Received: 29 June 2023 | Revised: 24 April 2024 | Accepted: 30 April 2024

DOI: 10.1002/acs.3835

RESEARCH ARTICLE

WILEY

Data-driven forecasting of ship motions in waves using machine learning and dynamic mode decomposition

Matteo Diez[1] | **Mauro Gaggero**[2] | **Andrea Serani**[1]

[1]National Research Council of Italy, INM, Rome, Italy

[2]National Research Council of Italy, INM, Genoa, Italy

Correspondence
Mauro Gaggero, National Research Council of Italy, INM, Via De Marini 6, I-16149 Genoa, Italy.
Email: mauro.gaggero@cnr.it

Funding information
US Office of Naval Research, Grant/Award Number: N62909-21-1-2042; Italian Ministry of University and Research, Grant/Award Number: CN00000023 - CUP B43C22000440001

Summary

Data-driven forecasting of ship motions in waves is investigated through feedforward and recurrent neural networks as well as dynamic mode decomposition. The goal is to predict future ship motion variables based on past data collected on the field, using equation-free approaches. Numerical results in two case studies involving the course-keeping of a naval destroyer in a high sea state using simulation data at model scale are presented. The proposed methods reveal successful in predicting ship motions both in short-term and medium-term perspectives with accuracy and reduced computational effort, thus enabling further advances in the identification, control, and optimization of ships operating in waves.

KEYWORDS

dynamic mode decomposition, forecasting, machine learning, neural networks, ship motions in waves

1 | INTRODUCTION

In severe sea and weather conditions, the availability of forecasting approaches for ship performance in waves, including motions, power requirements, loads, and structural responses, may be critical to ensure safety of crew, payload, and structures. In this context, digital-twin platforms able to learn from data, infer, and act are essential to safeguard resources.[1] Nowadays, data has emerged as a crucial asset for researchers across various domains, including system modeling and identification[2] as well as control theory and optimization.[3] In general, having at disposal a vast amount of data, some of which is publicly available, combined with advancements in machine learning, has led to the growth of data-driven modeling as a powerful approach to understand complex systems, make predictions, and take decisions.[4] More specifically, data-driven modeling consists of extracting insights and patterns from data to develop mathematical models that can accurately represent and predict the behavior of a given physical phenomenon. Unlike classical modeling approaches that typically rely on prime principles or theoretical assumptions and simplifications, data-driven modeling can capture hidden relationships and dynamics directly from available data. When the governing equations of the system under investigation are either unknown or not used in the learning process, the approach is also referred to as equation-[5] or model-[6] free. The benefits of data-driven modeling are numerous: by leveraging data, it is possible to optimize processes, detect anomalies, take decisions, and develop predictive models to forecast future outcomes. A wide range of techniques are available for data-driven modeling, such as statistical modeling, reduced-order modeling, and machine learning.[4] Statistical modeling focuses on inferring relationships between variables based on observed data, while reduced-order modeling aims at developing simplified versions of complex models, yet preserving their essential features

[Correction added on 11 September 2024, after first online publication: The copyright line was changed.]

and behaviors. The goal is to reduce the computational complexity of the model while maintaining an acceptable level of accuracy. Lastly, machine learning algorithms aim to automatically learn patterns and make predictions from data. Among the different machine learning approaches, neural networks are widely employed to extract hierarchical representations from data, enabling complex pattern recognition and prediction tasks.[7] The various methods can also be combined with each other. For instance, the work of Fu et al.[8] presents a method for integrating neural networks with physical information to enhance the extrapolation capabilities of reduced-order models using a small number of training data. Moreover, another work by Fu et al.[9] proposes a novel reduced-order model based on an autoencoder neural network and shows how it provides more accurate results than traditional linear proper orthogonal decomposition for problems with high nonlinearity.

The contribution of this article lies in the investigation of data-driven modeling to prove the feasibility of performing forecasts of the motions of ships operating in waves starting from data measured on the field. Toward this end, we present three different approaches in the context of time series prediction to forecast the future evolution over time of the variables describing the motion of a ship, such as roll, pitch, yaw and so on based on the recent past history. First, we focus on the use of machine learning techniques, and in particular on feed-forward neural networks (FFNN)[10] and recurrent neural networks (RNN),[11] with their methodological extensions such as long short-term memory networks (LSTM),[12] bidirectional LSTM (BiLSTM),[13] and gated recurrent units networks (GRU).[14] Then, we consider dynamic mode decomposition (DMD)[15] and its augmented formulation (ADMD)[1] as data-driven, reduced-order modeling approaches to time-series forecasting. DMD and ADMD represent an appealing alternative to neural-network-based methods since they do not require time-consuming training processes, and therefore they are suitable for real-time (or nearly real-time) learning in digital twin platforms.

Data-driven approaches to forecast the response of ships in waves have been developed and discussed in the literature, including machine learning via recurrent-type neural networks[16,17] and model-order reduction via DMD.[1,18–20] As previously pointed out, machine learning and reduced order modeling approaches are strictly connected with each other. For instance, the work of Wang et al.[21] is one of the first applications in fluid dynamics of LSTM to reduced-order modeling. In general, data-driven modeling studies for ships operating in waves,[16,22,23] especially deep-learning[24–27] and hybrid[28–31] approaches, are gaining prominence due to their capacity to capture intricate temporal and spatial relationships within ship motion data. However, such approaches are still limited if compared to other areas of application.

To assess the effectiveness of the proposed approaches, we focus on two challenging case studies regarding the course keeping of a naval destroyer (at model scale) in stern-quartering irregular waves at sea state 7 and at a nominal Froude number equal to 0.33. Such a problem is worth investigating due to the high sea state, with a significant wave height of 7 m (at full scale), the wave heading with respect to the ship course (300°, i.e., stern quartering), and the nominal peak period that is close to the rolling natural frequency, thus inducing roll motion resonance and possible capsizing of the vessel. The two case studies differ on the time horizon considered for the predictions. In more detail, the goal of the first case is to predict ship motions in the short period, that is, one future encounter wave is considered, while, in the second case, the horizon is longer since we aim at predicting the ship motion over three future encounter waves. Both case studies are significant since they correspond to key different situations, with the horizon length playing a crucial role in the prediction accuracy and required computational burden. As an example, the shorter prediction horizon could be adopted to develop feedback control actions to be used in real time as a reaction to unexpected events, while longer horizons could be used to develop suitable model-predictive-control-based approaches to pursue optimization of the vessel trajectory or any other desired behavior.

Successful simulation results using data taken from unsteady Reynolds-averaged Navier Stokes (URANS) computations from Serani et al.[32] are reported in this article, thus showcasing the good prediction capability of both neural networks and ADMD approaches. Concerning computational requirements, the obtained results are very satisfactory as well. In fact, after a training phase for both FFNN and RNN lasting about a few minutes on standard hardware, the prediction of the ship motion variables entails a negligible computational time in both the considered case studies. The same occurs when prediction is performed by using ADMD (in this case, no training is required owing to the different nature of ADMD). Hence, the adoption of the forecasting approaches proposed in this article may pave the way to further research on the development of control and optimization techniques for vessels operating in waves, for which the current requirements from the computational viewpoint represent a limitation.

The remainder of the article is organized as follows. Section 2 reports the problem statement. Section 3 summarizes the basics of FFNN and RNN for time series prediction, while Section 4 reports the use of ADMD for the same purpose. Section 5 showcases the used metrics for performance evaluation, while Section 6 discusses the obtained numerical results. Lastly, Section 7 draws conclusions and outlines future works.

2 | PROBLEM STATEMENT AND MODELING OBJECTIVES

In this section, we formulate the problem we deal with throughout the article, that is, the construction an approximate model to forecast the motions of ships in waves. Toward this end, we focus on three different kinds of models constructed starting from data collected on the field: FFNN, RNN, and ADMD. While constructing the model, we have to pursue the following objectives: (i) the resulting model should be able to capture the essential dynamics of the motions of vessels; (ii) it has to be characterized by a reduced computational effort to be used online during ship operations for the purpose of control and optimization.

Machine learning models based on neural networks and reduced-order approaches, such as ADMD, are different techniques that share the objective of simplifying the dynamics of complex systems. In particular, several connections can be identified between them, as described in the following.

1. *Dimensionality reduction*: both machine learning models and reduced-order ones may perform dimensionality reduction. The former can learn hierarchical representations that effectively capture the underlying structure of the data, whereas the latter extracts essential features from high-dimensional data.
2. *Feature extraction*: in both methods, feature extraction plays a crucial role. Neural networks learn representations of the data through layers of interconnected neurons, whereas reduced-order modeling identifies dominant modes, or patterns, in the data.
3. *Nonlinear mapping*: models based on neural networks are typically able to learn nonlinear mappings between input and output data, while reduced-order models often rely on linear methods for representation. This enables neural network models to capture complex relationships in the data that may be missed by linear techniques.
4. *Hybrid approaches*: there is a growing trend towards combining machine learning and reduced order models into hybrid approaches. For example, the latter can be used to reduce the dimensionality of input data before feeding it into a neural network for further processing, or neural networks can be used to enhance the accuracy of reduced-order predictions by learning correction terms or refining reduced representations.
5. *Equation/model-free learners*: all the methods discussed in this work (FFNN, RNN, and ADMD) do not require the knowledge of the equations governing the system, and therefore may be referred to as equation-free.[5] Albeit ADMD is not a machine learning method in the strict sense, from an artificial intelligence perspective FFNN, RNN, and ADMD are all model-free learners.[6]

In this article, we focus on a discrete-time setting, where time is discretized into several steps, with a constant sampling time Δt. Hence, we consider the time instants $t = 0, \Delta t, 2\Delta t, \ldots$. With a little abuse of notation, we indicate the discrete-time instants through integers by dropping Δt, that is $t = 0, 1, 2, \ldots$. The goal is to approximate, at each time instant t, the functional relationship between an input sequence of recent measures collected on the field and an output sequence of the same variables projected in the future. Toward this end, an approximating model based on FFNN, RNN, or ADMD in our case, is used to map an input, measured sequence into an output, predicted one. Figures 1 and 2 sketch

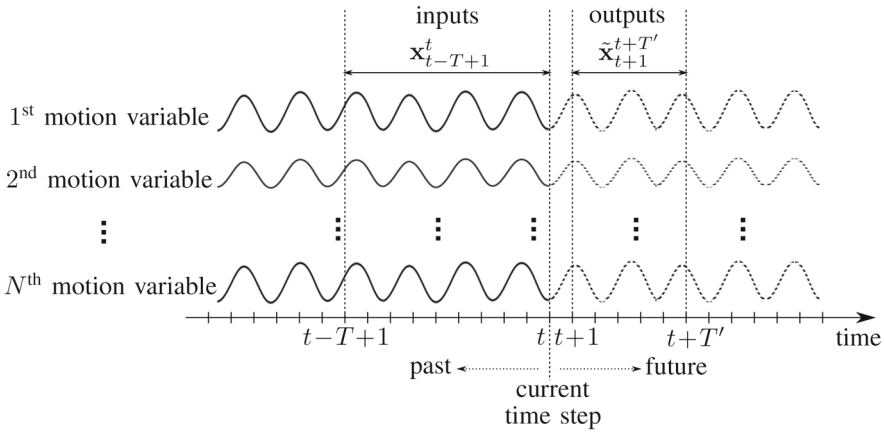

FIGURE 1 Sketch of the modeling approach considered in this article for the prediction of N ship motion variables.

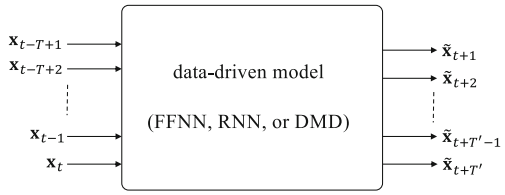

FIGURE 2 Sketch of the considered data-driven modeling approach approximating the mapping between an input, measured sequence to an output, predicted one. Specific versions of the diagram are reported later on for machine learning methods and DMD for time series forecasting.

such an idea. In more detail, let $\mathbf{x}_\tau \in \mathbb{R}^N$ be a vector quantity that is significant for the problem at hand at given time instants $\tau = t - T + 1, \ldots, t$, where t is the current time step (without loss of generality, in this article we assume that all the vectors are column vectors). In other words, we consider a window of past values of length $T \geq 1$ and construct an input sequence $\mathbf{x}_{t-T+1}, \ldots, \mathbf{x}_t$. In the case of the prediction of the behavior of ships in waves considered in this article, the vector $\mathbf{x}_\tau$ collects the variables describing the motions of the vessel at time τ: the ship six degrees of freedom, that is, surge, sway, heave, roll, pitch, and yaw along with the rudder angle (additional details on the types of used variables are reported in Section 6). The goal is to generate estimates $\tilde{\mathbf{x}}_\tau \in \mathbb{R}^N$ in a future time window of length $T' \geq 1$ starting from time t, that is, for $\tau = t + 1, \ldots, t + T'$, based on the observations of the same variables from time $t - T + 1$ to time t. The estimates $\tilde{\mathbf{x}}_{t+1}, \ldots, \tilde{\mathbf{x}}_{t+T'}$ describing the ship motions in the near future constitute the output sequence. For the sake of compactness, let us collect all the input variables $\mathbf{x}_\tau$, $\tau = t - T + 1, \ldots, t$, for a given time instant t in the vector $\mathbf{x}_{t-T+1}^{t} := \mathrm{col}(\mathbf{x}_{t-T+1}, \ldots, \mathbf{x}_t) \in \mathbb{R}^{N \times T}$, where, given two vectors $\mathbf{a} \in \mathbb{R}^p$ and $\mathbf{b} \in \mathbb{R}^q$, the notation $\mathrm{col}(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^{p+q}$ denotes the column-wise juxtaposition of the two vectors $\mathbf{a}$ and $\mathbf{b}$. Similarly, we collect the output estimates $\tilde{\mathbf{x}}_\tau$, $\tau = t + 1, \ldots, t + T'$ in the vector $\tilde{\mathbf{x}}_{t+1}^{t+T'} := \mathrm{col}(\tilde{\mathbf{x}}_{t+1}, \ldots, \tilde{\mathbf{x}}_{t+T'}) \in \mathbb{R}^{N \times T'}$. [Correction added on 24 May 2024, after first online publication: the term '$\mathfrak{R}$' has been changed to '$\mathbb{R}$' in the preceding paragraph.]

3 | MACHINE LEARNING APPROACHES

In this section, we review the basic concepts of the two types of machine learning approaches used throughout the article, that is FFNN (Section 3.1) and RNN (Section 3.2) for the purpose of time series prediction of the motion variables of vessels operating in waves based on measurements on the recent past. Figure 3 sketches the considered modeling approach for both types of methods.

3.1 | Feedforward neural networks

FFNN, also known as multilayer perceptrons, represent a fundamental and widely-used type of artificial neural network and constitute a basic block of several machine learning applications in different fields. They consist of interconnected layers of artificial neurons, also called nodes or computational units, arranged in a sequential manner, where information flows from the input layer to the output layer through one or more hidden layers. This unidirectional flow of data motivates the adjective "feedforward." The inputs are "transformed" through the hidden layers to generate the outputs. Examples of computational units in the hidden layer of such networks are radial basis functions, kernel units, and sigmoidal functions.

We focus on one-hidden-layer networks, which are particular kinds of networks with one input layer, one output layer, and only one layer in the middle, called "hidden." This kind of networks are also known as shallow networks, in contrast to deep ones where several hidden layers are present. Even if shallow networks are characterized by some limitations and deep ones are superior in certain tasks,[33] one-hidden-layer networks are still employed in applications,[34–39] and therefore their use for the purpose of predicting the motions of ships operating in waves is worth investigating. A number of theoretical results are available for one-hidden-layer FFNN. An important point is the so-called universal approximation property,[40] that is, such networks are able to approximate any well-behaved unknown function generating a set of data with arbitrary accuracy, provided a suitable number of computational units in the hidden layer is selected. Furthermore, under suitable regularity hypotheses on the unknown mapping generating the data, they guarantee

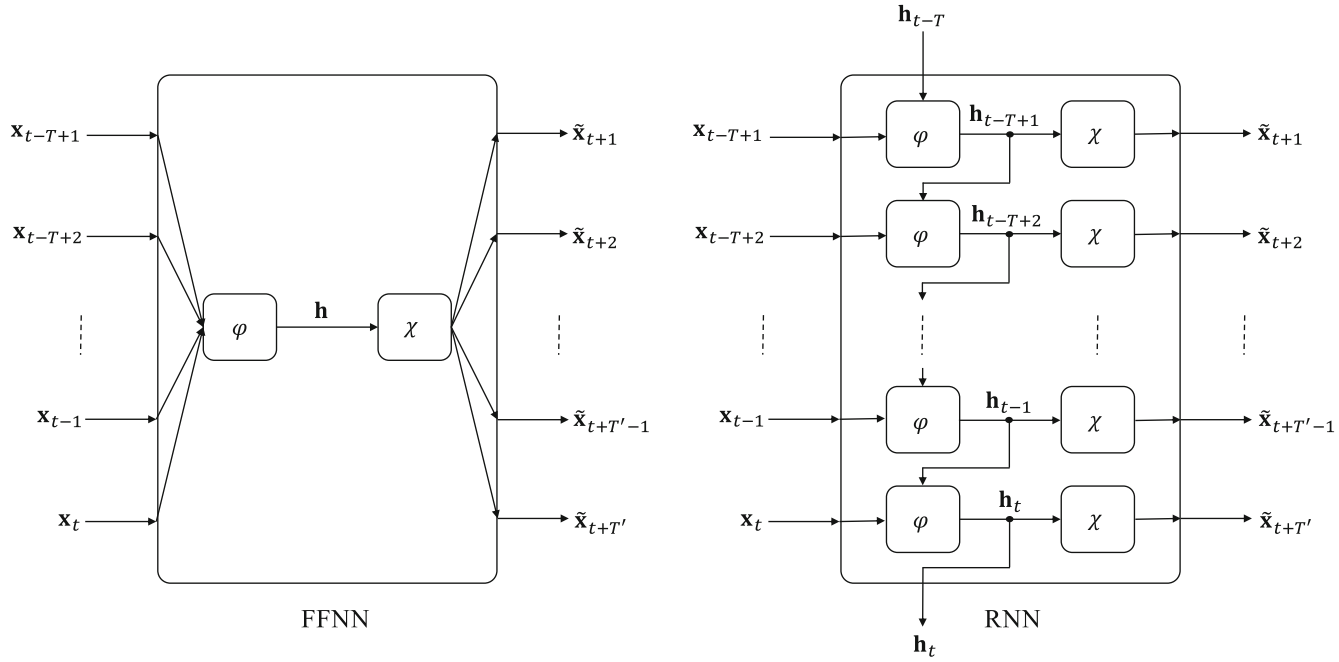

FIGURE 3 Sketch of the considered machine learning models: FFNN (left) and RNN (right).

uniform approximations characterized by upper bounds with a number of parameters growing at most polynomially with the dimension of the input of the function to be approximated.[40–42]

Given the input data $\mathbf{x}_{t-T+1}^{t}$ at a generic time step t, the output $\tilde{\mathbf{x}}_{t+1}^{t+T'}$ generated by a FFNN is computed as follows:

$$\mathbf{h} = \varphi\left(W_h\, \mathbf{x}_{t-T+1}^{t} + \mathbf{b}_h\right), \tag{1}$$

$$\tilde{\mathbf{x}}_{t+1}^{t+T'} = \chi(W_{\tilde{x}}\, \mathbf{h} + \mathbf{b}_{\tilde{x}}), \tag{2}$$

where $\mathbf{h} \in \mathbb{R}^{\nu}$ is the output of the hidden layer, $W_h \in \mathbb{R}^{\nu \times N \cdot T}$ is the weight matrix connecting the input layer to the hidden layer, $\mathbf{b}_h \in \mathbb{R}^{\nu}$ is the bias vector for the hidden layer, φ is the activation function applied element-wise, $W_{\tilde{x}} \in \mathbb{R}^{N \cdot T' \times \nu}$ is the weight matrix connecting the hidden layer to the output layer, $\mathbf{b}_{\tilde{x}} \in \mathbb{R}^{N \cdot T'}$ is the bias vector for the output layer, χ is the output function applied element-wise and ν is the number of computational units in the hidden layer.

As regards the form of the activation function φ, we focus on sigmoidal functions, as this is quite a common choice in the field of FFNN. Moreover, we consider a linear output function χ, as this is again quite a diffused practice. With these choices, (1) and (2) can be re-written by means of a parametrized mapping γ that generates the estimate $\tilde{\mathbf{x}}_{t+1}^{t+T'}$ starting from the input $\mathbf{x}_{t-T+1}^{t}$, as follows:

$$\tilde{\mathbf{x}}_{t+1}^{t+T'} = \gamma\left(\mathbf{x}_{t-T+1}^{t}, \mathbf{w}\right),$$

where

$$\gamma(\mathbf{x}_{t-T+1}^{t}, \mathbf{w}) = \sum_{i=1}^{\nu} \mathbf{c}_i\, \mathrm{sigm}\left(\sum_{j=1}^{N \cdot T} \mathbf{a}_{ij} \mathbf{x}_{t-T+1,j}^{t} + \mathbf{b}_i\right) + \mathbf{c}_0. \tag{3}$$

In (3), sigm(·) denotes a sigmoidal function, and $\mathbf{a}_{ij}$ $\mathbf{c}_i$, $\mathbf{b}_i$, and $\mathbf{c}_0$ are the parameters to be optimized (corresponding to W_h, $W_{\tilde{x}}$, $\mathbf{b}_h$, and $\mathbf{b}_{\tilde{x}}$, respectively, in (1) and (2)), which are collected in the vector $\mathbf{w} \in \mathbb{R}^{p}$.

The parameter vector $\mathbf{w}$ has to be properly adjusted through a training procedure, for which efficient algorithms based on nonlinear programming are available in the literature. Famous examples are the Levenberg–Marquardt and backpropagation methods. The purpose of the training is to adapt the parameter vector $\mathbf{w}$ to the available data, that is, to make the estimate $\tilde{\mathbf{x}}_{t+1}^{t+T'}$ as close as possible to the actual value of the target variables $\mathbf{x}_{t+1}^{t+T'}$ at the same time steps, which

are assumed to be available for the purpose of training. The goal is to choose the best parameter vector $\mathbf{w}^\star$ as the one minimizing an error function according to a mean squared error criterion, that is,

$$\mathbf{w}^\star := \arg\min_{\mathbf{w}\in\Re^p}\left(\frac{1}{M}\sum_{j=1}^{M}\left\|\mathbf{x}_{t+1}^{t+T',(j)} - \gamma\left(\mathbf{x}_{t-T+1}^{t,(j)}, \mathbf{w}\right)\right\|^2\right), \tag{4}$$

where M is the number of example realizations of the variables $\mathbf{x}_{t-T+1}^{t}$ and $\mathbf{x}_{t+1}^{t+T'}$, for which $\mathbf{x}_{t-T+1}^{t,(j)}$ and $\mathbf{x}_{t+1}^{t+T',(j)}$ represent the jth realization, respectively. In the neural network parlance, the vectors $\mathbf{x}_{t-T+1}^{t,(j)}, j = 1, \ldots, M$, are called patterns, while the vectors $\mathbf{x}_{t+1}^{t+T',(j)}, j = 1, \ldots, M$, are the so-called targets.

3.2 | Recurrent neural networks

Classical neural networks, such as FFNN described in Section 3.1, treat each observation independently, that is, they assume that the available data points are independent and identically distributed. However, in many applications such as time series forecasting, a strong correlation or dependency between current and past values exists. RNN have been developed to capture these dependencies and improve prediction accuracy in sequential data analysis. RNN are a class of artificial neural networks using connections between nodes in a directed graph along a temporal sequence, enabling them to exhibit a dynamic behavior over time. Derived from FFNN, RNN have the ability to process input sequences of different lengths by leveraging their internal state or memory.

The considered setting is again the sequence-to-sequence modeling approach described in Section 2 and in Figure 1, where the input data $\mathbf{x}_{t-T+1}^{t}$, given by the measurements collected in a window of length T starting at time t and projected toward the past up to $t - T + 1$, are used to generate the output $\tilde{\mathbf{x}}_{t+1}^{t+T'}$ composed of predictions of the ship motion variables from time $t + 1$ to time $t + T'$. The main difference with respect to FFNN lies in the fact that, in an RNN, the information is passed through time steps likewise in a discrete-time dynamical system. RNN implements a "memory mechanism" through the definition of a state vector, sometimes referred to as "hidden state," that tracks the temporal evolution of the inputs. In more detail, let $\mathbf{h}_t \in \mathbb{R}^\nu$ represent the hidden state of the network at a given time instant t. The state at time step t is determined as a function of the input data $\mathbf{x}_t$ at the same time step and of the previous state $\mathbf{h}_{t-1}$. The output $\tilde{\mathbf{x}}_{t+1}$ at time $t + 1$ is then computed as a function of the current state. More formally, the output vector $\tilde{\mathbf{x}}_{t+1}^{t+T'}$ of an RNN is computed by applying recursively the following equations:

$$\mathbf{h}_t = \varphi(W_x\,\mathbf{x}_t + U_h\,\mathbf{h}_{t-1} + \mathbf{b}_h), \tag{5}$$

$$\tilde{\mathbf{x}}_{t+1} = \chi(W_h\,\mathbf{h}_t + \mathbf{b}_{\tilde{x}}), \tag{6}$$

where $W_x \in \mathbb{R}^{\nu\times N}$ is the weight matrix for the input, $U_h \in \mathbb{R}^{\nu\times\nu}$ is the weight matrix for the hidden state, $\mathbf{b}_h \in \mathbb{R}^\nu$ is the bias vector for the hidden state, φ is the activation function applied element-wise, $W_h \in \mathbb{R}^{N\times N}$ is the weight matrix for the output, $\mathbf{b}_{\tilde{x}} \in \mathbb{R}^N$ is the bias vectors for the output, and χ is the output function computed element-wise.

Similarly to FFNN, all the weights and biases can be collected in a vector $\mathbf{w} \in \mathbb{R}^p$, which has to be properly tuned by means of a training procedure based on M example realizations of the input-output pairs. The training consists of solving a nonlinear programming problem similar to (4), which minimizes the difference between the prediction and the actual output. It can be done, for instance, by using again the backpropagation algorithm adapted to the RNN architecture.[43]

As it is known, the basic RNN structure given by (5) and (6) suffers from the so-called vanishing-gradient problem.[44] Such an issue arises when training is performed by using backpropagation, as the gradients tend to diminish exponentially over long data sequences, thus making it difficult for the network to learn long-term dependencies. To address this problem, various mathematical models have been developed that incorporate specialized mechanisms, called "gates" or "cells," at each time step. Such gate-based models have proven to be effective in capturing and preserving important information over long sequences. Two prominent examples are LSTM networks[12] and GRU networks.[45] LSTM and GRU tackle the vanishing gradient issue by using gating mechanisms that control the flow of information within the network. These gates selectively decide which information to retain, update, or discard at each time step, allowing the model to learn and exploit important long-term dependencies existing on data. The ability to selectively remember or forget information makes LSTM and GRU well-suited to modeling sequential data in various research domains. Owing to their properties, in the following we focus on LSTM and GRU architectures to predict the ship motion variables, thus discarding the basic RNN setting described above. They are based on more sophisticated versions of (5) and (6), as discussed in the following.

First, let us focus on the LSTM architecture. Each LSTM unit consists of several gates: the input gate $\mathbf{i}_t$, the forget gate $\mathbf{f}_t$, the output gate $\mathbf{o}_t$, and the cell input $\mathbf{g}_t$ activation vectors. Each of these gates plays a specific role in the LSTM model. They are ruled by the following equations, which are a specialized version of (5):

$$\mathbf{i}_t = \text{sigm}(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + \mathbf{b}_i), \tag{7}$$
$$\mathbf{f}_t = \text{sigm}\big(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + \mathbf{b}_f\big), \tag{8}$$
$$\mathbf{o}_t = \text{sigm}(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + \mathbf{b}_o), \tag{9}$$
$$\mathbf{g}_t = \tanh\big(W_g\mathbf{x}_t + U_g\mathbf{h}_{t-1} + \mathbf{b}_g\big), \tag{10}$$

where W_i, W_f, W_o, W_g, U_i, U_f, U_o, U_g are weight matrices of proper dimensions, and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$, $\mathbf{b}_g$ are bias vectors. The input vector $\mathbf{x}_t$ and the state vector $\mathbf{h}_{t-1}$ are combined and transformed by the sigmoid and hyperbolic tangent functions to compute the input gate, forget gate, output gate, and cell input. The cell state $\mathbf{c}_t$ and the state vector $\mathbf{h}_t$ are then updated as follows:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i} \odot \mathbf{g}_t, \tag{11}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{12}$$

where $\odot$ represents the element-wise multiplication (Hadamard product). The forget gate $\mathbf{f}_t$ controls the flow of information from the previous cell state $\mathbf{c}_{t-1}$, allowing relevant information to be retained. The input gate $\mathbf{i}_t$ determines the importance of new information, which is multiplied by the cell input $\mathbf{g}_t$. Finally, the output gate $\mathbf{o}_t$ filters the cell state $\mathbf{c}_t$ to compute the updated state vector $\mathbf{h}_t$. The output equation corresponds to (6) with output function χ equal to an identity, that is,

$$\tilde{\mathbf{x}}_{t+1} = W_h\ \mathbf{h}_t + \mathbf{b}_{\tilde{x}}. \tag{13}$$

An extension of the LSTM architecture is the so-called BiLSTM network, which incorporates two LSTM layers to process the input data. First, an LSTM layer is applied to the input sequence in its original order (referred to as the forward layer). Then, the reversed form of the input sequence is fed into another LSTM model (known as the backward layer). By using the LSTM architecture twice, the model can effectively capture long-term dependencies in the data, leading to enhanced learning capabilities and improved accuracy.

Finally, the mathematical model of the GRU network is a variation of the LSTM network with simplified gating mechanisms. It consists of two gates, the update gate $\mathbf{z}_t$ and the reset gate $\mathbf{r}_t$. These gates control the flow of information within the network. The GRU model updates the state vector $\mathbf{h}_t$ through the following equations, which are another specialized version of (5):

$$\mathbf{z}_t = \text{sigm}(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1} + \mathbf{b}_z), \tag{14}$$
$$\mathbf{r}_t = \text{sigm}(W_r\mathbf{x}_t + U_r\mathbf{h}_{t-1} + \mathbf{b}_r), \tag{15}$$
$$\mathbf{g}_t = \tanh\big(W_g\mathbf{x}_t + U_g(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_g\big), \tag{16}$$
$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{g}_t + \mathbf{z}_t \odot \mathbf{h}_{t-1}, \tag{17}$$

where W_z, W_r, W_g, U_z, U_r, U_g are weight matrices of proper dimensions, and $\mathbf{b}_z$, $\mathbf{b}_r$, $\mathbf{b}_g$ are bias vectors, different than the ones in (7)–(10) in general. The update gate $\mathbf{z}_t$ controls the amount of information to be updated from the previous hidden state $\mathbf{h}_{t-1}$ to the current state $\mathbf{h}_t$. It ranges from 0 to 1, allowing the model to determine how much of the past information should be preserved. The reset gate $\mathbf{r}_t$ determines which information from the previous hidden state $\mathbf{h}_{t-1}$ should be ignored. It selectively resets the information based on the input $\mathbf{x}_t$ and the previous hidden state, enabling the model to adaptively capture the temporal dependencies in the data. Likewise for the case of the LSTM architecture, the output equation is equal to (6) for all t, with output function χ equal to the identity, that is,

$$\tilde{\mathbf{x}}_{t+1} = W_h\ \mathbf{h}_t + \mathbf{b}_{\tilde{x}}. \tag{18}$$

The GRU architecture provides a simpler alternative to the LSTM one, while still capturing long-term dependencies and enabling effective sequence modeling in various domains.

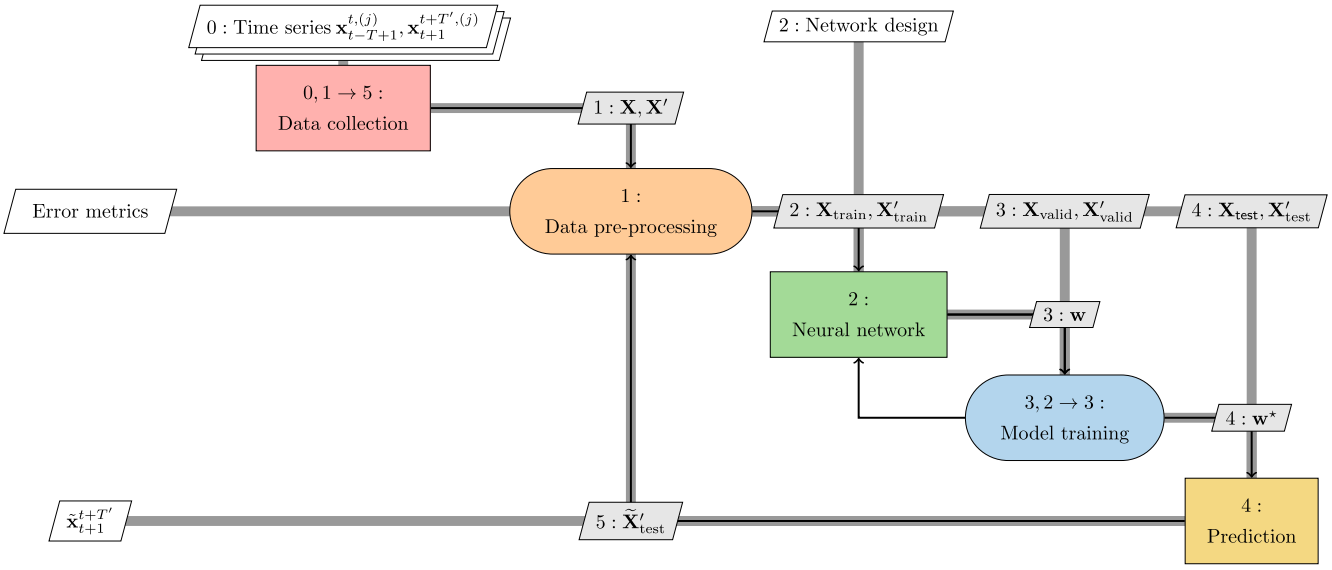

FIGURE 4 Neural network training and assessment flowchart, inspired by the extended design structure matrix formalism. The inputs are the white parallelograms at the top of the plot, while the outputs are the white parallelograms at the left.

A block diagram showing the overall training process of both FFNN and RNN, including the assessment of forecasting performance, is shown in Figure 4. The block diagram, inspired by the extended design structure matrix[46] used in the context of multidisciplinary design optimization, provides a structured overview of the time-series prediction process using both types of neural networks. Five main blocks can be identified, that is, data collection, data preprocessing, neural network, model training, and prediction. The inputs are given by the white parallelograms at the top of the flowchart, that is, the time series of ship motion variables and the neural network design (the type of network, FFNN or RNN, the number of layers, and the number of hidden units in each layer). The outputs are the white parallelograms on the left, that is, the predictions of the motion variables and the error metrics. The data collection block organizes the M available time series $\mathbf{x}_{t-T+1}^{t,(j)}$ and $\mathbf{x}_{t+1}^{t+T',(j)}, j = 1, \dots, M$, into a pair of input-output matrices $(\mathbf{X}, \mathbf{X}')$ for the supervised learning problem that will be managed, where $\mathbf{X} := \begin{bmatrix} \mathbf{x}_{t-T+1}^{t,(1)} & \dots & \mathbf{x}_{t-T+1}^{t,(M)} \end{bmatrix} \in \mathbb{R}^{N \cdot T \times M}$ is the matrix of the patterns and $\mathbf{X}' := \begin{bmatrix} \mathbf{x}_{t+1}^{t+T',(1)} & \dots & \mathbf{x}_{t+1}^{t+T',(M)} \end{bmatrix} \in \mathbb{R}^{N \cdot T' \times M}$ is the matrix of the targets. [Correction added on 24 May 2024, after first online publication: the term '$\mathbb{R}^{N \cdot T \times M}$' has been changed to '$\mathbb{R}^{N \cdot T' \times M}$' in the preceding sentence.] Each column of the matrices is a different realization of the ship motion variables. The preprocessing block includes the normalization of data and the division of data into training $(\mathbf{X}_{\text{train}}, \mathbf{X}'_{\text{train}})$, validation $(\mathbf{X}_{\text{valid}}, \mathbf{X}'_{\text{valid}})$, and test $(\mathbf{X}_{\text{test}}, \mathbf{X}'_{\text{test}})$ sets to ensure proper evaluation and avoid overfitting. Such sets are composed of different time series, as this is a common practice in the learning community: the time series used to evaluate performances are different than those used to train the models. The weights $\mathbf{w}$ of the network are then computed iteratively through the training procedure, by applying the chosen training algorithm. After the training procedure, the resulting trained network with the optimal weights $\mathbf{w}^\star$ is used with time series belonging to the test set as inputs to predict the ship motion variables. Lastly, the predicted outputs $\tilde{\mathbf{x}}_{t+1}^{t+T',(j)}, j = 1, \dots, M_{\text{test}}$, where M_{test} is the number of time series included in the test set, collected in the matrix $\tilde{\mathbf{X}}'_{\text{test}} := \begin{bmatrix} \tilde{\mathbf{x}}_{t+1}^{t+T',(1)} & \dots & \tilde{\mathbf{x}}_{t+1}^{t+T',(M_{\text{test}})} \end{bmatrix} \in \mathbb{R}^{N \cdot T' \times M_{\text{test}}}$, are compared to the test set $\mathbf{X}'_{\text{test}}$ to assess performances of the predictions by using suitable error metrics (see Section 5 for details). [Correction added on 24 May 2024, after first online publication: the term '$\mathbb{R}^{N \cdot T \times M}$' has been changed to '$\mathbb{R}^{N \cdot T' \times M}$' in the preceding sentence.]

4 | DYNAMIC MODE DECOMPOSITION

DMD is a dimensionality-reduction and reduced-order modeling technique widely used in various research fields, particularly in fluid dynamics. Originally introduced as a method to identify linear normal modes in linear systems, DMD has evolved to capture the underlying dynamics and coherent structures of complex, nonlinear systems. Its equation-free and data-driven nature has contributed to growing its popularity and success in the scientific community.[15,47] By leveraging

the concept of the Koopman operator, DMD approximates the eigenmodes and eigenvalues of the infinite-dimensional linear operator associated with the evolution over time of the system at hand. The resulting DMD modes and frequencies provide a linear finite-dimensional representation of the system dynamics, even in the presence of nonlinearities. This capability allows DMD to effectively analyze and characterize spatio-temporal coherent structures in complex flows and systems.[48–52] One of the key advantages of DMD is its data-driven approach. It only operates on measured or simulated data without requiring detailed knowledge of the state equations of the system. This makes DMD particularly well-suited for practical applications, where obtaining a precise mathematical model may be challenging or impractical.

In the context of the present work, DMD is employed to forecast the motion variables of ships operating in waves. The framework is again the sequence-to-sequence modeling approach introduced in Section 2. More specifically, the idea at the basis of the DMD approach for time series forecasting is the assumption of a linear relationship between the motion variables at a given time step τ and the same variables at the next step $\tau + 1$. In other words, we can write the following equation for $\tau = 0, 1, \ldots$:

$$\mathbf{x}_{\tau+1} = \mathcal{A}\mathbf{x}_{\tau}. \tag{19}$$

Equation (19) can be interpreted as a discrete-time linear dynamic system with a state vector given by the vector of motion variables, where $\mathcal{A} \in \mathbb{R}^{N\times N}$ is a matrix that is constructed starting from data. In particular, let us arrange the measurements of the motion variables $\mathbf{x}_{t-T+1}^{t}$ in a matrix $\mathcal{X}$ defined as follows

$$\mathcal{X} := \begin{bmatrix} \mathbf{x}_{t-T+1} & \mathbf{x}_{t-T+2} & \ldots & \mathbf{x}_{t} \end{bmatrix} \in \mathbb{R}^{N\times T}, \tag{20}$$

and let us further split $\mathcal{X}$ in two matrices $\mathcal{X}_1$ and $\mathcal{X}_2$ such that

$$\begin{aligned} \mathcal{X}_1 &:= \begin{bmatrix} \mathbf{x}_{t-T+1} & \mathbf{x}_{t-T+2} & \ldots & \mathbf{x}_{t-1} \end{bmatrix} \in \mathbb{R}^{N\times(T-1)}, \\ \mathcal{X}_2 &:= \begin{bmatrix} \mathbf{x}_{t-T+2} & \mathbf{x}_{t-T+3} & \ldots & \mathbf{x}_{t} \end{bmatrix} \in \mathbb{R}^{N\times(T-1)}. \end{aligned} \tag{21}$$

Then, the matrix $\mathcal{A}$ is approximated starting from the collected data using the equation

$$\mathcal{A} \approx \mathcal{X}_2\mathcal{X}_1^{\dagger}, \tag{22}$$

where $\mathcal{X}_1^{\dagger}$ is the Moore–Penrose pseudo-inverse of $\mathcal{X}_1$ that minimizes $|\mathcal{X}_2 - \mathcal{A}\mathcal{X}_1|_F$, with $|\cdot|_F$ being the Frobenius norm.

Then, the time evolution of the motions of the vessel in waves can be approximated (and predicted from time $t + 1$ up to $t + T'$) using the following modal expansion:

$$\tilde{\mathcal{X}}' := \begin{bmatrix} \boldsymbol{\phi}_1 & \ldots & \boldsymbol{\phi}_N \end{bmatrix} \begin{bmatrix} b_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & b_N \end{bmatrix} \begin{bmatrix} e^{\omega_1(t+1)\Delta t} & \ldots & e^{\omega_1(t+T')\Delta t} \\ \vdots & \ddots & \vdots \\ e^{\omega_N(t+1)\Delta t} & \ldots & e^{\omega_N(t+T')\Delta t} \end{bmatrix}, \tag{23}$$

where $\tilde{\mathcal{X}}' := [\tilde{\mathbf{x}}_{t+1} \ \ldots \ \tilde{\mathbf{x}}_{t+T'}] \in \mathbb{R}^{N\times T'}$, with

$$\tilde{\mathbf{x}}_{\tau} = \chi\left(\boldsymbol{\phi}_j, b_j, \omega_j\right) := \sum_{j=1}^{N} \boldsymbol{\phi}_j b_j e^{\omega_j \tau \Delta t}, \quad \tau = t+1, \ldots, t+T'. \tag{24}$$

In (23) and (24), $\boldsymbol{\Phi} := \begin{bmatrix} \boldsymbol{\phi}_1 & \ldots & \boldsymbol{\phi}_N \end{bmatrix} \in \mathbb{R}^{N\times N}$ collects the eigenvectors $\boldsymbol{\phi}_j$, $j = 1, \ldots, N$, of the approximated matrix $\mathcal{A}$ as in (22), the coefficients b_j, $j = 1, \ldots, N$, are the modal coordinates of the initial condition $\mathbf{x}_t$ in the eigenvector basis, for which $\mathbf{b} := \begin{bmatrix} b_1 & \ldots & b_N \end{bmatrix} = \boldsymbol{\Phi}^{-1}\mathbf{x}_t$ and $\omega_j = \ln(\lambda_j)/\Delta t$, $j = 1, \ldots, N$, with λ_j being the eigenvalues of $\mathcal{A}$, and Δt is the sampling time.[5] Figure 5 sketches how to use DMD for time series forecasting.

From a general perspective, DMD can be viewed as a method to compute the eigenvalues and eigenvectors (modes) of a finite-dimensional linear model that approximates the infinite-dimensional linear Koopman operator,[5] also known as composition operator. In the present context, due to the low dimensionality of the data, (22) can be directly computed

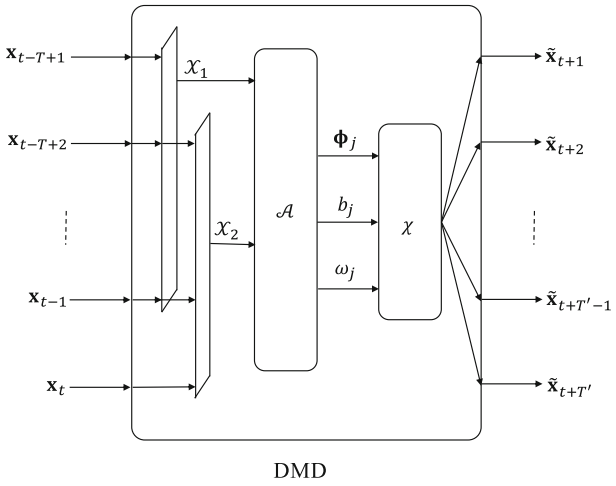

FIGURE 5 Sketch of DMD for time series forecasting.

without the need for performing singular value decomposition of $\mathcal{X}$ and projecting onto proper orthogonal decomposition modes.[5]

The DMD formulation can be extended through a technique called state augmentation,[1] with the goal of increasing prediction accuracy. The resulting approach is often referred to as augmented DMD (ADMD). In this case, the matrix $\mathcal{A}$ is transformed into an augmented matrix $\hat{\mathcal{A}}$. State augmentation consists of inserting additional variables into the state vector. In more detail, three strategies can be pursued: (i) time derivatives, (ii) time-shifted copies (delayed states), and (iii) a combination of time derivatives and time-shifted copies.

In the first case, we define the following time-derivative matrices:

$$\mathcal{D}_1 := \begin{bmatrix} \frac{\mathrm{d}\mathcal{X}_1}{\mathrm{d}t} \\ \frac{\mathrm{d}^2\mathcal{X}_1}{\mathrm{d}t^2} \\ \vdots \\ \frac{\mathrm{d}^l\mathcal{X}_1}{\mathrm{d}t^l} \end{bmatrix}, \quad \mathcal{D}_2 := \begin{bmatrix} \frac{\mathrm{d}\mathcal{X}_2}{\mathrm{d}t} \\ \frac{\mathrm{d}^2\mathcal{X}_2}{\mathrm{d}t^2} \\ \vdots \\ \frac{\mathrm{d}^l\mathcal{X}_2}{\mathrm{d}t^l} \end{bmatrix}, \tag{25}$$

where l is the number of time derivatives. The matrices $\mathcal{D}_1$ and $\mathcal{D}_2$ are then used to construct an augmented version of the matrices in (21), as follows:

$$\hat{\mathcal{X}}_1 := \begin{bmatrix} \mathcal{X}_1 \\ \mathcal{D}_1 \end{bmatrix}, \quad \hat{\mathcal{X}}_2 := \begin{bmatrix} \mathcal{X}_2 \\ \mathcal{D}_2 \end{bmatrix}. \tag{26}$$

In this article, time derivatives are evaluated using a backward finite difference scheme, which only requires past states.

In the second case, we define the following matrices of time-shifted copies in the form of Hankel matrices, that is,

$$\mathcal{S}_1 := \begin{bmatrix} \mathbf{x}_{t-T+1} & \mathbf{x}_{t-T+2} & \cdots & \mathbf{x}_{t-1} \\ \mathbf{x}_{t-T} & \mathbf{x}_{t-T+1} & \cdots & \mathbf{x}_{t-2} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}_{t-T-s+1} & \mathbf{x}_{t-T-s+2} & \cdots & \mathbf{x}_{t-s} \end{bmatrix}, \quad \mathcal{S}_2 := \begin{bmatrix} \mathbf{x}_{t-T+2} & \mathbf{x}_{t-T+3} & \cdots & \mathbf{x}_{t} \\ \mathbf{x}_{t-T+1} & \mathbf{x}_{t-T+2} & \cdots & \mathbf{x}_{t-1} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}_{t-T-s} & \mathbf{x}_{t-T-s+1} & \cdots & \mathbf{x}_{t-s+1} \end{bmatrix}, \tag{27}$$

where s is the number of time-shifted copies. Then, the matrices $\mathcal{S}_1$ and $\mathcal{S}_2$ are used to construct an augmented version of the matrices in (21) in the following way:

$$\hat{\mathcal{X}}_1 := \begin{bmatrix} \mathcal{X}_1 \\ \mathcal{S}_1 \end{bmatrix}, \quad \hat{\mathcal{X}}_2 := \begin{bmatrix} \mathcal{X}_2 \\ \mathcal{S}_2 \end{bmatrix}. \tag{28}$$

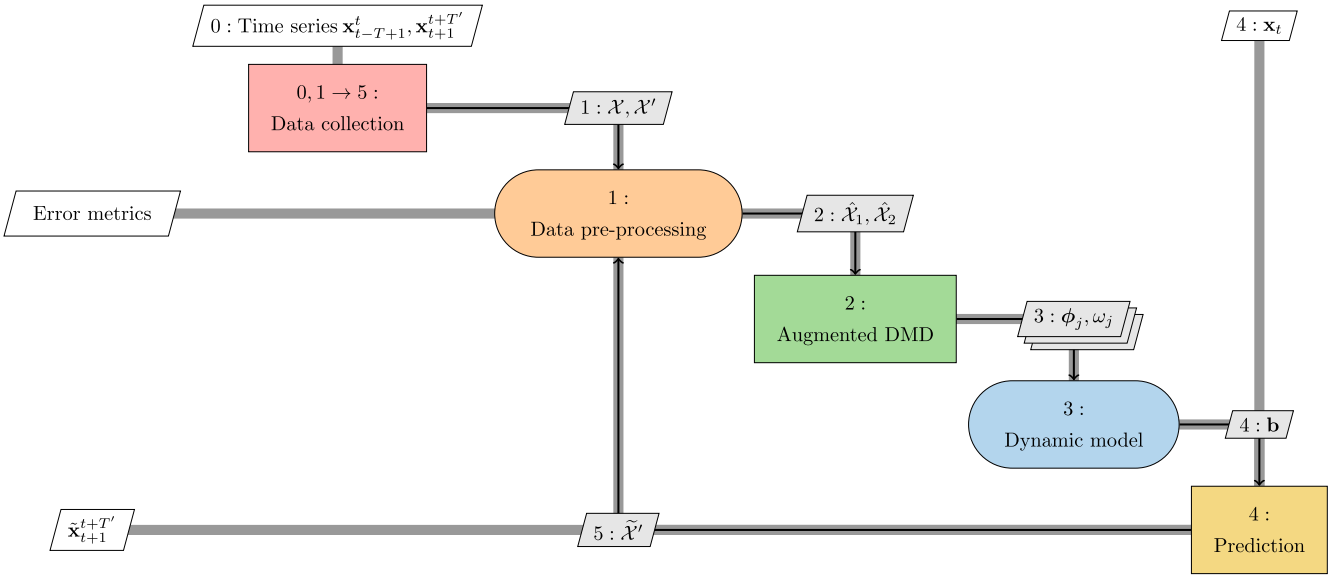

FIGURE 6 ADMD training and assessment flowchart, inspired by the extended design structure matrix formalism. The inputs are the white parallelograms at the top of the plot, while the outputs are the white parallelograms at the left.

Lastly, the use of both time derivatives and time-shifted copies provides a new augmented version of the matrices in (21), as follows:

$$\hat{\mathcal{X}}_1 := \begin{bmatrix} \mathcal{X}_1 \\ \mathcal{D}_1 \\ \mathcal{S}_1 \end{bmatrix}, \quad \hat{\mathcal{X}}_2 := \begin{bmatrix} \mathcal{X}_2 \\ \mathcal{D}_2 \\ \mathcal{S}_2 \end{bmatrix}. \tag{29}$$

It is worth noting that, for the case study considered in this article, the number of state and augmented variables is relatively low, which results in low-dimensional state matrices. Therefore, the overall modeling and forecasting process, including the solution of the matrix eigenproblem, does not suffer from significant complexity or computational cost. Model-order reduction, as typically employed in standard DMD, is not necessary in this case. Instead, the model order is increased by the inclusion of augmented states to improve modeling and forecasting accuracy.

A block diagram showing the overall approximation process, including the assessment of forecasting performance and following again the formalism of the extended design structure matrix,[46] is shown in Figure 6. The block diagram outlines the general flow from data acquisition to prediction and evaluation, emphasizing how ADMD is employed to predict the ship motion variables. Likewise for the case of neural networks, five main blocks can be identified, that is, data collection, data preprocessing, ADMD, dynamic model, and prediction. At a given time instant t, the inputs (represented by the white parallelograms at the top of the plot) are the time series input–output pairs of ship motion variables and their value $\mathbf{x}_t$ as the initial condition for the prediction up to time $t + T'$, while the outputs (given by the white parallelograms at the left) are the predictions of the motion variables and the performance evaluation. The key lies in the data preprocessing block building the augmented data matrices $\hat{\mathcal{X}}_1$ and $\hat{\mathcal{X}}_2$, along with the subsequent computation and use of dynamic modes to capture the essential characteristics of the time series, thus enabling informed predictions about the future behavior of the motion variables.

5 | PERFORMANCE METRICS

The effectiveness of the proposed approaches to forecast ship motions is assessed through the following four performance metrics: the normalized root mean square error (NRMSE), the Pearson correlation coefficient (R), the average angle measure (AAM), and the normalized average minimum/maximum absolute error (NAMMAE). All the metrics are computed over the test set and are averaged over the motion variables to provide an overall assessment of the prediction accuracy.

They constitute a comprehensive evaluation of the prediction accuracy by considering different aspects such as overall error, correlation, angle differences, and range of predicted and measured values.

The NRMSE measures the average root mean square error between the predicted values $\tilde{\mathbf{x}}_t$ and the measured (test) values $\mathbf{x}_t$ at the various time steps. It is computed through the square root of the average squared differences, normalized by the standard deviation of the measured values:

$$\text{NRMSE} := \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{\mathcal{T}\sigma_{x_i}^2}\sum_{j=1}^{\mathcal{T}}\left(\tilde{x}_{ij}-x_{ij}\right)^2}, \tag{30}$$

where N is the number of motion variables, $\mathcal{T}$ is the number of considered time instants, and σ_{x_i} is the standard deviation of the measured values for the variable i, $i = 1, \dots, N$.

The R coefficient measures the linear correlation between the predicted values and the measured ones. It is computed as the average correlation coefficient over all the variables and time instants, as follows:

$$\text{R} := \frac{1}{N(\mathcal{T}-1)}\sum_{i=1}^{N}\sum_{j=1}^{\mathcal{T}}\frac{\left(\tilde{x}_{ij}-\bar{\tilde{x}}_i\right)^2}{\sigma_{\tilde{x}_i}}\frac{\left(x_{ij}-\bar{x}_i\right)^2}{\sigma_{x_i}}, \tag{31}$$

where $\bar{\tilde{x}}_i$ and $\bar{x}_i$ are the time averages of the predicted and measured values for variable i, $i = 1, \dots, N$, respectively, while $\sigma_{\tilde{x}_i}$ and σ_{x_i} are the corresponding standard deviations.

The AAM is another metric that quantifies the accuracy of predicted time series compared to actual, measured ones. It is computed as the average of a measure based on the angles between the predicted and measured values. It is given by

$$\text{AAM} := \frac{1}{N}\sum_{i=1}^{N}\left(1-\frac{4}{\pi}\left(\frac{\sum_{j=1}^{\mathcal{T}}d_{ij}|\alpha_{ij}|}{\sum_{j=1}^{\mathcal{T}}d_{ij}}\right)\right), \tag{32}$$

where $\alpha_{ij} := \arccos\left(\frac{|\tilde{x}_{ij}+x_{ij}|}{\sqrt{2}d_{ij}}\right)$ and $d_{ij} := \sqrt{\tilde{x}_{ij}^2+x_{ij}^2}$.

Lastly, the NAMMAE metric provides an engineering-oriented assessment of the prediction accuracy. It measures the absolute difference between the minimum and maximum values of the predicted and measured time series, as follows:

$$\text{NAMMAE} := \frac{1}{2N\sigma_{x_i}}\sum_{i=1}^{N}\left(\left|\min_j(\tilde{x}_{ij})-\min_j(x_{ij})\right|+\left|\max_j(\tilde{x}_{ij})-\max_j(x_{ij})\right|\right), \tag{33}$$

where σ_{x_i} is again the standard deviation of the measured values for variable i, $i = 1, \dots, N$.

6 | DESCRIPTION OF THE CASE STUDIES AND NUMERICAL RESULTS

In this section, we first briefly describe the case studies we focused on together with the numerical setup used to apply the considered data-driven modeling approaches. Then, we present and discuss the obtained numerical results assessing the effectiveness of the proposed approaches.

6.1 | Description of the case studies

We considered the behavior of a naval destroyer evaluated at model scale in a high sea state in two case studies, which are quite challenging due to both the height and direction of waves. In more detail, we focused on the MARIN model scale 7967, which is equivalent to the naval destroyer 5415M model scale. Such a model was used as a test case for NATO AVT-280.[53] The model is self-propelled and kept on course by a proportional-derivative controller actuating the rudder angle. For the considered test cases, data were generated by computational fluid dynamics (CFD) simulations with

propeller revolutions per minute fixed to the self-propulsion point of the model for the nominal speed, corresponding to a Froude number equal to 0.33. The simulations, performed with the URANS code CFDShip-Iowa V4.5,[54] were conducted in irregular long-crested waves (following a JONSWAP spectrum), with a nominal peak period equal to 9.2 s and a wave heading of 300° (stern-quartering). The nominal significant wave height was equal to 7 m, corresponding to a sea state 7 according to the World Meteorological Organization definition. We observe that the simulation setup is close to a resonance condition for the roll, which means that the vessel model may easily capsize, thus making crucial the availability of effective prediction tools of the ship motion variables like those proposed in this article. In fact, accurate predictions may be the basis for the computation of suitable control actions to avoid an undesired behavior of the vessel.

Data for training and testing were generated by solving the six degrees of freedom rigid-body equations of motion to compute the linear and angular motions of the ship. A simplified body-force model was used for the propeller, which prescribes axisymmetric body force with axial and tangential components. The total number of grid points for the numerical solver was equal to about 45×10^6. Further details about the numerical solver of the motion equations can be found in Serani et al.[32] and van Walree et al.,[53] where potential flow computations and experimental data are presented and discussed. Eight different CFD runs were considered that correspond to a total number of encounter waves equal to 132. The variables describing the motions of ships, and for which we set the data-driven approaches described in Sections 3 and 4 for the purpose of forecasting, are the six degrees of freedom of the ship, that is, surge, sway, heave, roll, pitch, and yaw (the ship motions in the carriage coordinate system projected onto the ship axes) along with the rudder angle. The time resolution was set to have $T_e = 32$ time steps per encounter wave on the average. Time derivatives were evaluated using a second-order, backward finite-difference scheme. All the variables (including time derivatives) were standardized, that is, they were translated and scaled to have zero mean and unit variance (z-score standardization).

Results are discussed for two different case studies:

- Case study A: it considers the prediction over 1 future encounter wave period (short-term prediction);
- Case study B: it considers the prediction of 3 future wave encounter periods (medium-term prediction).

The two case studies mainly differ in the horizon length of the prediction, which plays a crucial role in both prediction accuracy and required computational burden, thus making the two settings very different from each other.

6.2 | Numerical results

Data-driven forecasting of ship motions in waves was first explored with FFNN. As a first simulation round, we investigated different network configurations with an increasing number of units in the hidden layer of the network, taken equal to 3, 5, 10, 30, and 50. Furthermore, we considered an increasing number of input waves, that is, the length of the input sequence expressed in terms of the number of encounter waves, fixed to 1, 3, 5, and 10 encounter waves, which correspond to 32, 96, 160, and 320 time steps, respectively. Note that self-repetition is not present within the input sequences. Output sequences made up of 1 (Case study A) and 3 (Case study B) encounter waves were considered, corresponding to 32 and 96 time steps, respectively. A five-fold cross validation was used, exploiting 4 URANS runs as the training set, 2 runs as the validation set, and 2 runs as the test set. As pointed out also in Section 3, such sets were composed of different time series to allow a fair performance evaluation. Training of neural networks was performed using the training and validation sets, while performances were evaluated over the test set. In particular, one network for each vessel motion variable was trained, that is, we considered a multi-input/single-output structure to reduce the computational burden required for the training.

The obtained results are summarized in Table 1 in terms of average value (avg) and standard deviation (std) of the performance indicators introduced in Section 5, and in Figure 7 in terms of boxplots of the NRMSE. The boxplots show the first, second (equivalent to the median value), and third quartiles, while the whiskers extend from the box to the farthest data point lying within 1.5 times the inter-quartile range, defined as the difference between the third and the first quartiles from the box. Outliers are not shown to enhance the readability of the plot. We observe that, for all the metrics introduced in Section 5, a reduced number of hidden units is enough to obtain satisfactory results, while more complex network structures, that is, FFNN with a larger number of hidden units, do not provide better results. The same behavior occurs when using a higher number of input waves, due to the difficulty of managing too long input sequences, which do not provide useful information for the future ship motions, but only contribute to making the training procedure more difficult to perform due to the increased amount of data to deal with. In other words, we may conclude that the

TABLE 1 FFNN performance metrics on the test set: Averages (avg) and standard deviations (std).

		Number of input waves	Case study A					Case study B				
			Number of hidden units					Number of hidden units				
			3	5	10	30	50	3	5	10	30	50
NRMSE	avg	1	0.2171	0.1977	0.2021	0.2241	0.2265	0.4042	0.3902	0.3900	0.4647	0.5080
	(std)		(0.0719)	(0.0727)	(0.0879)	(0.1076)	(0.1178)	(0.0988)	(0.1150)	(0.1264)	(0.1993)	(0.2359)
	avg	3	0.2295	0.2268	0.2042	0.2028	0.2128	0.4031	0.3829	0.3942	0.4526	0.4577
	(std)		(0.0809)	(0.0981)	(0.0799)	(0.0865)	(0.1031)	(0.0907)	(0.0963)	(0.1194)	(0.1867)	(0.1931)
	avg	5	0.2433	0.2195	0.2159	0.2505	0.2383	0.4387	0.4295	0.4148	0.4744	0.5198
	(std)		(0.0849)	(0.0806)	(0.0898)	(0.1273)	(0.1188)	(0.1084)	(0.1173)	(0.1238)	(0.1954)	(0.2572)
	avg	10	0.3499	0.3493	0.3326	0.3512	0.3748	0.6601	0.6874	0.7155	0.6719	0.6818
	(std)		(0.1529)	(0.1809)	(0.1808)	(0.2065)	(0.2392)	(0.2780)	(0.3471)	(0.3946)	(0.3889)	(0.4040)
NAMMAE	avg	1	0.1729	0.1608	0.1674	0.2036	0.2029	0.3086	0.3053	0.3207	0.4597	0.5467
	(std)		(0.0596)	(0.0624)	(0.0768)	(0.1080)	(0.1185)	(0.0748)	(0.1080)	(0.1320)	(0.2500)	(0.3270)
	avg	3	0.1776	0.1816	0.1657	0.1795	0.2025	0.3028	0.2824	0.3143	0.4578	0.4885
	(std)		(0.0615)	(0.0790)	(0.0658)	(0.0853)	(0.1139)	(0.0641)	(0.0697)	(0.1158)	(0.2388)	(0.2728)
	avg	5	0.1862	0.1751	0.1747	0.2368	0.2291	0.3302	0.3187	0.3137	0.4321	0.5432
	(std)		(0.0619)	(0.0641)	(0.0733)	(0.1363)	(0.1326)	(0.0746)	(0.0902)	(0.0983)	(0.2066)	(0.3256)
	avg	10	0.2806	0.2831	0.2913	0.3642	0.4286	0.5053	0.6069	0.7193	0.7282	0.7815
	(std)		(0.1230)	(0.1469)	(0.1625)	(0.2321)	(0.2952)	(0.2126)	(0.3139)	(0.4049)	(0.4288)	(0.4709)
R	avg	1	0.9269	0.9500	0.9506	0.9447	0.9428	0.7703	0.8335	0.8377	0.8444	0.8378
	(std)		(0.0151)	(0.0194)	(0.0220)	(0.0257)	(0.0295)	(0.0314)	(0.0402)	(0.0439)	(0.0502)	(0.0582)
	avg	3	0.9189	0.9388	0.9464	0.9415	0.9399	0.7793	0.8236	0.8301	0.8421	0.8390
	(std)		(0.0233)	(0.0285)	(0.0238)	(0.0281)	(0.0304)	(0.0409)	(0.0453)	(0.0507)	(0.0642)	(0.0592)
	avg	5	0.8809	0.9383	0.9408	0.9316	0.9221	0.7648	0.7971	0.8111	0.8276	0.8275
	(std)		(0.0292)	(0.0302)	(0.0297)	(0.0363)	(0.0399)	(0.0499)	(0.0641)	(0.0598)	(0.0746)	(0.0813)
	avg	10	0.8284	0.8347	0.8515	0.8452	0.8258	0.6508	0.6778	0.6860	0.6803	0.6808
	(std)		(0.0818)	(0.0939)	(0.0879)	(0.0931)	(0.1063)	(0.1572)	(0.1811)	(0.1898)	(0.1967)	(0.1974)
AAM	avg	1	0.8480	0.8670	0.8731	0.8690	0.8714	0.6993	0.7289	0.7380	0.7342	0.7332
	(std)		(0.0207)	(0.0215)	(0.0252)	(0.0317)	(0.0339)	(0.0327)	(0.0377)	(0.0433)	(0.0584)	(0.0589)
	avg	3	0.8419	0.8555	0.8619	0.8693	0.8700	0.6948	0.7215	0.7283	0.7356	0.7324
	(std)		(0.0281)	(0.0328)	(0.0300)	(0.0339)	(0.0380)	(0.0392)	(0.0447)	(0.0516)	(0.0671)	(0.0650)
	avg	5	0.8284	0.8488	0.8541	0.8535	0.8532	0.6726	0.6939	0.7099	0.7230	0.7280
	(std)		(0.0363)	(0.0354)	(0.0402)	(0.0498)	(0.0500)	(0.0532)	(0.0613)	(0.0645)	(0.0784)	(0.0875)
	avg	10	0.7496	0.7661	0.7788	0.7809	0.7774	0.5892	0.6167	0.6344	0.6433	0.6476
	(std)		(0.0991)	(0.1093)	(0.1094)	(0.1141)	(0.1235)	(0.1462)	(0.1726)	(0.1872)	(0.1937)	(0.1961)

future motions of the ship only depend on the recent past. The best compromise solution is achieved by using 5 hidden units with 1 and 3 input waves, for Case studies A and B, respectively, that is, a number of input waves equal to the number of output waves is necessary to obtain the best forecasts. Examples of FFNN predictions in these configurations for randomly-selected time series belonging to the test set are shown in Figure 8. It can be seen that all the motion variables as well as the rudder angle are well predicted for Case study A and fairly predicted for Case study B. In particular, for the considered time series of Case study B, angular motions and rudder are well predicted for all the considered 3 output waves, whereas linear motion predictions exhibit a small divergence after 1 output wave.

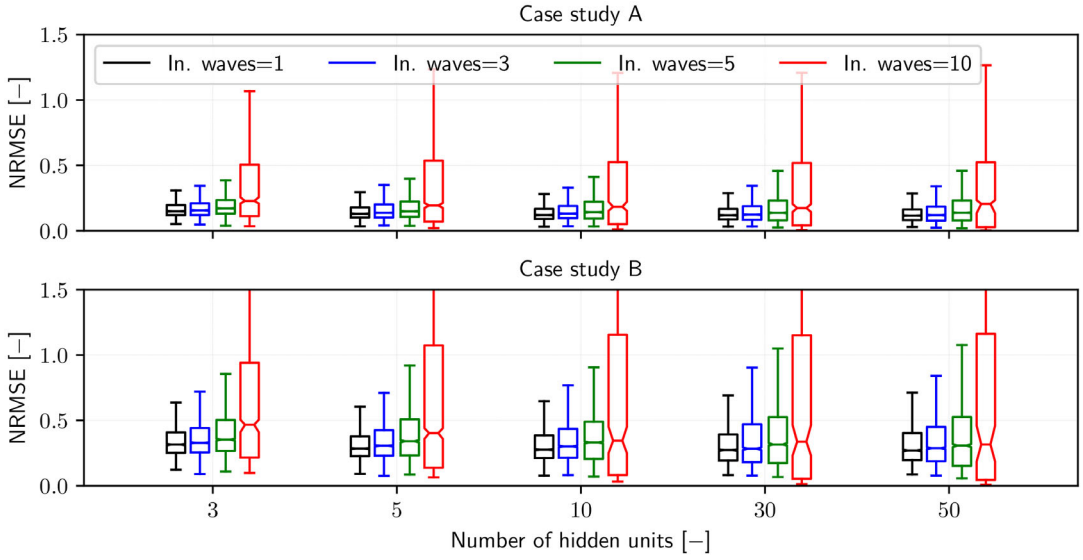

FIGURE 7 NRMSE boxplots for FFNN on the test set, conditional to the number of hidden units and input waves.

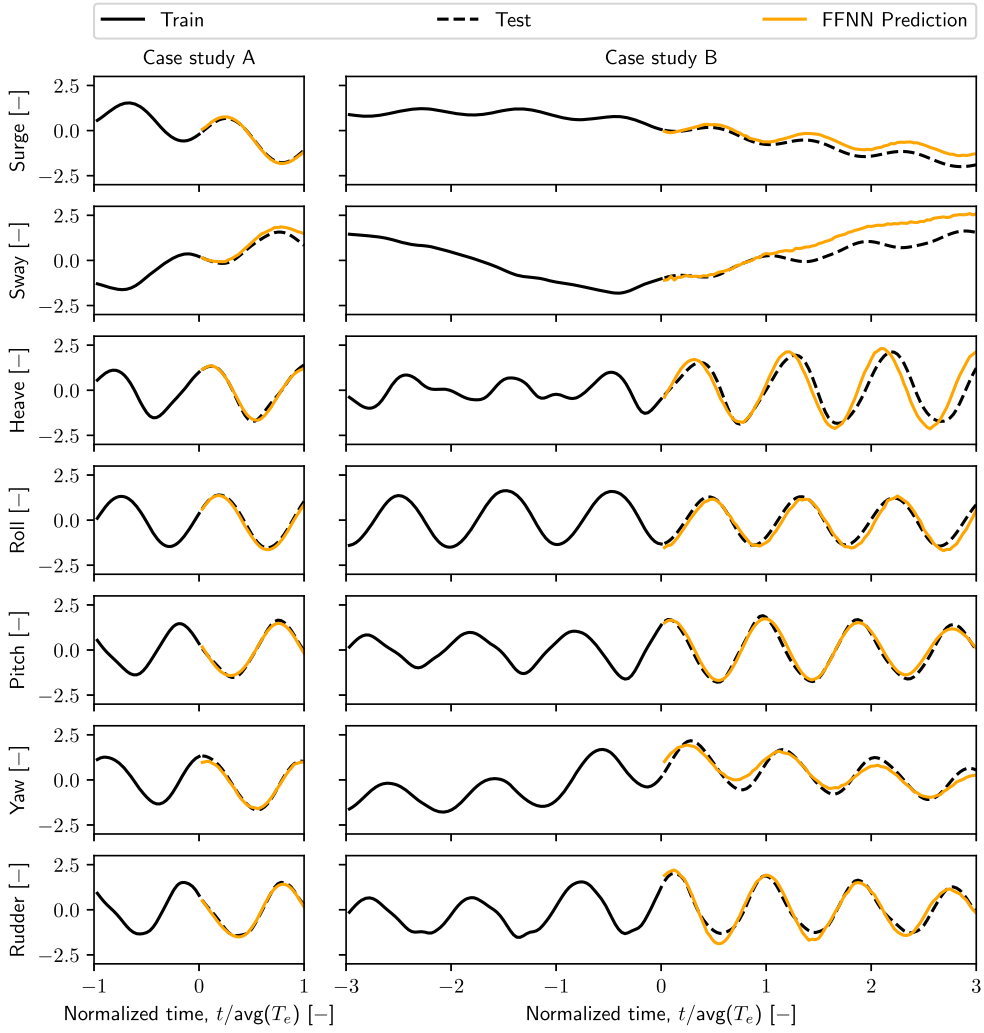

FIGURE 8 Example of prediction results provided by FFNN at a given time instant t of the six motion variables along with the rudder angle in randomly-selected instances belonging to the test set. The horizontal axis represents the normalized time, that is, the number of encounter waves from the current time instant (denoted with $t = 0$). Case study A is depicted on the left, while Case study B is reported on the right.

TABLE 2 RNN performance metrics on the test set: averages (avg) and standard deviations (std).

			Case study A					Case study B				
		RNN type	Number of hidden units					Number of hidden units				
			1	3	7	14	49	1	3	7	14	49
NRMSE	avg	LSTM	0.4556	0.2407	0.2007	0.1911	0.1800	0.5342	0.4087	0.3964	0.3943	0.4019
	(std)		(0.0950)	(0.0568)	(0.0498)	(0.0500)	(0.0492)	(0.0884)	(0.0708)	(0.0791)	(0.0844)	(0.0892)
	avg	BiLSTM	0.2637	0.2007	0.1695	0.1566	0.1440	0.4304	0.3783	0.3521	0.3462	0.3589
	(std)		(0.0638)	(0.0491)	(0.0412)	(0.0390)	(0.0384)	(0.0749)	(0.0762)	(0.0820)	(0.0826)	(0.0903)
	avg	GRU	0.4588	0.2449	0.2035	0.1910	0.1806	0.5364	0.4158	0.3985	0.3955	0.3913
	(std)		(0.0965)	(0.0584)	(0.0496)	(0.0490)	(0.0484)	(0.0910)	(0.0772)	(0.0795)	(0.0817)	(0.0848)
NAMMAE	avg	LSTM	0.3831	0.1575	0.1266	0.1127	0.1008	0.5565	0.2911	0.2358	0.2184	0.2174
	(std)		(0.1076)	(0.0512)	(0.0373)	(0.0302)	(0.0286)	(0.1334)	(0.0875)	(0.0733)	(0.0677)	(0.0689)
	avg	BiLSTM	0.1766	0.1320	0.1098	0.0981	0.0875	0.3372	0.2347	0.1967	0.1885	0.1943
	(std)		(0.0558)	(0.0400)	(0.0283)	(0.0258)	(0.0245)	(0.1035)	(0.0759)	(0.0619)	(0.0569)	(0.0592)
	avg	GRU	0.4182	0.1651	0.1366	0.1211	0.1067	0.5924	0.3125	0.2627	0.2342	0.2124
	(std)		(0.1205)	(0.0575)	(0.0405)	(0.0333)	(0.0293)	(0.1468)	(0.1139)	(0.0960)	(0.0788)	(0.0690)
R	avg	LSTM	0.6338	0.8732	0.9199	0.9275	0.9322	0.4231	0.6628	0.6701	0.6799	0.6778
	(std)		(0.1296)	(0.1063)	(0.0687)	(0.0655)	(0.0667)	(0.1209)	(0.1468)	(0.1607)	(0.1676)	(0.1714)
	avg	BiLSTM	0.8467	0.9187	0.9402	0.9466	0.9525	0.6478	0.6850	0.7226	0.7356	0.7392
	(std)		(0.1209)	(0.0687)	(0.0545)	(0.0500)	(0.0500)	(0.1468)	(0.1684)	(0.1685)	(0.1636)	(0.1613)
	avg	GRU	0.6292	0.8708	0.9193	0.9288	0.9335	0.4198	0.6606	0.6636	0.6772	0.7066
	(std)		(0.1296)	(0.1042)	(0.0697)	(0.0663)	(0.0667)	(0.1223)	(0.1471)	(0.1578)	(0.1606)	(0.1530)
AAM	avg	LSTM	0.5543	0.7855	0.8281	0.8382	0.8476	0.4075	0.5935	0.6153	0.6228	0.6175
	(std)		(0.0713)	(0.0822)	(0.0644)	(0.0643)	(0.0674)	(0.0820)	(0.1066)	(0.1081)	(0.1120)	(0.1140)
	avg	BiLSTM	0.7627	0.8280	0.8567	0.8686	0.8788	0.5660	0.6279	0.6580	0.6654	0.6602
	(std)		(0.0886)	(0.0643)	(0.0552)	(0.0525)	(0.0529)	(0.1002)	(0.1129)	(0.1161)	(0.1173)	(0.1164)
	avg	GRU	0.5504	0.7821	0.8247	0.8374	0.8468	0.4046	0.5891	0.6097	0.6185	0.6289
	(std)		(0.0723)	(0.0835)	(0.0655)	(0.0628)	(0.0657)	(0.0871)	(0.1106)	(0.1097)	(0.1131)	(0.1098)

A second simulation round involving RNN was performed starting from the outcomes of the first round based on FFNN. Hence, 1 and 3 input waves were considered as a test bed for Case studies A and B, respectively. The analysis was conducted by varying the type of hidden layer (one LSTM, BiLSTM, and GRU layer was considered), as well as the number of hidden units. Similarly to the case of FFNN, we also investigated the effect of adopting network structures of increasing complexity by considering 1, 3, 7, 14, and 49 hidden units. Such numbers were chosen to be proportional to the number of input variables of the networks, that is, the six degrees of freedom plus the rudder angle, in order to check the existence of possible correlations. Furthermore, to better generalize the network prediction, a 5% dropout layer was considered. A five-fold cross-validation was used, exploiting 5 URANS runs as the training set, 2 runs as the validation set, and the remaining one as the test set. Moreover, 11 Monte Carlo training procedures were conducted to consider the effects of the dropout. Conversely to FFNN, a single RNN for all the ship motion variables was trained, meaning that a multi-input/multi-output architecture was used.

The obtained results are summarized in Table 2 and Figure 9. It can be observed that BiLSTM networks always provide the best results on average, and that a number of hidden units equal to 49 (square of the number of input variables) and 14 (double the number of inputs) for Case studies A and B, respectively, allow to achieve the best prediction performance. As observed also for FFNN, we can deduce that the longer is the prediction time window, the less complex the hidden structure of the network should be to achieve good performance. In other words, it can be asserted that if the prediction time window increases, the forecasting problem becomes more complex, and training networks with a

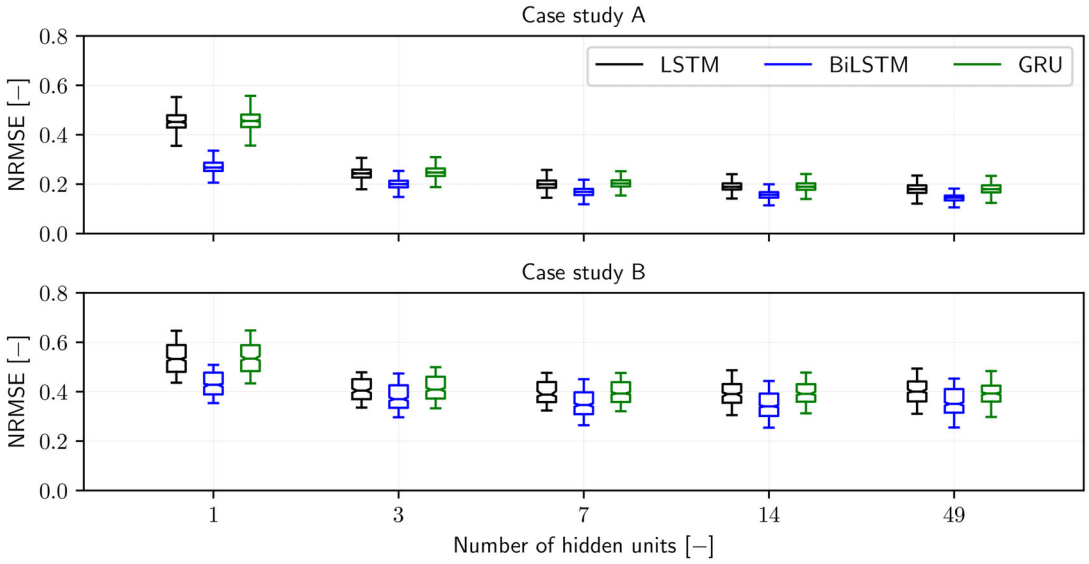

FIGURE 9 NRMSE boxplots for RNN on the test set, conditional to layer type and number of hidden units.

larger width become more challenging. It would be interesting to investigate in future studies whether similar results are obtained by also exploring the network depth in terms of the number of hidden layers. Examples of BiLSTM predictions for randomly-selected time series belonging to the test set are shown in Figure 10 for both Case study A and Case study B. We point out that all motion variables are well captured, and only a few discrepancies can be noted in Case study B for planar motion variables (surge and sway) after 2.5 output waves.

A third simulation round regarded the use of DMD in its augmented formulation (ADMD), with augmentation involving both time derivatives and time-shifted copies (delayed states), following Serani et al.[1] Likewise for the case of RNN, we considered 1 input wave and 1 output wave for Case study A, as well as 3 input waves and 3 output waves for Case study B. The number of time derivatives to augment the DMD data matrix was considered to vary from 1 to 4, whereas the number of time-shifted copies was taken equal to 1, 2, 4, 8, and 16. The results computed over the test set are summarized in Table 3 and Figure 11. We observe that the best compromise among the performance metrics is achieved by using 2 time derivatives and 16 shifted copies of time histories for Case study A, as well as 1 time derivative and 2 time-shifted copies for Case study B. We observe that adding time-shifted copies becomes counterproductive as the temporal length of the prediction increases, similar to the inclusion of higher-order derivatives beyond the second order. This phenomenon suggests that, for longer prediction horizons, a more nuanced approach is required, potentially involving a careful balance between the temporal expansion of the data and the complexity of the model. Examples of ADMD predictions with such configurations for randomly-selected time series belonging to the test set are reported in Figure 12 for both Case studies A and B. It turns out that the prediction is fairly in agreement with the test set, even if several discrepancies are visible for both case studies.

As regards the comparison of the three considered prediction methods, that is, FFNN, RNN, and ADMD, Figure 13 shows the correlation between the different evaluation metrics introduced in Section 5, where each point represents the accuracy of a sequence predicted by FFNN, BiLSTM, and ADMD considering the corresponding best configuration, as discussed in the aforementioned three rounds of tests. Performances are also summarized in Table 4. It can be noted that FFNN provide the best results in Case study B when focusing on the R and AAM metrics. However, if we consider all the metrics, BiLSTM networks have the lowest dispersion in terms of NRMSE and NAMMAE, and therefore they guarantee the best outcomes. Conversely, ADMD always provides the worst results.

Finally, a statistical assessment of the obtained results was conducted by considering again the best setup for FFNN, BiLSTM, and ADMD, both in Case study A and Case study B. In more detail, the Diebold-Mariano test[55] and the Kolmogorov-Smirnov test[56] were employed for pairwise comparisons of forecasting accuracy and performance distribution, respectively, for FFNN, BiLSTM, and ADMD across all performance metrics. The Diebold-Mariano and Kolmogorov-Smirnov tests were performed over 500 observations and summarized in Table 5. For the Diebold-Mariano test, the table reports the mean difference ($\bar{d}$), the associated standard error (SE($\bar{d}$)), and the

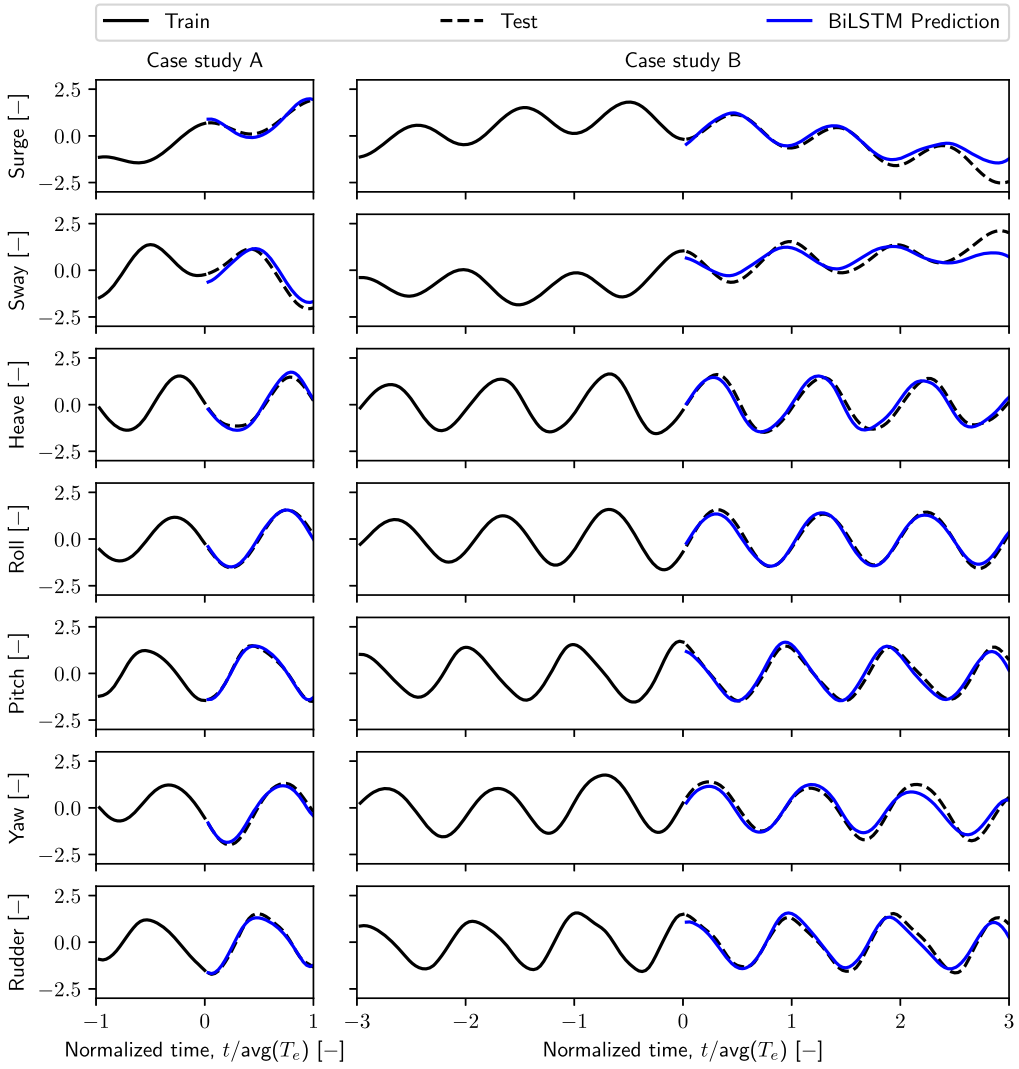

FIGURE 10 Example of prediction results provided by BiLSTM at a given time instant t of the six motion variables along with the rudder angle in randomly-selected instances belonging to the test set. The horizontal axis represents the normalized time, that is, the number of encounter waves from the current time instant (denoted with $t = 0$). Case study A is depicted on the left, while Case study B is reported on the right.

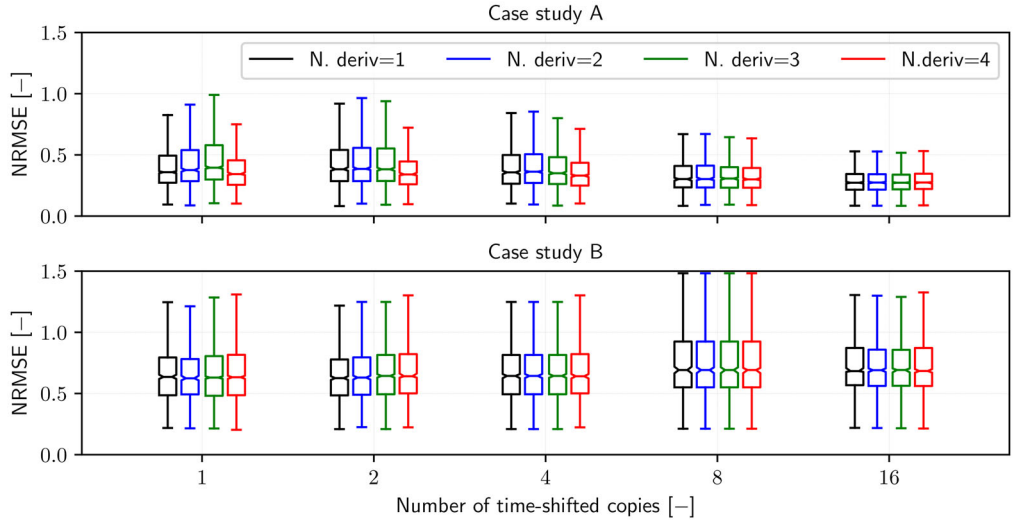

FIGURE 11 NRMSE boxplots for ADMD on the test set, conditional to the number of derivatives and time-shifted copies.

TABLE 3 ADMD performance metrics on the test set: averages (avg) and standard deviations (std).

		Number of time-derivatives	Case study A					Case study B				
			Number of time-shifts					Number of time-shifts				
			1	2	4	8	16	1	2	4	8	16
NRMSE	avg	1	0.4259	0.4733	0.4248	0.3438	0.2905	0.6554	0.6519	0.6791	0.7639	0.7425
	(std)		(0.2609)	(0.3288)	(0.2776)	(0.1803)	(0.1153)	(0.2316)	(0.2318)	(0.2578)	(0.3236)	(0.2900)
	avg	2	0.4663	0.4811	0.4336	0.3365	0.2884	0.6568	0.6686	0.6791	0.7639	0.7373
	(std)		(0.3206)	(0.3347)	(0.2987)	(0.1521)	(0.1128)	(0.2399)	(0.2659)	(0.2578)	(0.3236)	(0.2771)
	avg	3	0.5028	0.4840	0.4194	0.3355	0.2881	0.6613	0.6791	0.6791	0.7639	0.7471
	(std)		(0.3482)	(0.3527)	(0.2763)	(0.1610)	(0.1106)	(0.2471)	(0.2578)	(0.2578)	(0.3236)	(0.2771)
	avg	4	0.3953	0.3912	0.3684	0.3330	0.2914	0.6684	0.7018	0.7018	0.7639	0.7448
	(std)		(0.2446)	(0.2337)	(0.1891)	(0.1719)	(0.1133)	(0.2506)	(0.3152)	(0.3152)	(0.3236)	(0.3011)
NAMMAE	avg	1	0.3168	0.3552	0.3176	0.2581	0.2201	0.4205	0.4233	0.4396	0.5059	0.4866
	(std)		(0.1991)	(0.2579)	(0.2154)	(0.1360)	(0.0943)	(0.1997)	(0.2107)	(0.2203)	(0.3086)	(0.2489)
	avg	2	0.3480	0.3550	0.3297	0.2545	0.2198	0.4261	0.4323	0.4396	0.5059	0.4793
	(std)		(0.2369)	(0.2543)	(0.2427)	(0.1244)	(0.0934)	(0.2114)	(0.2366)	(0.2203)	(0.3086)	(0.2373)
	avg	3	0.3701	0.3571	0.3170	0.2509	0.2193	0.4292	0.4396	0.4396	0.5059	0.4888
	(std)		(0.2522)	(0.2689)	(0.2163)	(0.1223)	(0.0862)	(0.2241)	(0.2203)	(0.2203)	(0.3086)	(0.2699)
	avg	4	0.2877	0.2947	0.2767	0.2499	0.2221	0.4260	0.4553	0.4553	0.5059	0.4864
	(std)		(0.1922)	(0.1788)	(0.1438)	(0.1297)	(0.0891)	(0.2112)	(0.2850)	(0.2850)	(0.3086)	(0.2537)
R	avg	1	0.8782	0.8568	0.8758	0.9100	0.9307	0.7152	0.7195	0.6990	0.6592	0.6657
	(std)		(0.1537)	(0.1879)	(0.1618)	(0.1115)	(0.0864)	(0.2084)	(0.2173)	(0.2368)	(0.2474)	(0.2416)
	avg	2	0.8636	0.8570	0.8798	0.9131	0.9327	0.7155	0.7065	0.6990	0.6592	0.6661
	(std)		(0.1707)	(0.1777)	(0.1476)	(0.1020)	(0.0818)	(0.2172)	(0.2340)	(0.2368)	(0.2474)	(0.2428)
	avg	3	0.8371	0.8459	0.8764	0.9123	0.9332	0.7083	0.6990	0.6990	0.6592	0.6682
	(std)		(0.2176)	(0.2073)	(0.1720)	(0.1032)	(0.0797)	(0.2296)	(0.2368)	(0.2368)	(0.2474)	(0.2386)
	avg	4	0.8895	0.8933	0.8993	0.9141	0.9318	0.7045	0.6881	0.6881	0.6592	0.6686
	(std)		(0.1376)	(0.1293)	(0.1256)	(0.1084)	(0.0844)	(0.2313)	(0.2416)	(0.2416)	(0.2474)	(0.2384)
AAM	avg	1	0.7110	0.6871	0.7120	0.7583	0.7922	0.5277	0.5311	0.5138	0.4787	0.4847
	(std)		(0.1549)	(0.1740)	(0.1595)	(0.1292)	(0.0965)	(0.1709)	(0.1751)	(0.1869)	(0.1910)	(0.1847)
	avg	2	0.6898	0.6821	0.7111	0.7614	0.7922	0.5279	0.5209	0.5138	0.4787	0.4877
	(std)		(0.1686)	(0.1748)	(0.1556)	(0.1188)	(0.0958)	(0.1745)	(0.1849)	(0.1869)	(0.1910)	(0.1828)
	avg	3	0.6677	0.6709	0.7158	0.7613	0.7911	0.5236	0.5138	0.5138	0.4787	0.4877
	(std)		(0.1873)	(0.1822)	(0.1568)	(0.1209)	(0.0956)	(0.1813)	(0.1869)	(0.1869)	(0.1910)	(0.1811)
	avg	4	0.7257	0.7282	0.7393	0.7633	0.7875	0.5215	0.5066	0.5066	0.4787	0.4865
	(std)		(0.1478)	(0.1462)	(0.1362)	(0.1177)	(0.0985)	(0.1815)	(0.1902)	(0.1902)	(0.1910)	(0.1812)

Deibold-Mariano statistic (DM). At a 5% significance level (two-tailed), with a critical value of $\alpha = 1.96$, the null hypothesis of equal predictive accuracy is rejected since $|DM| > \alpha$ for all pairwise comparisons, except for FFNN versus BiLSTM in Case study A, considering NAMMAE and R metrics, and BiLSTM versus ADMD in Case study B, considering the R metric. As regards the Klomogorov-Smirnov test, the table reports the p-value and the Kolmogorov-Smirnov statistic (KS). At the 5% significance level, the null hypothesis is consistently rejected. Considering the results of both statistical tests, we can infer that the differences in the results of FFNN, BiLSTM, and ADMD are statistically significant, which further confirms that the BiLSTM model generally offers the best

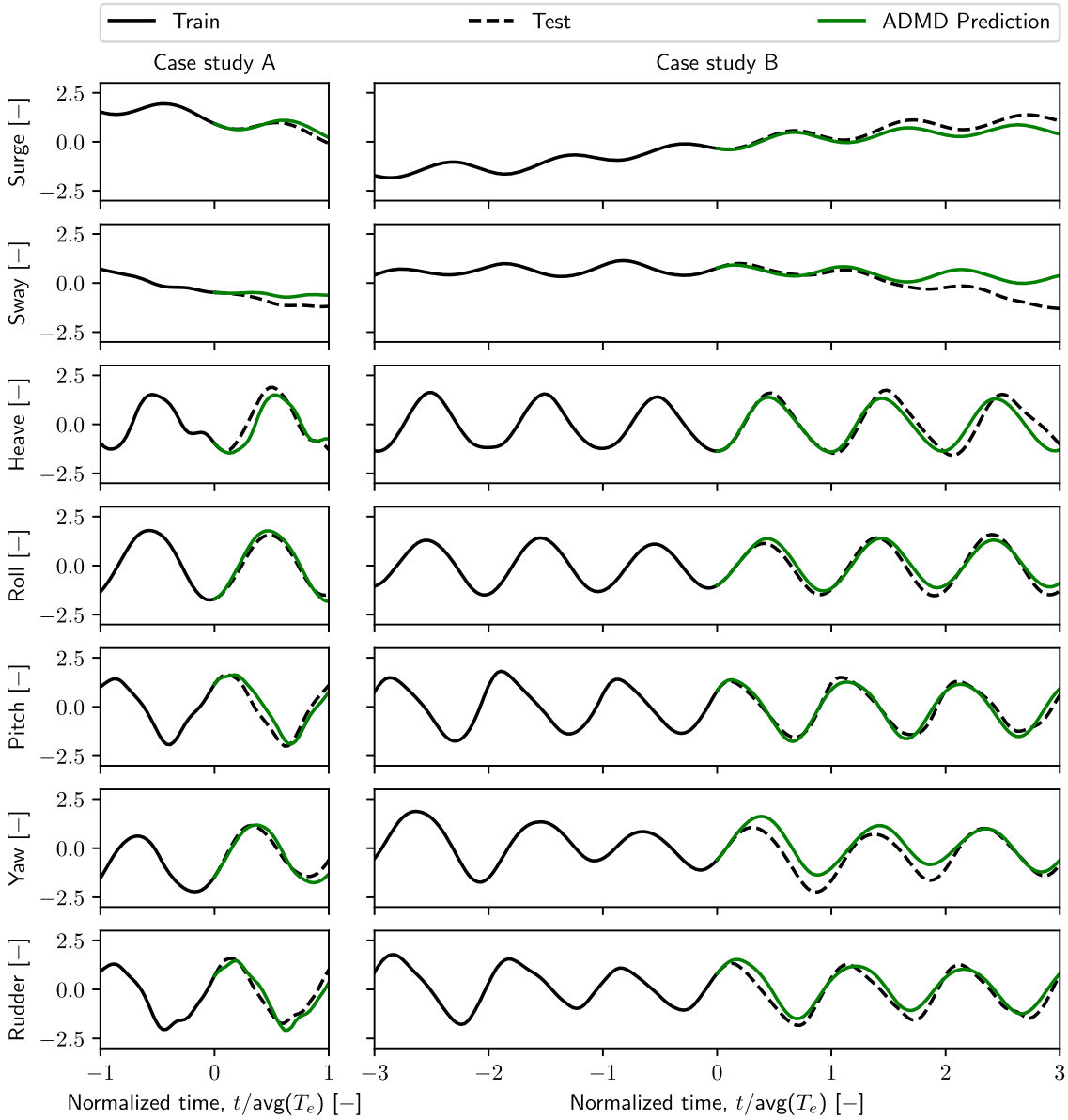

FIGURE 12 Example of prediction results provided by ADMD at a given time instant t of the six motion variables along with the rudder angle in randomly-selected instances belonging to the test set. The horizontal axis represents the normalized time, that is, the number of encounter waves from the current time instant (denoted with $t = 0$). Case study A is depicted on the left, while Case study B is reported on the right.

performance. Moreover, given that the null hypothesis was not confirmed for the R metric in two distinct cases, we might question the reliability of this metric in evaluating performance. Consequently, this metric may not be considered robust enough for inclusion in future studies.

Concerning the computational effort required to generate the predictions of the motion variables, the most demanding part is the training of both FFNN and RNN. However, it is typically performed offline, that is, before the actual operation of the system. In general, Case study B is more demanding as compared to Case study A since the former deals with longer time series. The training required up to 5 min for Case study A and 10 min for Case study B per network configuration on a laptop equipped with a 2.6 GHz Intel i7 processor with 32 GB of RAM. After the training, the generation of the estimates required a negligible amount of time both in Case study A and Case study B, as each estimate is just the result of an algebraic operation (less than 1 s was required). Similar values were experienced by ADMD (remember that such a method does not require a training procedure). On the contrary, the adoption of high-fidelity numerical solvers to generate future estimates typically requires days to run in high-performance computing platforms.

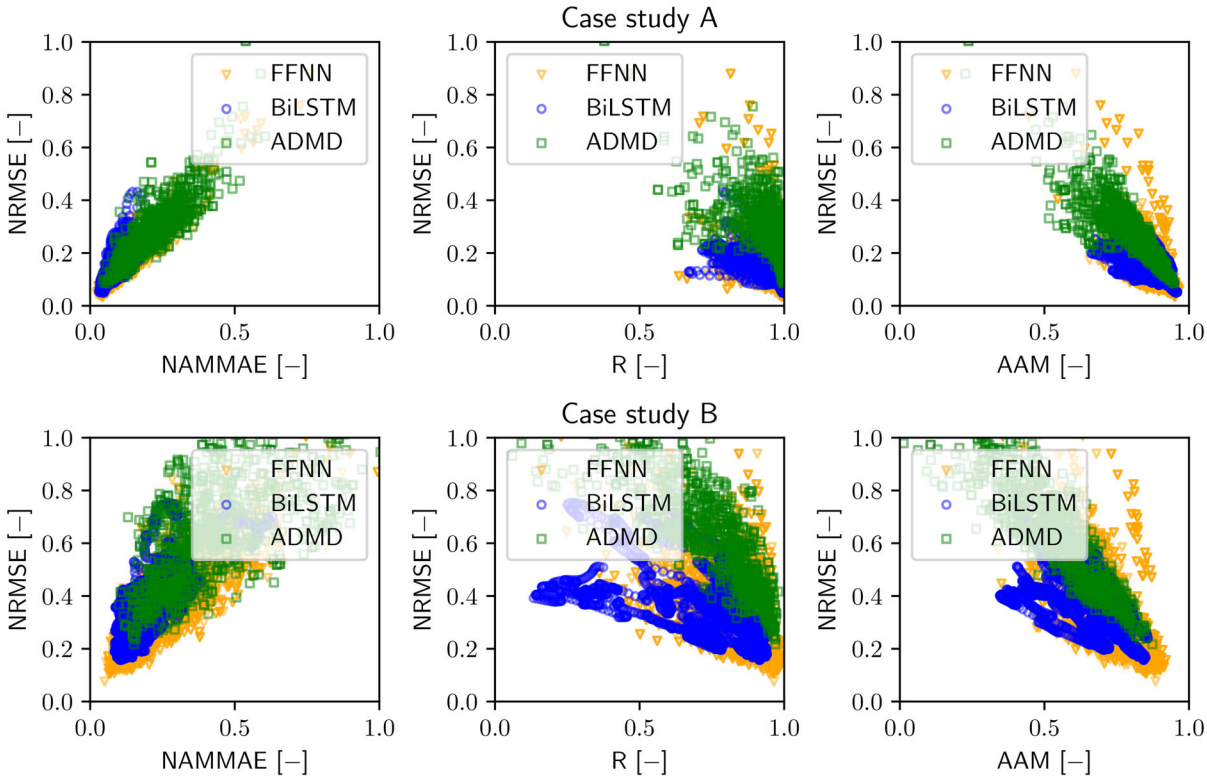

FIGURE 13 Correlation between evaluation metrics for FFNN, BiLSTM, and ADMD. Each of the forecasting approaches is in the configuration providing the best performances.

TABLE 4 Comparison of the best performances of the different data-driven models considered throughout the article in terms of averages (avg) and standard deviations (std) of the evaluation metrics.

		Data-driven model					
		FFNN		BiLSTM		ADMD	
	Case study	Avg	(Std)	Avg	(Std)	Avg	(Std)
NRMSE	A	0.1977	(0.0727)	0.1440	(0.0384)	0.2881	(0.1106)
	B	0.3829	(0.0963)	0.3462	(0.0826)	0.6519	(0.2318)
NAMMAE	A	0.1608	(0.0624)	0.0875	(0.0286)	0.2193	(0.0862)
	B	0.2824	(0.0697)	0.1885	(0.0569)	0.4233	(0.2107)
R	A	0.9500	(0.0194)	0.9525	(0.0500)	0.9332	(0.0797)
	B	0.8236	(0.0453)	0.7356	(0.1685)	0.7195	(0.2173)
AAM	A	0.8670	(0.0215)	0.8788	(0.0529)	0.7911	(0.0956)
	B	0.7215	(0.0447)	0.6654	(0.1173)	0.5311	(0.1751)

7 | CONCLUSIONS

In this article, we have investigated data-driven, equation-free forecasting of the motions of ships in waves via machine learning and reduced-order modeling approaches. In particular, we have considered FFNN and RNN as regards machine learning methods, along with ADMD, which is an appealing alternative to neural networks, not requiring time-consuming training processes and therefore allowing for real-time (or nearly real-time) learning in digital twin platforms. Simulation results in two case studies involving short- and medium-term predictions have been performed with the goal of evaluating the effectiveness of the proposed approaches to forecast ship motion variables. Both case studies involve the

TABLE 5 Results of the Diebol-Mariano (DM) and Kolmogorov-Smirnov (KS) statistical tests.

		Case study A					Case study B				
		DM test ($\alpha = 1.96$)			KS test ($p = 0.05$)		DM test ($\alpha = 1.96$)			KS test ($p = 0.05$)	
		$\bar{d}$	SE($\bar{d}$)	DM	p-value	KS	$\bar{d}$	SE($\bar{d}$)	DM	p-value	KS
NRMSE	FFNN versus BiLSTM	0.064	0.017	3.800	<1E-6	0.360	0.043	0.016	2.683	<1E-6	0.402
	FFNN versus ADMD	−0.079	0.017	−4.521	<1E-6	0.636	−0.257	0.018	−13.97	<1E-6	0.624
	BiLSTM versus ADMD	−0.143	0.005	−31.17	<1E-6	0.874	−0.300	0.011	−27.06	<1E-6	0.812
NAMMAE	FFNN versus BiLSTM	0.027	0.015	1.782	<1E-6	0.442	−0.055	0.012	−4.601	<1E-6	0.348
	FFNN versus ADMD	−0.116	0.016	−7.289	<1E-6	0.598	−0.356	0.015	−23.05	<1E-6	0.410
	BiLSTM versus ADMD	−0.199	0.005	−43.85	<1E-6	0.916	−0.455	0.011	−41.32	<1E-6	0.736
R	FFNN versus BiLSTM	−0.006	0.005	−1.185	<1E-6	0.428	0.098	0.007	13.36	<1E-6	0.642
	FFNN versus ADMD	0.017	0.006	2.844	<1E-6	0.194	0.099	0.011	9.069	<1E-6	0.224
	BiLSTM versus ADMD	0.022	0.004	6.037	<1E-6	0.258	0.001	0.010	0.116	<1E-6	0.466
AAM	FFNN versus BiLSTM	−0.014	0.004	−3.549	<1E-6	0.230	0.062	0.006	10.22	<1E-6	0.548
	FFNN versus ADMD	0.076	0.006	13.69	<1E-6	0.440	0.186	0.009	20.93	<1E-6	0.478
	BiLSTM versus ADMD	0.090	0.004	20.76	<1E-6	0.618	0.125	0.008	15.55	<1E-6	0.524

course keeping of the 5415M vessel model in stern-quartering sea state 7 irregular waves at nominal Froude number equal to 0.33.

The most promising configuration setups for FFNN, RNN, and ADMD have been identified for the considered test cases, based on a full-factorial combination of parameter settings tested against a random sample of sequences, using four evaluation metrics. The obtained results are successful, as accurate predictions of the ship motions have been obtained both in the short- and medium-term case studies with negligible computational requirements. Hence, we conclude that the use of FFNN/RNN/ADMD to forecast the motions of ships in the near future by exploiting information on the past is not only feasible but also extremely promising as regards the development of control and optimization techniques for vessels operating in waves, for which it is crucial to have at disposal fast and accurate predictions to be inserted in the corresponding algorithms. As regards in particular the used R performance metric, the results have revealed that a more comprehensive evaluation framework that incorporates a variety of metrics is needed, ensuring a holistic assessment of performance. The apparent limitations of the R metric have suggested that its utility in the assessment of predictive accuracy may be limited, warranting its exclusion in future studies.

Future efforts will be devoted to the hybridization of FFNN/RNN and ADMD results, with the aim of providing a viable approach to accurate and interpretable predictions, which will pave the way towards the development of suitable optimization and control techniques for ships operating in waves.

ACKNOWLEDGMENTS

The authors thank the financial support of the US Office of Naval Research, NICOP Grant N62909-21-1-2042, "Improving Knowledge, Prediction, and Forecasting of Ships in Waves via Hybrid Machine Learning Methods" (FORWARD), and of the Italian Ministry of University and Research through the National Recovery and Resilience Plan (PNRR), CN00000023 - CUP B43C22000440001, "Sustainable Mobility Center" (CNMS), Spoke 3 "Waterways." The authors are also grateful to NATO Science and Technology Organization, Applied Vehicle Technology task groups AVT-280 ("Evaluation of Prediction Methods for Ship Performance in Heavy Weather") and AVT-351 ("Enhanced Computational Performance and Stability & Control Prediction for NATO Military Vehicles") for the fruitful collaboration through the years. Open access publishing facilitated by Consiglio Nazionale delle Ricerche, as part of the Wiley - CRUI-CARE agreement. [Correction added on 11 September 2024, after first online publication: Projekt Deal funding statement has been added.]

ORCID

Matteo Diez https://orcid.org/0000-0001-6113-7893
Mauro Gaggero https://orcid.org/0000-0002-5048-4141

Andrea Serani https://orcid.org/0000-0002-8814-1793

REFERENCES

1. Serani A, Dragone P, Stern F, Diez M. On the use of dynamic mode decomposition for time-series forecasting of ships operating in waves. *Ocean Eng*. 2023;267:113235.
2. Silva K, Maki K. Data-driven system identification of 6-DoF ship motion in waves with neural networks. arXiv preprint arXiv:2111.01773, 2021.
3. Brunton S, Kutz J. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press; 2019.
4. Montans F, Chinesta F, Gomez-Bombarelli R, Kutz J. Data-driven modeling and learning in science and engineering. *C R Mec*. 2019;347:845-855.
5. Kutz JN, Brunton SL, Brunton BW, Proctor JL. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM; 2016.
6. Geffner H. Model-free, model-based, and general intelligence. *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. AAAI Press; 2018:10-17.
7. Hesthaven J, Ubbiali S. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *J Comput Phys*. 2018;363:55-78.
8. Fu J, Xiao D, Fu R, et al. Physics-data combined machine learning for parametric reduced-order modelling of nonlinear dynamical systems in small-data regimes. *Comput Methods Appl Mech Eng*. 2023;404:115771.
9. Fu R, Xiao D, Navon IM, et al. A non-linear non-intrusive reduced order model of fluid flow by auto-encoder and self-attention deep learning methods. *Int J Numer Methods Eng*. 2023;124:3087-3111.
10. Svozil D, Kvasnicka V, Pospichal J. Introduction to multi-layer feed-forward neural networks. *Chemom Intel Lab Syst*. 1997;39:43-62.
11. Medsker L, Jain LC. *Recurrent Neural Networks: Design and Applications*. CRC Press; 1999.
12. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*. 1997;9(8):1735-1780.
13. Siami-Namini S, Tavakoli N, Namin AS. The performance of LSTM and BiLSTM in forecasting time series. *Proceedings of the IEEE International Conference on Big Data*. IEEE; 2019:3285-3292.
14. Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. NIPS 2014 Workshop on Deep Learning; 2014.
15. Schmid P. Dynamic mode decomposition of numerical and experimental data. *J Fluid Mech*. 2010;656:5-28.
16. D'Agostino D, Serani A, Stern F, Diez M. Time-series forecasting for ships maneuvering in waves via recurrent-type neural networks. *J Ocean Eng Mar Energy*. 2022;8:479-487.
17. Jiang Z, Ma Y, Li W. A data-driven method for ship motion forecast. *J Mar Sci Eng*. 2024;12(2):291.
18. Diez M, Serani A, Campana EF, Stern F. Time-series forecasting of ships maneuvering in waves via dynamic mode decomposition. *J Ocean Eng Mar Energy*. 2022;8(4):471-478.
19. Chen CZ, Liu SY, Zou ZJ, Zou L, Liu JZ. Time series prediction of ship maneuvering motion based on dynamic mode decomposition. *Ocean Eng*. 2023;286:115446.
20. Chen CZ, Zou ZJ, Zou L, Zou M, Kou JQ. Time series prediction of ship course keeping in waves using higher order dynamic mode decomposition. *Phys Fluids*. 2023;35(9):097139.
21. Wang Z, Xiao D, Fang F, Govindan R, Pain C, Guo Y. Model identification of reduced order fluid dynamics systems using deep learning. *Int J Numer Methods Fluids*. 2018;86(4):255-268.
22. Ma Y, Larson D, Sclavounos P. Support vector machine learning model of the nonlinear viscous ship roll hydrodynamics. Proceedings of the 33rd Symposium on Naval Hydrodynamics, Osaka, Japan; 2020.
23. Xu W, Maki KJ, Silva KM. A data-driven model for nonlinear marine dynamics. *Ocean Eng*. 2021;236:109469.
24. Zhang T, Zheng XQ, Liu MX. Multiscale attention-based LSTM for ship motion prediction. *Ocean Eng*. 2021;230:109066.
25. Zhou X, Zou L, Ouyang ZL, Liu SY, Zou ZJ. Nonparametric modeling of ship maneuvering motions in calm water and regular waves based on R-LSTM hybrid method. *Ocean Eng*. 2023;285:115259.
26. Wang N, Kong X, Ren B, Hao L, Han B. SeaBil: self-attention-weighted ultrashort-term deep learning prediction of ship maneuvering motion. *Ocean Eng*. 2023;287:115890.
27. Gao N, Chuang Z, Hu A. Real-time prediction of ship motion based on improved empirical mode composition and dynamic residual neural network. *Ocean Eng*. 2024;292:116528.
28. Wei Y, Chen Z, Zhao C, Tu Y, Chen X, Yang R. A BiLSTM hybrid model for ship roll multi-step forecasting based on decomposition and hyperparameter optimization. *Ocean Eng*. 2021;242:110138.
29. Li MW, Xu DY, Geng J, Hong WC. A ship motion forecasting approach based on empirical mode decomposition method hybrid deep learning network and quantum butterfly optimization algorithm. *Nonlinear Dyn*. 2022;107(3):2447-2467.
30. Li MW, Xu DY, Geng J, Hong WC. A hybrid approach for forecasting ship motion using CNN–GRU–AM and GCWOA. *Appl Soft Comput*. 2022;114:108084.
31. Zhang D, Zhou X, Wang ZH, Peng Y, Xie SR. A data driven method for multi-step prediction of ship roll motion in high sea states. *Ocean Eng*. 2023;276:114230.
32. Serani A, Diez M, Walree F, Stern F. URANS analysis of a free-running destroyer sailing in irregular stern-quartering waves at sea state 7. *Ocean Eng*. 2021;237:109600.

33. Mhaskar H, Liao QL, Poggio T. Learning real and Boolean functions: when is deep better than shallow. Technical Report Memo No. 045. The Center for Brains; 2016.
34. Alessandri A, Cervellera C, Gaggero M. Predictive control of container flows in maritime intermodal terminals. *IEEE Trans Control Syst Technol*. 2013;21(4):1423-1431.
35. Alessandri A, Bagnerini P, Gaggero M. Optimal control of propagating fronts by using level set methods and neural approximations. *IEEE Trans Neural Netw Learn Syst*. 2019;30(3):902-912.
36. Pineda-Jaramillo J. A shallow neural network approach for identifying the leading causes associated to pedestrian deaths in Medellin. *J Transp Health*. 2020;6:100912.
37. Orimoloye L, Sung MC, Ma T, Johnson J. Comparing the effectiveness of deep feedforward neural networks and shallow architectures for predicting stock price indices. *Expert Syst Appl*. 2020;139:112828.
38. Alessandri A, Bagnerini P, Gaggero M, Mantelli L, Santamaria V, Traverso A. Black-box modeling and optimal control of a two-phase flow using level set methods. *IEEE Trans Control Syst Technol*. 2022;30(2):520-534.
39. Alessandri A, Gaggero M, Sanguineti M. Data-driven performance metrics for neural network learning. *Int J Adapt Control Signal Process*. 2025;39(10):2081-2092.
40. Barron A. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans Inf Theory*. 1993;39(3):930-945.
41. Kurková V, Sanguineti M. Comparison of worst-case errors in linear and neural network approximation. *IEEE Trans Inf Theory*. 2002;48(1):264-275.
42. Zoppoli R, Sanguineti M, Parisini T. Approximating networks and the extended Ritz method for the solution of functional optimization problems. *J Optim Theory Appl*. 2002;112(2):403-439.
43. Goodfellow I, Bengio Y, Courville A. *Deep Learning*. MIT Press; 2016.
44. Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on International Conference on Machine Learning*. JMLR.org; 2013:1310-1318.
45. Cho K, Merrienboer BV, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.
46. Lambe A, Martins J. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Struct Multidiscipl Optim*. 2012;46:273-284.
47. Kutz JN, Fu X, Brunton SL. Multiresolution dynamic mode decomposition. *SIAM J Appl Dyn Syst*. 2016;15(2):713-735.
48. Rowley CW, Mezić I, Bagheri S, Schlatter P, Henningson D. Spectral analysis of nonlinear flows. *J Fluid Mech*. 2009;641(1):115-127.
49. Magionesi F, Dubbioso G, Muscari R, Di Mascio A. Modal analysis of the wake past a marine propeller. *J Fluid Mech*. 2018;855:469-502.
50. Dogan T, Wang Z, Stern F. Experimental and numerical study of air-water interface instabilities with machine learning for experimental data analysis. 33th Symposium on Naval Hydrodynamics, Osaka, Japan; 2020.
51. Alessandri A, Bagnerini P, Gaggero M, Lengani D, Simoni D. Dynamic mode decomposition for the inspection of three-regime separated transitional boundary layers using a least squares method. *Phys Fluids*. 2019;31:1-13.
52. Alessandri A, Bagnerini P, Gaggero M, Lengani D, Simoni D. Detection of flow-regime transitions using dynamic mode decomposition and moving horizon estimation. *IEEE Trans Control Syst Technol*. 2021;29:1324-1331.
53. Walree F, Serani A, Diez M, Stern F. Prediction of heavy weather seakeeping of a destroyer hull form by means of time domain panel and CFD codes. Proceedings of the 33nd Symposium on Naval Hydrodynamics, Osaka, Japan; 2020.
54. Huang J, Carrica P, Stern F. Semi-coupled air/water immersed boundary approach for curvilinear dynamic overset grids with application to ship hydrodynamics. *Int J Numer Methods Fluids*. 2008;58(6):591-624.
55. Diebold FX, Mariano RS. Comparing predictive accuracy. *J Bus Econ Stat*. 2002;20(1):134-144.
56. Massey FJ, The J. Kolmogorov-Smirnov test for goodness of fit. *J Am Stat Assoc*. 1951;46(253):68-78.

How to cite this article: Diez M, Gaggero M, Serani A. Data-driven forecasting of ship motions in waves using machine learning and dynamic mode decomposition. *Int J Adapt Control Signal Process*. 2025;39(10):2119-2142. doi: 10.1002/acs.3835