. T. Le, “Ship heading control system using neural network,” *J. Mar. Sci. Technol.*, vol. 26, no. 3, pp. 963–972, Sep. 2021, doi: 10.1007/s00773-020-00783-w.

[145] R. Lou, W. Wang, X. Li, Y. Zheng, and Z. Lv, “Prediction of Ocean Wave Height Suitable for Ship Autopilot,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2021, doi: 10.1109/TITS.2021.3067040.

[146] Y. Shuai *et al.*, “An efficient neural-network based approach to automatic ship docking,” *Ocean Engineering*, vol. 191, no. October, p. 106514, Nov. 2019, doi: 10.1016/j.oceaneng.2019.106514.

[147] J. Yu, R. Bu, and L. Li, “Ship RBF neural network sliding mode PID heading control,” *MATEC Web of Conferences*, vol. 355, p. 03064, Jan. 2022, doi: 10.1051/matecconf/202235503064.

[148] X. Cheng, G. Li, A. L. Ellefsen, S. Chen, H. P. Hildre, and H. Zhang, “A Novel Densely Connected Convolutional Neural Network for Sea-State Estimation Using Ship Motion Data,” *IEEE Trans. Instrum. Meas.*, vol. 69, no. 9, pp. 5984–5993, Sep. 2020, doi: 10.1109/TIM.2020.2967115.

[149] M.-C. Fang, Y.-Z. Zhuo, and Z.-Y. Lee, “The application of the self-tuning neural network PID controller on the ship roll reduction in random waves,” *Ocean Engineering*, vol. 37, no. 7, pp. 529–538, May 2010, doi: 10.1016/j.oceaneng.2010.02.013.

[150] H. Jeon and J. Kim, “Application of Reference Voltage Control Method of the Generator Using a Neural Network in Variable Speed Synchronous Generation System of DC Distribution for Ships,” *J. Mar. Sci. Eng.*, vol. 8, no. 10, p. 802, Oct. 2020, doi: 10.3390/jmse8100802.

[151] Y. Qian, D. Hu, Y. Chen, Y. Fang, and Y. Hu, “Adaptive Neural Network-Based Tracking Control of Underactuated Offshore Ship-to-Ship Crane Systems Subject to

Unknown Wave Motions Disturbances,” *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 6, pp. 3626–3637, Jun. 2022, doi: 10.1109/TSMC.2021.3071546.

[152] T. Yang, N. Sun, H. Chen, and Y. Fang, “Neural Network-Based Adaptive Antiswing Control of an Underactuated Ship-Mounted Crane With Roll Motions and Input Dead Zones,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 901–914, Mar. 2020, doi: 10.1109/TNNLS.2019.2910580.

[153] K. Cao, Z. Zhang, Y. Li, W. Zheng, and M. Xie, “Ship fuel sulfur content prediction based on convolutional neural network and ultraviolet camera images,” *Environmental Pollution*, vol. 273, p. 116501, Mar. 2021, doi: 10.1016/j.envpol.2021.116501.

[154] Y. Han, G. Li, Q. Qin, S. Wang, and Y. Li, “Research on Machine Vision Detection Method of Ship Sulfur Emission Based on Convolutional Neural Network,” *J. Phys. Conf. Ser.*, vol. 2171, no. 1, p. 012071, Jan. 2022, doi: 10.1088/1742-6596/2171/1/012071.

[155] E. Bal Beşikçi, O. Arslan, O. Turan, and A. I. Ölçer, “An artificial neural network based decision support system for energy efficient ship operations,” *Comput. Oper. Res.*, vol. 66, no. January 2013, pp. 393–401, Feb. 2016, doi: 10.1016/j.cor.2015.04.004.

[156] J. Zheng *et al.*, “A voyage with minimal fuel consumption for cruise ships,” *J. Clean. Prod.*, vol. 215, no. 2019, pp. 144–153, Apr. 2019, doi: 10.1016/j.jclepro.2019.01.032.

[157] L. Zhao and G. Shi, “Maritime Anomaly Detection using Density-based Clustering and Recurrent Neural Network,” *Journal of Navigation*, vol. 72, no. 04, pp. 894–916, Jul. 2019, doi: 10.1017/S0373463319000031.

[158] W. Zhang, X. Feng, F. Goerlandt, and Q. Liu, “Towards a Convolutional Neural Network model for classifying regional ship collision risk levels for waterway risk analysis,” *Reliab. Eng. Syst. Saf.*, vol. 204, no. June 2019, p. 107127, Dec. 2020, doi: 10.1016/j.ress.2020.107127.

[159] S. Vukša, P. Vidan, M. Bukljaš, and S. Pavić, “Research on Ship Collision Probability Model Based on Monte Carlo Simulation and Bi-LSTM,” *J. Mar. Sci. Eng.*, vol. 10, no. 8, p. 1124, Aug. 2022, doi: 10.3390/jmse10081124.

[160] L. Hao, Y. Han, C. Shi, and Z. Pan, “Recurrent neural networks for nonparametric modeling of ship maneuvering motion,” *International Journal of Naval Architecture and Ocean Engineering*, vol. 14, p. 100436, 2022, doi: 10.1016/j.ijnaoe.2022.100436.

[161] Y. Li, R. W. Liu, Z. Liu, and J. Liu, “Similarity Grouping-Guided Neural Network Modeling for Maritime Time Series Prediction,” *IEEE Access*, vol. 7, pp. 72647–72659, 2019, doi: 10.1109/ACCESS.2019.2920436.

[162] T. Cepowski, “The prediction of ship added resistance at the preliminary design stage by the use of an artificial neural network,” *Ocean Engineering*, vol. 195, no. October 2019, p. 106657, Jan. 2020, doi: 10.1016/j.oceaneng.2019.106657.

[163] R. Vicen-Bueno, R. Carrasco-Álvarez, M. Rosa-Zurera, J. C. Nieto-Borge, and M. P. Jarabo-Amores, “Artificial Neural Network-Based Clutter Reduction Systems for Ship Size Estimation in Maritime Radars,” *EURASIP J. Adv. Signal Process.*, vol. 2010, no. 1, p. 380473, Dec. 2010, doi: 10.1155/2010/380473.

[164] M. Meler-Kapcia, S. Zieliński, and Z. Kowalski, “On application of some artificial intelligence methods in ship design,” *Polish Maritime Research*, pp. 14–20, 2005.

[165] A. Stateczny, “The neural method of sea bottom shape modelling for the spatial maritime information system,” in *Maritime Engineering and Ports II*, vol. 51, C. A. Brebbia, Ed., Wessex Institute of Technology, 2000, pp. 251–259. doi: 10.2495/PORTS000221.

[166] M. Jeon, Y. Noh, Y. Shin, O. K. Lim, I. Lee, and D. Cho, “Prediction of ship fuel consumption by using an artificial neural network,” *Journal of Mechanical Science and Technology*, vol. 32, no. 12, pp. 5785–5796, 2018, doi: 10.1007/s12206-018-1126-4.

[167] Y. Jiang, X.-R. Hou, X.-G. Wang, Z.-H. Wang, Z.-L. Yang, and Z.-J. Zou, “Identification modeling and prediction of ship maneuvering motion based on LSTM deep neural network,” *J. Mar. Sci. Technol.*, vol. 27, no. 1, pp. 125–137, Mar. 2022, doi: 10.1007/s00773-021-00819-9.

[168] S. Baressi Šegota, I. Lorencin, N. Anđelić, V. Mrzljak, and Z. Car, “Improvement of Marine Steam Turbine Conventional Exergy Analysis by Neural Network Application,” *J. Mar. Sci. Eng.*, vol. 8, no. 11, p. 884, Nov. 2020, doi: 10.3390/jmse8110884.

[169] C. W. Mohd Noor, R. Mamat, G. Najafi, M. H. Mat Yasin, C. K. Ihsan, and M. M. Noor, “Prediction of marine diesel engine performance by using artificial neural network model,” *Journal of Mechanical Engineering and Sciences*, vol. 10, no. 1, pp. 1917–1930, Jun. 2016, doi: 10.15282/jmes.10.1.2016.15.0183.

[170] Weifeng Shi, Jianmin Yang, and Tianhao Tang, “RBF NN based marine diesel engine generator modeling,” in *Proceedings of the 2005, American Control Conference, 2005.*, IEEE, 2005, pp. 2745–2749. doi: 10.1109/ACC.2005.1470384.

[171] M. Katalinić, P. Matić, N. Assani, and J. Parunov, “Encounter spectra computation of heave motion based on full-scale measurements using ANN,” in *Advances in the Analysis and Design of Marine Structures*, London: CRC Press, 2023, pp. 85–93. doi: 10.1201/9781003399759-10.

[172] K. N. Gyftakis, P. A. Panagiotou, and D. Spyrakis, “Detection of simultaneous mechanical faults in 6-kV pumping induction motors using combined MCSA and stray flux methods,” *IET Electr. Power Appl.*, vol. 15, no. 5, pp. 643–652, May 2021, doi: 10.1049/elp2.12054.

[173] I. Zamudio-Ramirez *et al.*, “Automatic diagnosis of electromechanical faults in induction motors based on the transient analysis of the stray flux via MUSIC methods,” *IEEE Trans. Ind. Appl.*, vol. 56, no. 4, pp. 1–1, Jul. 2020, doi: 10.1109/TIA.2020.2988002.

[174] A. Glowacz, “Acoustic based fault diagnosis of three-phase induction motor,” *Applied Acoustics*, vol. 137, pp. 82–89, Aug. 2018, doi: 10.1016/j.apacoust.2018.03.010.

[175] X. Liang, “Temperature Estimation and Vibration Monitoring for Induction Motors and the Potential Application in Electrical Submersible Motors,” *Canadian Journal of Electrical and Computer Engineering*, vol. 42, no. 3, pp. 148–162, Jun. 2019, doi: 10.1109/CJECE.2018.2875111.

[176] P. Krata, “The Impact of Sloshing Liquids on Ship Stability for Various Dimensions of Partly Filled Tanks,” *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, vol. 7, no. 4, pp. 481–489, Dec. 2013, doi: 10.12716/1001.07.04.02.

[177] R. K. Pearson and Ü. Kotta, “Nonlinear discrete-time models: state-space vs. I/O representations,” *J. Process Control*, vol. 14, no. 5, pp. 533–538, Aug. 2004, doi: 10.1016/j.jprocont.2003.09.007.

[178] R. E. Uhrig, “Introduction to artificial neural networks,” in *Proceedings of IECON ’95 - 21st Annual Conference on IEEE Industrial Electronics*, IEEE, 1995, pp. 33–37. doi: 10.1109/IECON.1995.483329.

[179] J. J. Hopfield, “Artificial neural networks,” *IEEE Circuits and Devices Magazine*, vol. 4, no. 5, pp. 3–10, Sep. 1988, doi: 10.1109/101.8118.

[180] J. Liu, T. Li, Z. Zhang, and J. Chen, “NARX Prediction-Based Parameters Online Tuning Method of Intelligent PID System,” *IEEE Access*, vol. 8, pp. 130922–130936, 2020, doi: 10.1109/ACCESS.2020.3007848.

[181] Q. Xiong, W.-J. Cai, and M.-J. He, “Equivalent transfer function method for PI/PID controller design of MIMO processes,” *J. Process Control*, vol. 17, no. 8, pp. 665–673, Sep. 2007, doi: 10.1016/j.jprocont.2007.01.004.

[182] M. Dulău, H. Grif, and S. Oltean, “ASPECTS OF ROBUSTNESS IN FLOW CONTROL PROCESSES,” in *The 5 th Edition of the Interdisciplinarity in Engineering International Conference*, “Petru Maior” University of Tîrgu Mureş, Romania, 2011.

[183] Y. Zhu, "Identification by the Least-Squares Method," in *Multivariable System Identification For Process Control*, Y. Zhu, Ed., Oxford: Pergamon, 2001, pp. 65–96. doi: https://doi.org/10.1016/B978-008043985-3/50006-5.

[184] T. Söderström and P. Stoica, "Instrumental variable methods for system identification," *Circuits Syst. Signal Process.*, vol. 21, no. 1, pp. 1–9, Jan. 2002, doi: 10.1007/BF01211647.

[185] K. A. Petsounis and S. D. Fassois, "Parametric time-domain methods for the identification of vibrating structures-a critical comparison and assessment," *Mech. Syst. Signal Process.*, vol. 15, no. 6, pp. 1031–1060, Feb. 2001, doi: 10.1006/mssp.2001.1424.

[186] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989, doi: 10.1007/BF02551274.

[187] L. Ljung, *System identification Theory for the User*. 1999.

[188] A. K. Tangirala, *Principles of System Identification*. CRC Press, 2018. doi: 10.1201/9781315222509.

[189] Z. Boussaada, O. Curea, A. Remaci, H. Camblong, and N. Mrabet Bellaaj, "A Nonlinear Autoregressive Exogenous (NARX) Neural Network Model for the Prediction of the Daily Direct Solar Radiation," *Energies (Basel).*, vol. 11, no. 3, p. 620, Mar. 2018, doi: 10.3390/en11030620.

[190] N. J. De Vos and T. H. M. Rientjes, "Hydrology and Earth System Sciences Constraints of artificial neural networks for rainfall-runoff modelling: trade-offs in hydrological state representation and model evaluation," 2005. [Online]. Available: www.copernicus.org/EGU/hess/hess/9/111/SRef-ID:1607-7938/hess/2005-9-111EuropeanGeosciencesUnion

[191] S. Yadav and S. Shukla, "Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification," in *Proceedings - 6th International Advanced Computing Conference, IACC 2016*, Institute of Electrical and Electronics Engineers Inc., Aug. 2016, pp. 78–83. doi: 10.1109/IACC.2016.25.

[192] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. New York, NY: Springer New York, 2013. doi: 10.1007/978-1-4614-6849-3.

[193] J. M. S. Ribeiro, M. F. Santos, M. J. Carmo, and M. F. Silva, "Comparison of PID controller tuning methods: analytical/classical techniques versus optimization algorithms," in *2017 18th International Carpathian Control Conference (ICCC)*, IEEE, May 2017, pp. 533–538. doi: 10.1109/CarpathianCC.2017.7970458.

[194] T. S. Schei, “Automatic tuning of PID controllers based on transfer function estimation,” *Automatica*, vol. 30, no. 12, pp. 1983–1989, Dec. 1994, doi: 10.1016/0005-1098(94)90060-4.

[195] R. P. Borase, D. K. Maghade, S. Y. Sondkar, and S. N. Pawar, “A review of PID control, tuning methods and applications,” *Int. J. Dyn. Control*, vol. 9, no. 2, pp. 818–827, Jun. 2021, doi: 10.1007/s40435-020-00665-4.

[196] M. W. Foley, R. H. Julien, and B. R. Copeland, “A Comparison of PID Controller Tuning Methods,” *Can. J. Chem. Eng.*, vol. 83, 2005.

[197] A. Marino and F. Neri, “PID Tuning with Neural Networks,” in *Intelligent Information and Database Systems*, Springer Verlag, 2019, pp. 476–487. doi: 10.1007/978-3-030-14799-0_41.

[198] N. H. Sabri, N. Hidayu, A. Rani, and M. Arbanah, “OPTIMAL PI TUNING RULES FOR LIQUID FLOW PROCESS CONTROL SYSTEMUSING COHEN- COON’S (C-C), ZIEGLER-NICHOLS’S (Z-N) AND TAKAHASHI’S TUNING METHOD,” *Malaysian Journal of Industrial Technology*, vol. 4, no. 1, pp. 1–6, 2020, [Online]. Available: https://www.researchgate.net/publication/348663625

[199] J. G. Ziegler and N. B. Nichols, “Optimum Settings for Automatic Controllers,” *Transactions of the ASME*, pp. 759–768, 1942, [Online]. Available: http://asme.org/terms

[200] D. Pavković, S. Polak, and D. Zorc, “PID controller auto-tuning based on process step response and damping optimum criterion,” in *ISA Transactions*, ISA - Instrumentation, Systems, and Automation Society, 2014, pp. 85–96. doi: 10.1016/j.isatra.2013.08.011.

[201] A. A. Ishak and M. A. Hussain, “Reformulation of the tangent method for PID controller tuning,” in *2000 TENCON Proceedings. Intelligent Systems and Technologies for the New Millennium (Cat. No.00CH37119)*, IEEE, pp. 484–488. doi: 10.1109/TENCON.2000.892314.

[202] N. Hambali, M. N. K. Zaki, and A. A. Ishak, “Reformulated tangent method of various PID controller tuning for air pressure control,” in *2012 IEEE International Conference on Control System, Computing and Engineering*, IEEE, Nov. 2012, pp. 17–22. doi: 10.1109/ICCSCE.2012.6487108.

[203] G. H. Cohen and G. A. Coon, “Theoretical Consideration of Retarded Control,” *J. Fluids Eng.*, vol. 75, no. 5, pp. 827–834, Jul. 1953, doi: 10.1115/1.4015451.

[204] K. L. Chien, J. A. Hrones, and J. B. Reswick, “On the Automatic Control of Generalized Passive Systems,” *J. Fluids Eng.*, vol. 74, no. 2, pp. 175–183, Feb. 1952, doi: 10.1115/1.4015724.

[205] Boxhammer J, “Experiment Instructions RT 580 Control systems and fault finding,” Barsbüttel, Jun. 2019.

LIST OF FIGURES AND DIAGRAMS

LIST OF TABLES

LIST OF ABBREVIATIONS

AE	Absolute Error
AIoU	Average Intersection over Union
AIS	Automatic Identification System
AMQP	Advanced Message Queuing Protocol
ANN	Artificial Neural Network
AP	Average Precision
AVR	Automatic Voltage Regulator
BEM	Boundary Element Method
BMU	Best Matching Unit
BTNN	Bayesian-Transformer Neural Network
CAN	Controller Area Network
CBM	Condition-Based Maintenance
CDMA	Code-Division Multiple Access
CE	Cross-Entropy
CIP	Common Industrial Protocol
CMV	Cycle Mean Value
CNN	Convolutional Neural Network
CoAP	Constrained Application Protocol
CPS	Cyber-Physical System
CSMA/CD	Carrier-Sense Multiple Access with Collision Detection
CSMA/NBA	Carrier-Sense Multiple Access with Non-Blocking Arbitration
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DDM	Data-Driven Model
DeviceNet	Device Network Protocol
DG	Diesel Generator
DP	Dynamic Positioning
DT	Digital Twin
DTW	Dynamic Time Warping
ELU	Exponential Linear Unit
EtherCAT	Ethernet for Control Automation Technology
FEM	Finite Element Method

FFANN	Feedforward Artificial Neural Network
FFT	Fast Fourier Transform
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
FN	False Negative
FOMCON	Fractional-Order Modelling and Control
FOPTD	First-Order Plus Time Delay
FP	False Positive
FPS	Frames per Second
GBM	Grey-Box Model
GHG	Greenhouse Gas
GoF	Goodness of Fit
GRNN	Generalised Regression Neural Network
HIL	Hardware-in-the-Loop
IoT	Internet of Things
IoU	Intersection over Union
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
LPV	Linear Parameter-Varying
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
mAP	Mean Average Precision
MAPE	Mean Absolute Percentage Error
MBD	Model-Based Design
MBSE	Model-Based Systems Engineering
MSE	Mean Squared Error
MTConnect	Manufacturing Technology Connect Standard
MQTT	Message Queuing Telemetry Transport
NARX	Nonlinear Autoregressive Model with Exogenous Inputs
NMEA	National Marine Electronics Association
NRMSE	Normalized Root Mean Squared Error
NTP	Network Time Protocol
OPC	Open Platform Communications
OPC UA	Open Platform Communications Unified Architecture

OSP	Open Simulation Platform
OSI	Open Systems Interconnection
PB	Percent Bias
PI	Persistence Index
PID	Proportional-Integral-Derivative
PROFIBUS	Process Field Bus
PROFINET	Process Field Network
PR	Precision-Recall
PSO	Particle Swarm Optimisation
PTP	Precision Time Protocol
RBF	Radial Basis Function
RBF-ANN	Radial Basis Function Artificial Neural Network
ReLU	Rectified Linear Unit
RE	Relative Error
RMS	Root Mean Square
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RRR	Roll Reduction Rate
RS	Recommended Standard
SCR	Signal-to-Clutter Ratio
SDT	Ship’s Digital Twin
SGD	Stochastic Gradient Descent
SOAP	Simple Object Access Protocol
SOM	Self-Organising Map
SVM	Support Vector Machine
TCP/IP	Transmission Control Protocol / Internet Protocol
TF	Transfer Function
TN	True Negative
TP	True Positive
UDP	User Datagram Protocol

LIST OF SYMBOLS

a_i	denominator polynomial coefficient
α	slope coefficient of activation function
b	bias vector
b_i	numerator polynomial coefficient
β	output weight matrix, regularization parameters
β_1	exponential decay rate for first moment estimate (Adam algorithm)
β_2	exponential decay rate for second moment estimate (Adam algorithm)
c_1	acceleration coefficient (PSO algorithm)
c_2	acceleration coefficient (PSO algorithm)
$D(i,j)$	cumulative alignment distance in Dynamic Time Warping
d_i	Euclidean distance to neuron *i* (SOM)
$E(t)$	network error at time step *t*
$E(x_t)$	error function for training sample x_t
e	prediction error vector
ε	positive constant, neighbourhood radius
$f(\cdot)$	nonlinear mapping or activation function
f_i	fitness value of solution *i* (GA)
$G(s)$	transfer function
g	global best position (PSO)
$\mathrm{g_h}$	gradient at iteration *t*
H	hidden-layer output matrix
H^+	Moore-Penrose pseudoinverse of matrix *H*
I	identity matrix

i	order of numerator polynomial
j	order of denominator polynomial
J	Jacobian matrix
K	number of classes in Softmax function
k	discrete-time index
K_d	derivative gain
K_i	integral gain
K_p	proportional gain
$L(q)$	evidence lower bound (ELBO)
m	number of output delays (autoregressive order)
$\hat{m}_t$	bias-corrected first moment estimate
μ	damping factor (Levenberg–Marquardt)
n	number of input delays (exogenous input order)
n_h	number of hidden neurons
n_x	number of input neurons
$N_\varepsilon(p)$	ε-neighbourhood of point p (DBSCAN)
$p(w)$	prior distribution of weights
p_i	best personal position (PSO)
P_i	selection probability (GA)
$P_i(s)$	numerator polynomial
$q(z)$	variational distribution
q_t	flow at time t
$\hat{q}_i$	predicted flow sample
$\bar{q}$	mean value of measured flow
$Q_j(s)$	denominator polynomial

r_1, r_2	uniformly distributed random numbers
s	complex frequency variable
T	target-output matrix
$u(k)$	discrete-time input
$u(t)$	continuous-time input
$v_i(t)$	particle velocity
v_t	second moment estimate (Adam)
$\hat{v}_t$	bias-corrected second moment estimate
W_x	input weight matrix
W_γ	recurrent weight matrix
w	neural network weight vector
$w^1{}_{0h}$	bias weight of hidden neuron h
$w^2{}_{0j}$	bias weight of output neuron j
$w^2{}_{hj}$	weight between hidden neuron h and output neuron j
$w^1{}_{ih}$	weight between input neuron i and hidden neuron h
w_j	weight vector of neuron j
x	observed data
x_i	i-th input variable
$x_i(t)$	i-th input variable at time t
$y(k)$	discrete-time output
$y(t)$	continuous-time output
y_j	j-th output variable
$y_j(t)$	j-th output variable at time t
$y_j(t-1)$	j-th output variable at previous time step
Y	network-output matrix

Z	instrument matrix
z^{-1}	unit time delay operator
θ	threshold in QuickBundle algorithm
θ_t	parameter vector in Adam algorithm
θ_{nom}	nominal parameter vector in Adam algorithm
$\hat{\theta}_{IV}$	instrumental-variable estimate of parameter vector
Δw	weight update vector
$\Delta w_{ij}(t)$	weight change at time step t
η	learning rate
ω	inertia weight (PSO)
Φ	regressor matrix
Φ_h	radial basis activation function of hidden neuron h
σ	sigmoid activation function

BIOGRAPHY

Nur Assani was born in Split, Croatia, in 1994. He received the B.S. and M.S. degrees in marine electrical engineering from the Faculty of Maritime Studies, University of Split in 2018 and 2020, respectively. Prior to his academic career, he worked as an electrician, gaining practical experience in electrical installations and maintenance.

He is currently employed as a Teaching and Research assistant at the Faculty of Maritime Studies, University of Split. Earlier, he worked as a student demonstrator in mathematics, marine electrical machinery, and electronic circuits at the same institution.

His scientific interests include ship's digital twins, data-driven and mathematical modelling of ship systems, artificial neural networks, process identification, and ship automation. He is a student member of IEEE and member of several IEEE technical societies. For his academic excellence, he has received the Rector's Award for Excellence, two summa cum laude awards and Dean's Award for best scientific paper in doctoral studies.

He authored and co-authored fourteen scientific papers so far, as listed in reverse chronological order below:

1. Assani, N.; Čalić, A.; Matić, P.; Katalinić, M. *Multi-branch Artificial Neural Network-based decomposition of Ship's Added Resistance using on*-*board measured data.* IEEE Access, 14 (2026), 29349–29360. doi:10.1109/access.2026.3666677.

2. Assani, N.; Matić, P. *Evaluating the ANN Model Performance for PID Controller Tuning in Flow Process Control: A Comparative Study.* IEEE Access, 13 (2025), 88499–88508. doi:10.1109/access.2025.3571222.

3. Martinić-Cezar, S.; Jurić, Z.; Assani, N.; Račić, N. *Controlling Engine Load Distribution in LNG Ship Propulsion Systems to Optimize Gas Emissions and Fuel Consumption.* Energies, 18 (2025), 485–506. doi:10.3390/en18030485.

4. Kaštelan, N.; Vidan, P.; Assani, N.; Miličević, M. *Digital Horizon: Assessing Current Status of Digitalization in Maritime Industry.* Transactions on Maritime Science, 13 (2024), 21–26. doi:10.7225/toms.v13.n01.w13.

5. Assani, N.; Matić, P.; Kezić, D.; Pleić, N. *Modeling Fluid Flow in Ship Systems for Controller Tuning Using an Artificial Neural Network.* Journal of Marine Science and Engineering, 12 (2024), 1318–1329. doi:10.3390/jmse12081318.

6. Martinić-Cezar, S.; Jurić, Z.; Assani, N.; Lalić, B. *Optimization of Fuel Consumption by Controlling the Load Distribution between Engines in an LNG Ship Electric Propulsion Plant.* Energies, 17 (2024), 3718–3737. doi:10.3390/en17153718.

7. Assani, N.; Matić, P.; Kezić, D. *Flow control process identification using Matlab's System Identification Toolbox.* 8th International Conference on Control, Decision and Information Technologies (CoDIT), 2022, 1228–1232. doi:10.1109/codit55151.2022.9803906.

8. Kaštelan, N.; Vujović, I.; Krčum, M.; Assani, N. *Switchgear Digitalization—Research Path, Status, and Future Work.* Sensors, 22 (2022), 7922. doi:10.3390/s22207922.

9. Assani, N.; Matić, P.; Katalinić, M. *Ship's Digital Twin—A Review of Modelling Challenges and Applications.* Applied Sciences, 12 (2022), 6039. doi:10.3390/app12126039.

10. Assani, N.; Matić, P.; Kaštelan, N.; Čavka, I. R. *A review of artificial neural networks applications in maritime industry.* IEEE Access, XI (2023), 139823–139848. doi:10.1109/ACCESS.2023.3341690.

11. Assani, N.; Matić, P.; Kezić, D. *Evaluating the Performance of the Well-Known Controller Tuning Methods for the Flow Control Using the Process Model.* 9th International Conference on Control, Decision and Information Technologies (CoDIT), IEEE, 2023.

12. Katalinić, M.; Matić, P.; Assani, N.; Parunov, J. *Encounter spectra computation of heave motion based on full-scale measurements using ANN.* Proceedings of the 9th International Conference on Marine Structures (MARSTRUCT 2023), CRC Press, 2023, 85–93. doi:10.1201/9781003399759.

13. Assani, N.; Pavić, I.; Vukša, S.; Laušić, M. *Analysis of the Nomoto ship model response to course changes using PID controller in Matlab/Simulink.* ICTS 2020: Maritime, Transport and Logistics Science – Conference Proceedings, 14–18.

14. Andrić, D.; Assani, N.; Cvitković, B.; Glavinović, R. *Challenges of Marine Education with Implementation of Modern Technologies.* Book of Proceedings, 8th International Maritime Science Conference, 2019, 103–110.