IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 October 2023, accepted 7 December 2023, date of publication 18 December 2023,
date of current version 16 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3343874

RESEARCH ARTICLE

Computationally Aware Surrogate Models for the Hydrodynamic Response Characterization of Floating Spar-Type Offshore Wind Turbine

DAVIDE ILARDI[1], MILTIADIS KALIKATZARAKIS[2], LUCA ONETO[1], (Senior Member, IEEE), MAURIZIO COLLU[2], AND ANDREA CORADDU[3], (Member, IEEE)

[1]Department of Informatics, Bioengineering, Robotics and Systems Engineering, Università di Genova, 16124 Genova, Italy
[2]Department of Naval Architecture, Ocean and Marine Engineering, University of Strathclyde, G1 1XQ Glasgow, U.K.
[3]Department of Maritime and Transport Technology, Delft University of Technology, 2628 CD Delft, The Netherlands

Corresponding author: Andrea Coraddu (a.coraddu@tudelft.nl)

This work was supported in part by the European Union—NextGenerationEU. Piano Nazionale di Ripresa e Resilienza, Missione 4 Componente 2 Investimento 1.4: Potenziamento strutture di ricerca e creazione di "campioni nazionali di R&S" su alcune Key Enabling Technologies, and in part by the Centro Nazionale per la Mobilitã Sostenibile (CNMS) under Grant CN00000023.

ABSTRACT Due to increasing environmental concerns and global energy demand, the development of Floating Offshore Wind Turbines (FOWTs) is on the rise. FOWTs offer a promising solution to expand wind farm deployment into deeper waters with abundant wind resources. However, their harsh operating conditions and lower maturity level compared to fixed structures pose significant engineering challenges, notably in the design phase. A critical challenge is the time-consuming hydromechanics analysis traditionally done using computationally intensive Computational Fluid Dynamics (CFD) models. In this study, we introduce Artificial Intelligence-based surrogate models using state-of-the-art Machine Learning algorithms. These surrogate models achieve CFD-level accuracy (within 3% difference) while dramatically reducing computational requirements from minutes to milliseconds. Specifically, we build a surrogate model for characterizing the hydrodynamic response of a floating spar-type offshore wind turbine (including added mass, radiation damping matrices, and hydrodynamic excitation) using computationally efficient shallow Machine Learning models, optimizing the trade-off between computational efficiency and accuracy, based on data generated by a cutting-edge potential-flow code.

INDEX TERMS Floating offshore wind turbines, hydrodynamic response, computational fluid dynamics, surrogate models, machine learning, accuracy, computational requirements.

I. INTRODUCTION

It is estimated that the global electricity consumption will reach 31,657 TWh by 2030 [1]. Consequently, increasing environmental concerns pushed the governments to reduce by 40% the CO_2 emissions and to increase by 25% in energy efficiency for the same date [2]. Renewable energy sources are currently the only alternative to fossil fuels with the real potential to achieve this goal [3]. Floating wind is one of the fastest-growing sectors within the Offshore renewable energy industry and internationally recognized as one of the most promising renewable energy sources to satisfy a significant proportion of global energy demands [4]. The ability to economically deploy Floating Offshore Wind Turbines (FOWTs) in deep-water areas, that were previously unfeasible for development using fixed-bottom turbines, is one of the fundamental driving forces behind the success of floating wind [5]. In fact, deep-water areas are often characterized by higher average wind speeds and consequently higher average capacity factors that could improve the economic viability of offshore wind energy [6]. However, floating wind is still an emerging market, and only

The associate editor coordinating the review of this manuscript and approving it for publication was Turgay Celik.

a limited number of pilots have been deployed. In fact, there are still significant engineering challenges in the design and construction of FOWTs [7], [8], [9].

One of the most critical challenges is the assessment of the technical and economic feasibility of alternative FOWT designs to ensure the best trade off between manufacturing and maintenance costs and energetic performance [10], [11]. In particular, there is a need for the development of accurate modelling tools to facilitate the complex and iterative design and optimization processes of the FOWTs [12]. In fact, current FOWT designs are largely based on the ones exploited for onshore applications [8]. This occurs due to the lack of accurate and computationally efficient modelling tools during the design phase [13] as it happens in many other applications [14], [15], [16], [17], [18], [19].

This issue has been clearly identified by both academia and industry [20], [21]. Traditional numerical methods based on Computational Fluid Dynamics (CFD), while being very accurate, are computationally demanding at a level that prevents their use in iterative design and optimization processes [22]. For this reason, many studies have focused on reducing the computational requirements of CFD models by means of surrogate models in many fields of research [23], [24], [25], [26], [27] but also for FOWT design and analysis [21], [28], [29], [30], [31], [32], [33], [34], [35], [36]. All these research studies point out that surrogate models can be a very promising approach for replacing CFD for FOWTs design and optimization. In fact, all these studies show how surrogate models can substantially reduce the computational requirements, with minimal sacrifice in terms of quality of the obtained solutions.

A fundamental part of the conceptual and preliminary design of a FOWT is the assessment of its hydromechanics characteristics and more precisely the added mass matrix, the radiation damping matrix and the hydrostatic restoring matrix, as functions of the frequency of oscillation [37]. Current state-of-the-art highly-accurate approaches for this analysis are based on CFD and higher order boundary element methods [38], [39], [40]. However, their high computational requirements prevent the exploration of new and unconventional designs [22]. Despite the efforts to incorporate surrogate models for FOWT hydromechanics characteristics modelling [28], [29], [30], their application is still in the infancy stage.

For this reason, in this work, building upon the authors' previous work [41] on surrogate model for Response Amplitude Operators (RAOs), we will focus on the prediction of the hydromechanics characteristics. Contrarily to the RAOs, the hydromechanics characteristics depend only on the wet geometry of the platform, on the wave frequency and, for the wave loads transfer functions, on the wave direction, i.e., they do not depend on the mass, nor on the center of gravity or moments of inertia, which can be usually obtained with little computational cost [42]. This allows the use of the surrogate model for different structural mass distributions, greatly enhancing its applicability. In order to develop our surrogate models we will rely on Data-Driven Models (DDMs), i.e., models built based on observation (examples) of the inputs, and associated outputs, of a possibly unknown system without any prior knowledge about it [43]. These models require as many examples as possible and a large amount of computational resources to be constructed. More precisely, the more examples are available, the more accurate the solution is [43] even if, in some cases, a small amount of examples could be sufficient for solving real world problems [44], [45]. For what concerns the model construction, to train and optimize the performance of a data-driven based surrogate model, a large amount of computational power is always needed [43], [46], [47]. Instead, once the model is constructed, its use for making predictions (i.e., the forward phase) is mostly computationally inexpensive [43], [46]. For the scope of this paper, we will compare the performances achieved by now-classical Machine Learning (ML) models (i.e., Kernel Methods and Ensemble Methods) with the ones resulting from the exploitation of Deep Neural Networks (DNNs) [46]. The best performing DDM will be selected based on the best trade-off between accuracy and efficiency.

In order to validate our approach we will rely on data generated by NEMOH, a state-of-the-art code developed by researchers at École Centrale de Nantes [48], which calculates the FOWTs hydromechanics characteristics (e.g., the added mass matrix, the radiation damping matrix and the hydrostatic restoring matrix, as functions of the frequency of oscillation), by means of a Boundary Element Method (BEM). In particular, we will consider a series of possible spar type FOWT geometries exploiting a simple parametrization approach. The proposed parametrization technique will allow us to explore a richer set of geometries than the conventional one. This could lead to the definition of novel substructure geometry, capable of enhancing the performance of the FOWT and eventually lowering the cost of electricity produced. For each of these geometries, using a Monte Carlo sampling approach [49], NEMOH has been run, producing the desired outputs. The generated datasets have been used to train, validate, and estimate the performance [47] of the proposed surrogate model, showing that it is possible to reach the best trade-off between computational requirements and accuracy, i.e., comparable accuracy of BEM at a fraction of the computational requirements.

Before starting the presentation, we would like to underline the novelties of this paper and its possible impacts. To the best knowledge of the authors, this work is the first one that shows the ability of ML-based models to over-performing state-of-the-art BEM-based tools in terms of computational requirements (from minutes to milliseconds) without compromising the ability to make accurate predictions (ML predictions are less than 3% far away from the BEM predictions). In order to achieve this goal, authors have exploited the most recent results and techniques coming from

TABLE 1. List of acronyms utilised in this work.

Acronym	Description
ANN	Artificial Neural Network
BEM	Boundary Element Method
CFD	Computational Fluid Dynamics
CoB	Centre of Buoyancy
CoG	Centre of Gravity
DDM	Data-Driven Model
DNN	Deep Neural Networks
DTU	Technical University of Denmark
EE	Error Estimation
FOWT	Floating Offshore Wind Turbines
IEC	International Electrotechnical Commission
ML	Machine Learning
MS	Model Selection
NREL	National Renewable Energy Laboratory
OC3	Offshore Code Comparison Collaboration
PCE	Polynomial Chaos Expansion
RANS	Reynolds-Averaged Navier Stokes
RAO	Response Amplitude Operator
RBF	Radial Basis Function
SWE	Stuttgart Wind Energy

the ML field of research. Moreover, this paper is also the first one that actually created a real dataset from a state-of-the-art BEM code to test the quality of the proposed models. Finally, the impact of this work is substantial, as a matter of fact, this is a fundamental step toward a framework for FOWTs geometry optimization, with the capability of automatizing the design process, reducing the human intervention to a minimum level, and allowing the generation of unconventional and previously unexplored geometries.

The rest of the paper is organized as follows. Section II summarises the related works. Section III gives an overview of the state-of-the-art hydromechanics analysis. Section IV describes the problem that we want to address in this work and the developed sampling methodology to generate the geometries and the datasets utilized to train, validate, and estimate the performance of the proposed surrogate model. Section V describes the proposed data-driven based surrogate models, their properties, and the reasons behind the proposed approach. Section VI discusses the results, both in terms of accuracy and computational requirements, of exploiting the surrogate model proposed in Section V by means of the data generated in Section IV. Section VII concludes the paper. To improve the readability of the paper, we reported in Tables 1 and 2 the summary of the acronyms and symbols exploited in this work.

II. RELATED WORKS

Surrogate models have been used by many different works in many different ways and for many different purposes in designing and analyzing FOWTs. This section will briefly review the most important works according to their bibliographic impact. These works are also summarised in Table 3, which reports each work's original and proposed surrogate model. For each surrogate model, it further reports the method exploited, its inputs and outputs, the data exploited to build it, its final accuracy, and its computational requirements.

In detail, authors of [28] developed a surrogate model to approximate lifetime fatigue loads based on Kriging and Polynomial Chaos Expansion (PCE). They performed 62500 aero-elastic simulations with the software packages FAST [50] and TurbSIM [51] on the NREL 5MW reference turbine [52] utilizing wind data from 99 international sites. They considered the main components of the turbine, namely, the blades, the drivetrain, the yaw bearing, and the tower. They developed surrogate models that are based on wind speed, turbulence, wind shear exponent, air density, and flow inclination to evaluate the fatigue loads. The authors concluded that the Kriging with a second order trend, in conjunction with the Matern 3/2 correlation model, is actually able to predict fatigue loads effectively. PCE, instead, did not show consistently acceptable performance.

Authors of [29] investigated the impacts of Platform Mounting Orientation on the lifetime dynamic performance of Y-shaped semi-submersible FOWTs, utilizing Kriging, Radial Basis Functions (RBF), and Artificial Neural Networks (ANNs). The latter was established based on a database containing 25 years of met-ocean data, clustered using the K-means algorithm and self-organizing maps, and time-domain simulations. The surrogate models proved to be capable of efficiently predicting the systems' lifetime dynamic responses. In particular, exploiting a set of selected environmental load cases, authors observed that the ANNs-based model could make accurate predictions on the platform and tower peak motions, while predictions on the tower base and fairlead fatigue were less accurate. Kriging and RBF, instead, showed high accuracy on most of the reference data.

Authors of [21] proposed to exploit ANNs for FOWTs fatigue assessment. In particular, a combination of the DTU 10MW reference turbine [53] with the SWE TripleSpar developed by the authors of [54] was investigated. The authors generated synthetic data by means of 600 runs of software FAST sampling the environmental conditions space with the Latin Hypercube Sampling algorithm. In particular, they consider the variation of four environmental conditions: wind speed, turbulence intensity, wave height, and wave period. Authors of [21] observed that the predictions of the ANNs were not entirely satisfying due to the quality of the available data and argued that a data source that can provide higher quantity and quality data would significantly enhance their results.

Authors of [30] investigated Surrogate Monte Carlo simulations implemented by least-squares fits and collocation methods as alternatives to Physical Model Monte Carlo simulations for stochastic flutter analysis of wind turbine blades. They argue that finite-element methods are the state-of-the-art approach for this type of analysis in terms of accuracy. However, the associated computational requirements are quite high due to the complex aerodynamic

TABLE 2. List of symbols utilized in this work.

Symbol	Description	Symbol	Description
A	Added mass matrix	r_i	i-th decision rule of each k tree
$\mathscr{A}_{\mathcal{H}}$	ML algorithm characterised by $\mathcal{H}$	s_i	Structure of each g_i regression tree
B	Radiation damping matrix	T	Draft of the FOWT substructure
b	Parameter of f in Kernel methods	$\mathcal{T}_k$	Space of k possible regression trees
$\boldsymbol{b}_i$	Bias vector of layer i of an ANN	$\mathcal{T}_m$	Test set consisting of m samples
b_s	Batch size of ANN training process	$\mathcal{T}_t^r$	Test set with n_r replacement
C	Total restoring matrix	W_i	Weight matrix of layer i of an ANN
C^{H}	Hydrostatic restoring matrix	$\boldsymbol{w}$	Parameter of f in Kernel methods
C^{G}	Gravitational restoring matrix	$w_{i,j}$	Score of each i-th tree and its j-th leaf
C^{M}	Mooring stiffness matrix	$\mathcal{V}_v^r$	Validation set with n_r replacement
c	Number of truncated cones in the FOWT substructure	X	First order wave load transfer function vector
$\mathcal{D}_n$	Dataset consisting of n samples	$\mathcal{X}$	Input space
d	Dimensionality of the input space	$\boldsymbol{x}$	Single sample from $\mathcal{X}$
e_i	Number of leaves of each g_i tree	$\mathcal{Y}$	Output space
$\mathcal{F}$	Set of models to be selected by $\mathscr{A}_{\mathcal{H}}$	$\boldsymbol{y}$	Single sample from $\mathcal{Y}$
f	model to be selected by $\mathscr{A}_{\mathcal{H}}$	z_{G}	Vertical position of FOWT CoG
f^*	Best model to be selected by $\mathscr{A}_{\mathcal{H}}$	$z_{\mathrm{G}}^{\mathrm{WT}}$	Vertical position of wind turbine CoG
g_i	i-th tree belonging to $\mathcal{T}_k$ space	$z_{\mathrm{G}}^{\mathrm{SD}}$	Vertical position of structural CoG
g_t	Regression tree model at t-th iteration	$z_{\mathrm{G}}^{\mathrm{SB}}$	Vertical position of ballast material CoG
g_t^*	Best tree model at t-th iteration	$\boldsymbol{\alpha}$	Kernel Methods' set of parameters
H	Complex response transfer function vector	β	Kernel Methods regulariser's hyperparameter
$\mathcal{H}$	Hyperparameters of $\mathscr{A}_{\mathcal{H}}$	γ	Hyperparameter (Gaussian Kernel) and minimum loss reduction to split (Ensemble Methods)
$\mathcal{H}^*$	Best hyperparameters of $\mathscr{A}_{\mathcal{H}}$	ϵ	Mean Epsilon Insensitive Error metric's hyperparameter
h_i	Number of neurons in layer i of ANN	η	Wave amplitude and learning step (Ensemble Methods)
$\mathtt{K}$	Kernel function	θ_i	Fractional value $\in [0, 1]$ determining W_i's dropout
k	Number of regression trees	ι	Balancing parameter
$\mathcal{L}_l^r$	Learning set with n_r replacement	$\boldsymbol{\varkappa}$	Maximum number of leaves of tree
l	Number of layers of an ANN	λ	Hyperparameter regulating the over- and under-fitting tendency of f
l_r	Learning rate of ANN training process	μ	Unknown rule to be estimated
$\mathtt{M}$	Error Metric in approximating μ	ϱ_f	Features' fraction to train the trees
M	Total mass matrix	ϱ_s	Samples' fraction to train the trees
M^{WT}	Mass matrix of the wind turbine	$\boldsymbol{\sigma}_i$	ANN i-th layer's non-linear activation
M^{SD}	Structural mass matrix	τ	Maximum depth of the regression tree
M^{SB}	Ballast material matrix	$\boldsymbol{\phi}$	Non-linear mapping
n_r	Number of resampling	ξ	Displacement
q	Dimensionality of the output space	ω	Wave frequency
$\mathtt{R}$	Regulariser	$\boldsymbol{r}_i$	ANN i-th layer's representation

shape and structural layout of a wind turbine blade. For this reason, they propose surrogate models to study stochastic blade flutter. By considering the NREL 5MW reference blade [52] as a case study, the authors derived exact results using Physical Model Monte Carlo to verify the robustness of the results obtained by Surrogate Monte Carlo simulations. They concluded that the resulting probability density function of the critical flutter speed of the blade from the Surrogate Monte Carlo simulations was almost identical to the Physical Model Monte Carlo simulations. In particular, average errors below 1% have been achieved with a reduction of an order of magnitude of the computational requirements.

Authors of [31] propose a methodology to implement sparse polynomial surrogates for aeroelastic wind turbine models utilizing PCE to characterize the energy production and lifetime equivalent fatigue loads of the DTU 10MW reference wind turbine [53] under realistic turbulent inflow conditions. These conditions were defined based on probability distributions for the 10-minute mean and standard deviation hub height wind speed and the 10-minute mean shear exponent and yaw miss-align. They focused on estimating the 10-minute mean power production and mean thrust coefficient and several bending moments on the blade and tower of the wind turbine. By performing a total of 14000 simulations with the software HAWC2 [55], they captured the variability caused by different turbulent inflow fields. Then, they developed independent surrogate models for the mean and standard deviation of each estimated target. They underline that PCE methods are highly efficient in estimating the statistical properties of the considered targets and can be a very efficient alternative with respect to the traditional Physical Model Monte Carlo simulations.

Authors of [32] employed a Kriging-based surrogate to optimize a fixed-pitch and fixed-speed wind turbine blade's performance. The authors utilized as design variables the chord, twist, and three different airfoil profiles (described by their Bezier curves), maximizing the annual energy production. A similar study was performed by the authors of [20] utilizing a third-order Response Surface Model to reduce the computational requirements of a complex

TABLE 3. Related works.

Work	Original Model	Surrogate Model					
		Method	Input	Output	Available Data	Accuracy	Computational Requirements
[28]	FAST, TurbSIM	PCE, Kriging	wind speed, turbulence, and shear exponent, air density, flow inclination	lifetime fatigue load	62500 simulations	Kriging: $\leq$ 1%, PCE: $\leq 4\%$	Not addressed.
[29]	FAST	ANN, Kriging	mean wind speed, wind direction, significant wave height, peak wave period and wave direction	Long-term dynamic response	representative cases from 25 years of met-ocean data, $2,928$ simulations	NRMSE [-]: $0.04 - 0.13$ for ANN, $0.03 - 0.06$ for RBF, $0.04 - 0.09$ for Kriging	Training time: 1300 [sec] for RBF, 750 [sec] for Kriging, 30 [sec] for ANN
[21]	FAST	ANN	wind speed, turbulence intensity, wave height, and wave period	Damage equivalent load	600 simulations	$2.5 \times 10^7 - 4.5 \times 10^7$ [kNm] depending on the load case considered	Not addressed
[30]	In-house computational code based on the Euler-Bernoulli beam theory	least-squares fits, collocation methods	flow forces, lift coefficient, torsional rigidity	critical flutter speed	up to 10^4 simulations	average errors of $1\% - 3.7\%$ based on the number of simulations	1.47% w.r.t physical model
[31]	HAWC2	PCE	10-minute mean wind speed, and standard deviation, mean shear exponent, mean yaw misalignment	10-minute mean power production, mean thrust, 5 bending moments	14000 simulations	maximum errors between $1\% - 7\%$	Not addressed
[32]	In-house developed code based on Blade Element Momentum Theory	Kriging	chord, twist, airfoil profile (by Bezier curves)	annual energy production	420 simulations	maximum error: 4%	Not addressed
[33]	MIRAS	RBF	chord, twist profiles	power coefficient	20000 simulations	R^2: 0.881, NRMSE: 0.0679	MIRAS: 3 days, RBF: $\approx$ 1 minute
[34]	Star CCM+	Kriging	blade profile (7-control-point Bezier curve)	nominal power production, efficiency, blade weight	800 simulations	average errors between $2\% - 6\%$	Not addressed
[35]	In-house developed RANS solver	Kriging	blade profile (10 Hicks-Henne bump functions)	pressure coefficient	100 simulations	error ranging between 1.2% - 3.5%	Not addressed
[36]	ANSYS Fluent	ANN	blade profile (6-point Bezier curve)	lift to drag force ratio	280 simulations	Not specified	50% reduction of computational time w.r.t to CFD-based optimisation

two-step multi-objective optimization problem for blade design. In this work, authors utilized as decision variables the blade diameter, design rotational speed, chord length, and twist angle.

Authors of [33] developed a surrogate optimization methodology utilizing RBF. A database of synthetic data was created using three three-dimensional viscous-inviscid interaction code MIRAS [56] for the aerodynamic design of wind turbine rotors. The Latin Hypercube sampling algorithm was employed to generate a grid in the design-space parameterized based on blade chord and twist. A custom optimisation scheme was developed, combining both global and local search. The authors concluded that surrogate-based optimization can achieve significant savings in computational requirements with accuracy comparable to the conventional optimization methods.

Many older works [34], [35], [36] employed Response Surface Models to approximate the results of computationally expensive numerical simulations of the Navier-Stokes equations to design airfoil profiles. In particular, authors of [34] propose to parameterize blade profiles with a 7-control-point Bezier curve built based on Bernstein functions. They first employed a full-factorial Design of Experiments utilizing the Star CCM+ software [57] to estimate the response surface of each objective function, namely, nominal power production, efficiency, and blade weight. Then, by means of a meta-heuristic optimization method, they obtained the optimal blade profile. Similarly, authors of [35] proposed to approximate the Navier-Stokes equations within a design optimization framework. The authors first exploited Kriging and the Latin Hypercube sampling algorithm to construct a surrogate model that approximates the computational results of an in-house developed Reynolds-Averaged Navier Stokes (RANS) solver. Then they use this surrogate model to minimize the drag force of the blade with a Genetic Algorithm. Analogously, authors of [36] propose first to approximate the Navier-Stokes equations by means of ANNs. Then, the Genetic Algorithm was exploited to maximize the lift-to-drag force ratio of a wind turbine blade. They parameterized the blade geometry on the basis of 6-point Bezier curves. The Genetic Algorithm was initially run using the time-intensive CFD-based code ANSYS Fluent [58] for 10 generations. Then, they built a surrogate of the CFD based on the points explored by the Genetic Algorithm. Finally, they run again the Genetic Algorithm using the surrogate instead of the CFD-based code.

III. HYDROMECHANIC ANALYSIS

The hydromechanics analysis of FOWTs is a fundamental and time-consuming part of the design process, aimed at characterizing the dynamic response of the offshore system to the hydrodynamic loads. Hydromechanics analysis estimates the dynamic response of a floating offshore system at a certain sea state.

This dynamic response, in the frequency domain, is described by the RAO and depends on two main aspects of the offshore system, linked to two analyses: mass distribution analysis to derive the system's center of gravity and mass matrix, and hydrodynamics analysis, to derive the center of buoyancy and hydrostatic and wave loads.

In order to quantify these two aspects, Section III-A describes the geometry parametrization scheme required, able to identify each configuration uniquely. Then, the methodology adopted to estimate the center of gravity and mass matrix is defined in Section III-B together with the approach adopted to perform the hydrodynamics analysis in Section III-C. Finally, Section III-D shows how the RAOs are derived from the mass distribution and hydrodynamics analysis results.

A. FOWT SUBSTRUCTURE GEOMETRY PARAMETRIZATION

Differently from the complex geometries adopted in naval architecture, floating offshore structures and FOWT adopt relatively simple shapes, i.e., rectangular cuboids and cylinders. Therefore, the parametrization schemes adopted for FOWT substructures analysis and optimization are typically very simple, often involving just cylinders [59], [60], [61]. A slightly more advanced approach has been proposed by [62] for SPAR-type FOWTs where, instead of cylinders with varying heights and radii, truncated cones have been adopted.

In the present work, a similar approach to the one reported in [62] has been implemented. As reported in [62], although relatively simple, this approach can be quite effective in exploring a richer set of geometries than the conventional approaches based on simple cylinders. This could potentially lead, during the design and optimization phases, to the definition of a novel substructure geometry, capable of enhancing the performance of the FOWT, and eventually to lower the cost of the electricity produced [63].

The FOWT substructure is therefore composed of c truncated cones, each one with an upper radius equal to r_{c-1} and a lower radius equal to r_c. The total draft of the substructure is equal to T, and each cone has the same height, equal to $^{T}/_{c}$. A viable alternative would be to consider the cone height as a parameter. However, the same result can be obtained using a larger value of c, and this solution should be preferred to the previous one since it would result in an over-parametrization of the given geometry, resulting in a huge number of configurations corresponding to the very same structure.

Consequently, the vector describing the considered geometry is defined by the following design vector

$$[c, r_0, r_1, \cdots, r_c, T]. \tag{1}$$

B. MASS DISTRIBUTION ANALYSIS

According to the technical specification of the International Electrotechnical Commission (IEC TS 61400-3-2:2019), a FOWT system can be divided into two main subsystems: the rotor-nacelle assembly and the support structure. According to the same technical specification, the support structure can be divided into the tower, floating sub-structure, and mooring system and anchors.

In the present work, the rotor-nacelle assembly and tower mass moments of inertia and centers of gravity have been considered constant for all the analyzed floating sub-structure configurations and are assumed equal to the rotor-nacelle assembly and tower of the Offshore Code Comparison Collaboration (OC3) Spar FOWT reference system [64]. The rotor-nacelle assembly and tower are indicated as wind turbine, therefore $\mathsf{M}^{\mathrm{WT}} \in \mathbb{R}^{6\times6}$ is the mass matrix of the wind turbine, and $z_G^{\mathrm{WT}} \in \mathbb{R}$ is the vertical position of its Centre of Gravity (CoG).

The mooring system and anchor mass moments of inertia and CoGs are not directly considered since these are flexible bodies [42]. On the contrary, the mooring line tension forces acting on the substructure are considered and modelled through a constant stiffness matrix when deriving the RAOs [42].

Therefore, the mass, moments of inertia, and vertical CoG position of the floating substructure system depend only on the design vector, as previously reported in Section III-A. The floating sub-structure mass is the sum of its dry mass $\mathsf{M}^{\mathrm{SD}} \in \mathbb{R}^{6\times6}$, which corresponds to the structural mass, and the mass of the ballast material $\mathsf{M}^{\mathrm{SB}} \in \mathbb{R}^{6\times6}$, with corresponding vertical CoG positions $z_G^{\mathrm{SD}} \in \mathbb{R}$ and $z_G^{\mathrm{SB}} \in \mathbb{R}$, respectively. Indeed, a SPAR FOWT system mainly relies on lowering as much as possible the vertical position of the total CoG to fulfill its static stability requirements [65], and this is achieved by using a ballast system at the bottom of the floating substructure. It is, therefore, necessary to determine the substructure dry mass and ballast mass matrices and the vertical positions of their CoGs.

The classical approach exploited to quantify M^{SD} is based on a simplified methodology, suitable for the early design phases [66]: it is assumed to be a fraction of the displaced mass of water, which can be derived from the design vector. Therefore, each truncated cone composing the floating sub-structure has a dry mass equal to a fraction of the water's displaced mass. The local center of gravity and moments of inertia of the truncated cone are then calculated using the formulas in [67]. M^{SD} and $z_{\mathrm{G}}^{\mathrm{SD}}$ are then calculated considering the weighted contribution of all the truncated cones.

Finally, assuming that the ballast tank is cylindrical, M^{SB} and $z_{\mathrm{G}}^{\mathrm{SB}}$ are derived by using the design draft, imposing the vertical equilibrium of forces to determine the ballast mass. In particular, the sum of the rotor-nacelle assembly, tower, floating substructure dry mass, and ballast weight forces, plus the vertical downward component of the total mooring force, must be equal to the buoyancy force.

Therefore, the total mass matrix M and the vertical position of its center of gravity position z_{G} can be defined as

$$\mathsf{M} = \mathsf{M}^{\mathrm{WT}} + \mathsf{M}^{\mathrm{SD}} + \mathsf{M}^{\mathrm{SB}}, \tag{2}$$

$$\mathsf{z}_{\mathrm{G}} = \frac{1}{\mathsf{M}_{3,3}} \left(z_{\mathrm{G}}^{\mathrm{WT}} \mathsf{M}_{3,3}^{\mathrm{WT}} + z_{\mathrm{G}}^{\mathrm{SD}} \mathsf{M}_{3,3}^{\mathrm{SD}} + z_{\mathrm{G}}^{\mathrm{SB}} \mathsf{M}_{3,3}^{\mathrm{SB}} \right). \tag{3}$$

C. HYDROSTATIC, STIFFNESS, AND HYDRODYNAMICS ANALYSIS

In this section, we briefly describe the approach used to calculate the FOWTs hydrodynamics characteristics, namely, the total restoring matrix $\mathsf{C} \in \mathbb{R}^{6\times6}$, the added mass matrix $\mathsf{A} \in \mathbb{R}^{6\times6}$, the radiation damping matrix $\mathsf{B} \in \mathbb{R}^{6\times6}$ and the first order wave load transfer function vector $\mathsf{X} \in \mathbb{R}^{6\times1}$.

Given the design vector, uniquely identifying the wet geometry of the sub-structure, the vertical position of the Centre of Buoyancy (CoB) $z_{\mathrm{B}}^{\mathrm{SS}}$ can be easily calculated as the weighted center of volume of the single truncated cones composing the sub-structure, and the center of volume of each truncated cone can be calculated using the formulas in [67]. The longitudinal and lateral positions of the CoB are zero due to the axisymmetry of the wet geometry. Then, knowing the total mass and CoG of the whole system and the CoB, the sum of the hydrostatic restoring matrix $\mathsf{C}^{\mathrm{H}} \in \mathbb{R}^{6\times6}$ and gravitational restoring matrix $\mathsf{C}^{\mathrm{G}} \in \mathbb{R}^{6\times6}$ can be derived according to [42]. By adding the mooring stiffness matrix C^{M}, we obtain the total stiffness matrix as

$$\mathsf{C} = \mathsf{C}^{\mathrm{H}} + \mathsf{C}^{\mathrm{G}} + \mathsf{C}^{\mathrm{M}}. \tag{4}$$

A, B, and X are, instead, computationally demanding to be quantified, and a BEM is usually adopted to solve the partial differential equation characterizing the radiation and diffraction potential flow problems.

1) BOUNDARY ELEMENT METHOD

The software NEMOH is an open-source BEM code utilized for hydromechanics analysis developed by researchers at Ecole Centrale de Nantes [48]. As mentioned before, NEMOH has been used to estimate A and B as function of the wave frequency ω, while X as functions of ω and the wave direction.

In general, 6×6 matrices are necessary to capture the impact of each wave load (in surge, sway, and heave) and moments (in roll, pitch, and yaw) on the platform's 6 rigid-body degrees of freedom. Although the rotor-nacelle assembly mass distribution is not axisymmetric, it constitutes only a minor fraction of the total mass. Therefore, exploiting the axisymmetric geometry of the FOWT considered configuration [42], the whole-system FOWT mass distribution can be considered approximately axisymmetric. This allows to consider only a fraction of the 36 coefficients and only the wave transfer loads in one plane (longitudinal), i.e., surge X_1 and heave X_3 forces, and pitch moment X_5.

In the present work, the coupled effects have not been considered and therefore only $\mathsf{A}_{1,1}$, $\mathsf{A}_{3,3}$, and $\mathsf{A}_{5,5}$, as well as $\mathsf{B}_{1,1}$, $\mathsf{B}_{3,3}$, and $\mathsf{B}_{5,5}$ are modelled. Nonetheless, the approach can be easily extended [68].

D. FOWT DYNAMIC RESPONSE: EVALUATING THE RAO

The six simultaneous equations of motion for a floating body in regular waves, as shown in [42], can be written as

$$\sum_{j=1}^{6} \xi_j \left[-\omega^2 \left(\mathsf{M}_{i,j} + \mathsf{A}_{i,j} \right) + i\omega \mathsf{B}_{i,j} + \mathsf{C}_{i,j} \right] = \eta \mathsf{X}_i, \tag{5}$$

where $i \in \{1, \cdots, 6\}$, ξ_j is the j-th degrees of freedom displacement (rigid platform global response), and η is the wave amplitude.

These are six simultaneous linear equations of motion, which can be solved to obtain the body displacement in the j-th degree of freedom

$$\xi_j = \eta \sum_{i=1}^{6} \left[-\omega^2 \left(\mathsf{M}_{i,j} + \mathsf{A}_{i,j} \right) + i\omega \mathsf{B}_{i,j} + \mathsf{C}_{i,j} \right]^{-1} \mathsf{X}_i. \tag{6}$$

The complex response transfer function between the amplitude of the wave and the amplitude of the oscillation of the system in the j-th degree of freedom is therefore

$$\mathsf{H}_j = \frac{\xi_j}{\eta} = \sum_{i=1}^{6} \left[-\omega^2 \left(\mathsf{M}_{i,j} + \mathsf{A}_{i,j} \right) + i\omega \mathsf{B}_{i,j} + \mathsf{C}_{i,j} \right]^{-1} \mathsf{X}_i. \tag{7}$$

Finally, the RAO, being a function of the wave frequency, is composed of the RAO magnitude and RAO phase. The RAO magnitude in the j-th degree of freedom, modelled through the surrogate model, is defined as the complex magnitude of the transfer function H_j according to

$$\mathsf{RAO}_j = |\mathsf{H}_j|. \tag{8}$$

IV. PROBLEM DEFINITION AND DATASET GENERATION

The scope of this section is twofold. The first one is to formally describe the proposed framework to perform the hydromechanics analysis replacing the computationally expensive NEMOH code (Section III-C1) with data-driven based surrogate models (Section V). The second one is to describe our procedure to generate the data necessary for building a surrogate model. In particular, we generated a set of representative geometries following the parametrization described in Section III-A and estimated the hydromechanics characteristics for each of these geometries.

Let us start with the description of the proposed framework.

In order to predict the RAO, three steps are required. The first one is to perform the mass distribution analysis to estimate total mass M and the center of gravity z_{G}. These tasks are computationally inexpensive (see Section III-B). The second step is to perform hydrostatic and total stiffness analysis (Step 2b) and hydrodynamic analysis (Step 2a). Also, the hydrostatic and total stiffness analysis is computational inexpensive (see Section III-C) and, based on the design vector $[r_0, \cdots, r_c, T]$, the total mass M, the center of gravity z_{G} and the mooring stiffness matrix C^{M}, it is possible to estimate the total stiffness matrix, with particular reference to the surge, heave, and pitch components $\mathsf{C}_{1,1}$, $\mathsf{C}_{3,3}$ and $\mathsf{C}_{5,5}$. Instead, The hydrodynamics analysis is computationally expensive and carried out using the NEMOH code. It exploits the design vector $[r_0, \cdots, r_c, T]$ and the frequency ω to estimate the added mass matrix, the radiation damping matrix and the first order wave load transfer function vector, with particular reference to the surge, heave and pitch components $\mathsf{A}_{1,1}, \mathsf{A}_{3,3}, \mathsf{A}_{5,5}, \mathsf{B}_{1,1}, \mathsf{B}_{3,3}, \mathsf{B}_{5,5}, \mathsf{X}_{1,1}, \mathsf{X}_{3,3}, \mathsf{X}_{5,5}$. In the last step, the total mass M together with the added mass matrix A, the radiation damping matrix B, the first order wave load transfer function vector X, the total stiffness matrix C and the frequency ω are exploited to compute the RAO according to Equations (7)-(8). This framework is depicted in Figure 1(a).

In order to reduce the computational requirements needed to predict the RAO, we propose to substitute the hydrodynamics analysis (see Figure 1(a)) performed with the NEMOH code, with data-driven surrogate models as depicted in Figure 1(b). In particular, the summary of the inputs feeding the surrogate models and the related outputs is reported in Table 4. Note that we could have opted here for two strategies: (i) build a model where the frequency is an input (i.e., a model able to predict A, B, and X for each possible frequency) or (ii) build a model for a series of selected frequency (i.e., models able to predict relevant quantities at a particular frequency). For strategy (i), it is necessary to construct a huge number of models, with a consequent waste of memory and computational requirements. Moreover, if we need to estimate A, B, and X at a particular frequency, for which the model is not built, it is necessary to interpolate, resulting in less accuracy. Nonetheless, the model prediction for selected frequencies can be more accurate than strategy (ii). For strategy (ii), a single model with frequency as input needs to be constructed, resulting in a larger save of memory and computational requirements. Moreover, the model can be used for all the frequencies of interest without the need for interpolating. For a particular frequency, a model constructed with strategy (ii) can be less accurate than the one of (i). However, in some preliminary tests, the accuracy of the two approaches was statistically indistinguishable, so we decided to adopt the approach (ii) characterized by much fewer memory and computational requirements.

At this point, we can describe the procedure we utilized to generate the data necessary to train the surrogate models. Following the parametrization described in Section III-A we considered $c = 5$ (i.e., five cones), $r_i \in [0, 5]$ $[m]$ with $i \in \{0, \cdots, 5\}$, $T \in \{60, 70, \cdots, 140\}$ $[m]$ and $\omega \in \{0, 0.01, \cdots, 2\}$. These values have been chosen considering a range of reasonable values for FOWTs already in operation [3], [69], [70]. The resulting space of possible geometries is therefore large and for each of these geometries we compute the response for 207 values of ω.

Figure 2 reports first in Figure 2(a) a general geometry to explain the parametrization via $r_0, \cdots, r_5$, and T and then in Figures 2(b) 2(d) some examples of these geometries together with the behavior of the associated values of $\mathsf{A}_{1,1}$, $\mathsf{A}_{3,3}$, $\mathsf{A}_{5,5}$, $\mathsf{B}_{1,1}$, $\mathsf{B}_{3,3}$, $\mathsf{B}_{5,5}$, $\mathsf{X}_{1,1}$, $\mathsf{X}_{3,3}$, and $\mathsf{X}_{5,5}$ as function of the frequency. It is important to note that the behavior of these quantities is non-linear across the range of frequencies examined. This indicates the system's inherent complexity and the consequent challenge in the presented learning problem, making it more difficult to model and predict the system's behavior accurately.

Since, using the NEMOH code, it is unfeasible to estimate the hydrodynamics characteristics for too many geometries[1],[2] we randomly sampled a subset of 28125 geometries in this space.[3]

V. MODELING APPROACH

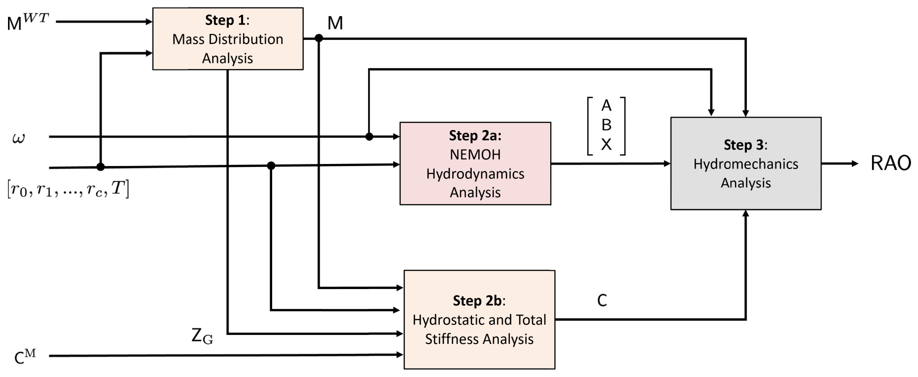
The problem described in Section IV can be mapped to a ML regression problem [43]. However, our scope is broader, and we want to learn a model to contemporary achieve high accuracy and limited computational requirements, namely, the computational effort needed to compute the model output given its inputs should be as limited as possible, while still reaching highly accurate outputs. In fact, the more accurate and the more limited the computational requirements of the learned model, the more this model can be exploited to facilitate the FOWT's complex and iterative design and optimization processes.

Therefore, let us recall the ML regression problem [43] and map our problem to it. Let $\mathcal{X} \subseteq \mathbb{R}^d$ be the input space consisting of f features (in our case the input space defined in Table 4), and let $\mathcal{Y} \subseteq \mathbb{R}^q$ be the output space (in our case the output space defined in Table 4). There exists an unknown (possibly non deterministic) relation $\mu : \mathcal{X} \rightarrow \mathcal{Y}$ that needs to be estimated. In our case, this relation is deterministic and known, namely the model for which we have to build a surrogate as indicated in Figure 1(a), but computationally expensive to compute and hard to approximate without compromising its accuracy. For this purpose, a set of $n \in \mathbb{N}^+$ examples of the input/output relation are collected $\mathcal{D}_n = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, where $\boldsymbol{x}_i \in \mathcal{X}$ and $\boldsymbol{y}_i \in \mathcal{Y}$ $\forall i \in \{1, \cdots, n\}$. In our case, we sample the examples according to the sampling strategy described in Section IV. An ML algorithm [43], [46],

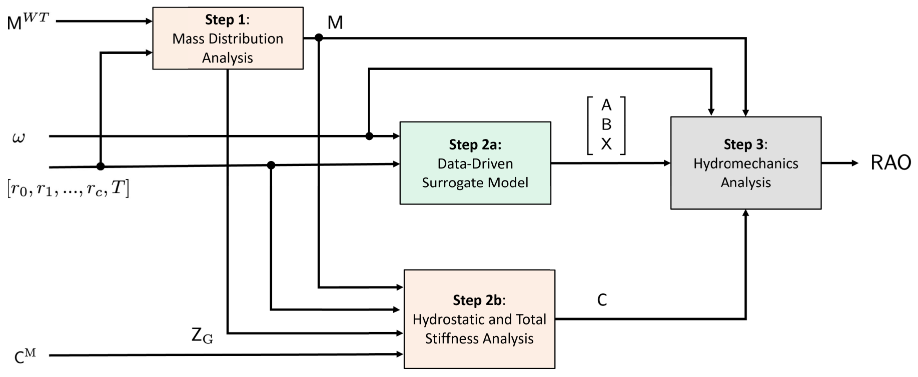
[1]A run for one geometry requires approximately $\approx 15 \div 20$ minutes on our workstation.

[2]Our workstation was equipped with two Intel Xeon Silver 4216 CPUs, 128 GB of RAM, and 512 GB SSD running Windows Server 2019.

[3]In order to generate the database, we needed 3 months of simulation[2].


M^{WT}
Step 1: Mass Distribution Analysis
M
ω
$[r_0, r_1, ..., r_c, T]$
Step 2a: NEMOH Hydrodynamics Analysis
$\begin{bmatrix} A \\ B \\ X \end{bmatrix}$
Step 3: Hydromechanics Analysis
RAO
Z_G
C^M
Step 2b: Hydrostatic and Total Stiffness Analysis
C

(a) Original approach using the NEMOH code


M^{WT}
Step 1: Mass Distribution Analysis
M
ω
$[r_0, r_1, ..., r_c, T]$
Step 2a: Data-Driven Surrogate Model
$\begin{bmatrix} A \\ B \\ X \end{bmatrix}$
Step 3: Hydromechanics Analysis
RAO
Z_G
C^M
Step 2b: Hydrostatic and Total Stiffness Analysis
C

(b) Proposes approach using surrogate models

FIGURE 1. Description of the original and proposed approach to RAO prediction.

TABLE 4. List of inputs and outputs of the surrogate models.

	Name	Description
Input space	$[r_0, \cdots, r_c]$	External radii of the cones describing the parametrization of the platform
	T	Foundation draft
	ω	Frequency
Output space	$A_{1,1}$	Added mass matrix coefficient - surge motion
	$A_{3,3}$	Added mass matrix coefficient - heave motion
	$A_{5,5}$	Added mass matrix coefficient - pitch motion
	$B_{1,1}$	Radiation damping matrix coefficient - surge motion
	$B_{3,3}$	Radiation damping matrix coefficient - heave motion
	$B_{5,5}$	Radiation damping matrix coefficient - pitch motion
	$X_{1,1}$	1st order wave load transfer function matrix coefficient - surge motion
	$X_{3,3}$	1st order wave load transfer function matrix coefficient - heave motion
	$X_{5,5}$	1st order wave load transfer function matrix coefficient - pitch motion

[71] $\mathscr{A}_{\mathcal{H}}$ characterized by its hyperparameters $\mathcal{H}$ selects a model (function) $f : \mathcal{X} \rightarrow \mathcal{Y}$ from a set $\mathcal{F}$ of possible ones based on $\mathcal{D}_n$ to approximate μ. Many different algorithms exist with different approximation properties, practical effectiveness, and computational requirements and $\mathcal{H}$ implicitly or explicitly defines $\mathcal{F}$ and the quality of the learned model [43], [47]. For this reason, in Section V-A, we will select a subset of the possible algorithms since their properties make them suited for our application. The error of f in approximating μ is measured by a prescribed metric $\mathtt{M} : \mathcal{F} \times (\mathcal{X} \times \mathcal{Y}) \rightarrow \mathbb{R}$ computed over the available data [71]. The metric needs to be computed on a fresh set of data $\mathcal{T}_m = \{(\boldsymbol{x}'_1, \boldsymbol{y}'_1), \ldots, (\boldsymbol{x}'_m, \boldsymbol{y}'_m)\}$, where $\boldsymbol{x}'_i \in \mathcal{X}$ and $\boldsymbol{y}'_i \in \mathcal{Y}$ $\forall i \in \{1, \cdots, m\}$, different from $\mathcal{D}_n$, to avoid overfitting problems (or data snooping) and ensure reliable results thanks to a grounded Error Estimation (EE) phase [47]. We will devote Section V-B to this topic. $\mathcal{H}$ must be carefully tuned to optimize the desired metric $\mathtt{M}$. For this purpose, a careful and grounded Model Selection (MS) phase will be performed [47] and Section V-C will be devoted to describing this phase. Finally, Section V-D is devoted to reducing the computational requirements (i.e., memory and computation) of the

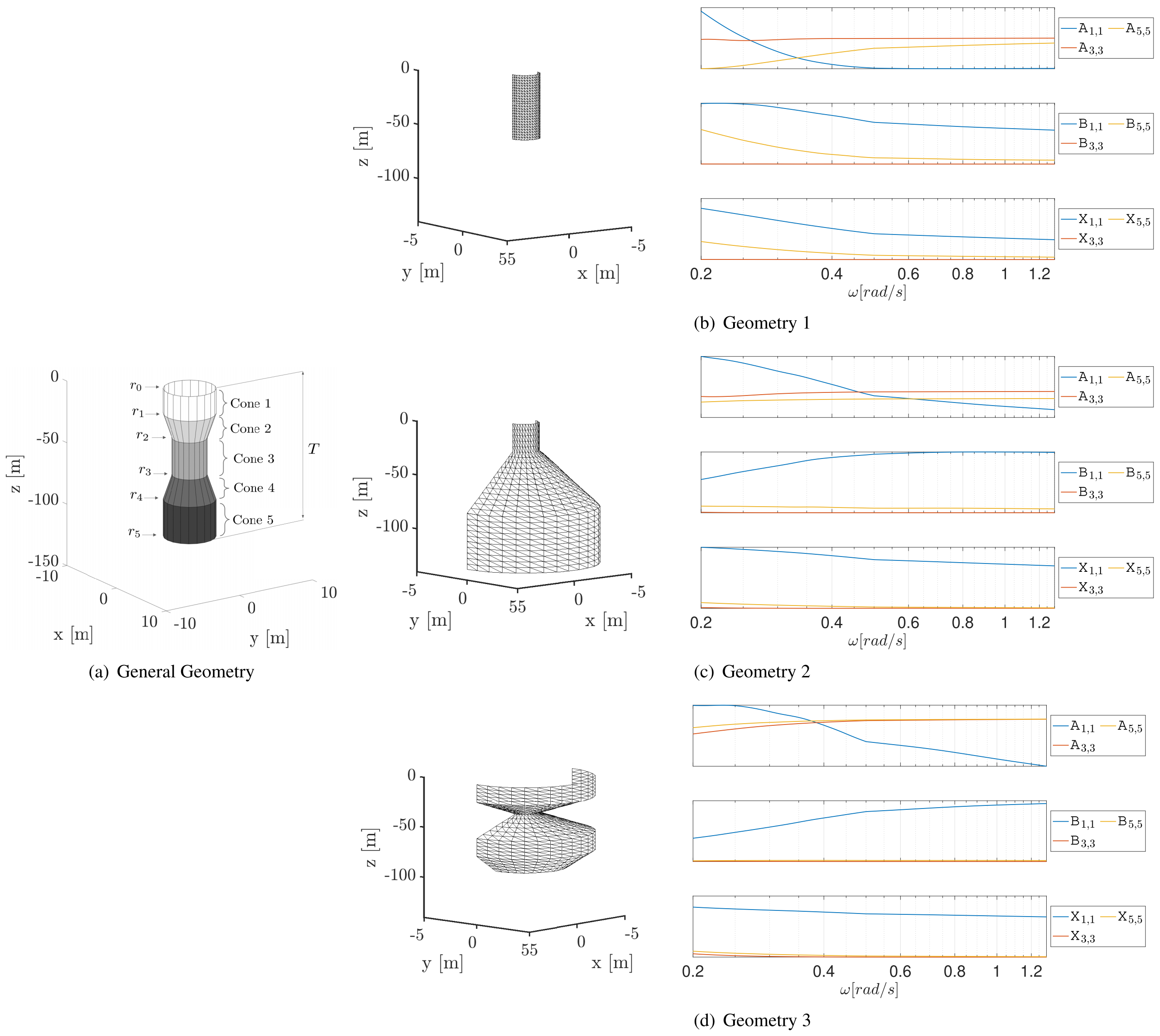


FIGURE 2. General geometry showing the employed parametrization ($r_0, \cdots, r_5$, and T) and Examples of geometries with the associated values of the added mass coefficients in surge $A_{1,1}$, heave $A_{3,3}$, and pitch $A_{5,5}$, the radiation damping matrix coefficients in surge $B_{1,1}$, heave $B_{3,3}$, and pitch $B_{5,5}$, and the first order wave load transfer function matrix coefficients in surge $X_{1,1}$, heave $X_{3,3}$, and pitch $X_{5,5}$.

learned model as much as possible without impacting their accuracy.

A. CHOOSING THE ALGORITHM

Note that many ML algorithms for solving regression problems exist in the literature. In particular, it is possible to identify three main different families of ML algorithms which are mostly effective in practice[4] [72], [73]: the Kernel Methods in their Bayesian [74] and frequentist [75] versions, the Neural Networks in their shallow [76] and deep [46] versions and the Ensemble Methods [77]. Inside any of these families, there are many different algorithms and variations of the same algorithms with different approximation properties, practical effectiveness and computational requirements. Unfortunately, the no-free-lunch theorem [78] and some pathological example [79], ensure us that the choice of an algorithm strongly depends on the specific application and there is no way to choose the best solutions a-priori. Nevertheless, keeping the approach as simple as possible [80] and also keeping in mind that the more data you have the more complex the algorithm can be and vice versa [81], [82] are always good guidelines. Moreover, the experience of the data scientists [83], [84] can further improve the quality of the resulting selection strategy.

[4]https://www.kdnuggets.com/2015/12/harasymiv-lessons-kaggle-machine-learning.html

In the following sections, we will describe the most effective approach, according to the recent results in the literature and the experience of the authors, inside each single family of ML algorithms. For each of the methods, we will describe the idea behind it, its formal definition, its approximation properties, and its computational requirements.

In order to simplify the notation, the above-mentioned algorithms have been described in different subsections. In this way, the same symbol can have a different meaning based on the specific algorithms we are describing.

1) KERNEL METHODS

Kernel methods [74], [75] owe their name to the use of kernel trick for distances [85], which enable them to implicitly operate in higher-dimensional spaces with respect to the original input space. This allows to easily transform linear models into nonlinear ones, maintaining all their properties (i.e., approximation properties, practical effectiveness, and computational requirements) [43].

The general approach of kernel methods for regression is quite simple. For the sake of readability, we consider the case of $q = 1$, namely, the output space is mono-dimensional. The extension to $q > 1$ will be discussed in Section V-D1. Let us consider a linear model in the original input space

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b, \tag{9}$$

with $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ which basically represent the $\mathcal{F}$. Given a metric $\mathtt{M}$ computed on $\mathcal{D}_n$ the best values for the model parameters are selected according to the Occam's razor principle, namely trading off the error on $\mathcal{D}_n$, measured with $\mathtt{M}$ [71], [86], and the complexity of the solution, measured with a complexity measure [43] also called regularizer $\mathtt{R}$ [87]. Implementing this concept via Tikhonov regularization [88], we can formulate then the problem as

$$f^* : \quad \arg\min_{f \in \mathcal{F}} \quad \mathtt{M}(f) + \lambda \mathtt{R}(f). \tag{10}$$

In other words, the best model f^* is chosen to be the simplest one that is able to learn from data without overfitting them. $\lambda \in \mathbb{R}^+$ is a hyperparameter, that must be tuned and set a-priori (i.e., not obtained as an output of the optimization procedure): it regulates the trade-off between the overfitting tendency, related to the minimization of $\mathtt{M}$, and the underfitting tendency, related to the minimization of $\mathtt{R}$. We will discuss how to tune λ in Section V-C.

Depending on the choice of $\mathtt{M}$ and $\mathtt{R}$ one can derive many algorithms. For example, ridge regression can be derived using mean squared error for $\mathtt{M}$ and Euclidean norm of $\boldsymbol{w}$ for $\mathtt{R}$ [89], lasso regression can be derived using mean squared error for $\mathtt{M}$ and Manhattan norm of $\boldsymbol{w}$ for $\mathtt{R}$ [90], and support vector regression can be derived using mean epsilon insensitive error for $\mathtt{M}$ and Euclidean norm of $\boldsymbol{w}$ for $\mathtt{R}$ [91]. Note also that, if $\mathtt{M}$ and $\mathtt{R}$ are convex, Problem (10) results in convex optimization problem which can be solved effectively and efficiently [92]. For this reason most kernel methods rely on convex $\mathtt{M}$ and $\mathtt{R}$ [93].

If $\mathtt{M}$ and $\mathtt{R}$ satisfy certain properties [94] it is possible to prove that the $f(\boldsymbol{x})$ can be reformulated as a linear combination of the input data

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b, \tag{11}$$

with $\boldsymbol{\alpha} \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Note that if we use a non-linear model, instead of a linear one

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}) + b, \tag{12}$$

namely, by projecting the data from the original space into another space by means of the non-linear mapping $\boldsymbol{\phi} : \mathbb{R}^d \rightarrow \mathbb{R}^D$, $f(\boldsymbol{x})$ can be reformulated as

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}) + b. \tag{13}$$

This mapping can be explicit, i.e., via feature mapping or engineering [95], [96], or implicit via kernel trick [85]. By exploiting the kernel trick or distances [85] we can reformulate $f(\boldsymbol{x})$ as

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \mathtt{K}(\boldsymbol{x}_i, \boldsymbol{x}) + b, \tag{14}$$

where $\mathtt{K}$ is a kernel function [97] which implicitly maps the input data in $\boldsymbol{\phi}$ and computes the dot product in that space.

Several kernel functions can be retrieved in literature, each one with a particular property that can be exploited based on the problem under exam. Usually, the Gaussian kernel is chosen

$$\mathtt{K}(\boldsymbol{x}_i, \boldsymbol{x}) = e^{-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}\|^2}, \tag{15}$$

because of the theoretical reasons described in [98] and because of its effectiveness [72], [73]. $\gamma \in \mathbb{R}^+$ is another hyperparameter, which regulates the nonlinearity of the solution that must be tuned (see Section V-C). Note that the Gaussian kernel is able to implicitly create an infinite dimensional $\boldsymbol{\phi}$ and thanks to this, the kernel methods are able to learn any possible function [98]. Note that this implicit expensive computation requires to compute dot products in the original space and does not require any explicit feature mapping or engineering. Moreover, with this trick, if finding f^* when the model is linear results in a convex problem, it remains convex also when finding f^* when the model is non-linear. We will discuss in Section V-D the problem of tuning the computational requirements of the model.

2) NEURAL NETWORKS

Neural networks [46], [76] owe their name to the original scope of these models to try to emulate a biological brain. Neural Networks are based on connected units called neurons, which loosely model the neurons in a brain [99]. Like synapses in a biological brain, each connection can transmit a signal to other neurons. A neuron receives a signal, then processes it, and can signal neurons connected to it. The signal at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs.

From a mathematical and practical perspective, the approach of neural networks for regression is quite similar to the kernel methods (see Section V-A1) with one single exception. In neural networks, the non-linear projection from the original space to another space is not fixed a-priori, as in kernel methods, explicitly (via feature mapping) or implicitly (via kernel trick), but learned from the data. In other words, neural networks are contemporarily able to learn the non-linear mapping, called representation [46], and the regressor. More formally, a single-layer, i.e., shallow, multi-output neural network for regression can be defined as follows

$$\boldsymbol{f}(\boldsymbol{x}) = W_0 \boldsymbol{\sigma}_1(W_1 \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_0, \tag{16}$$

with $W_0 \in \mathbb{R}^{q \times h_1}$, $\boldsymbol{b}_0 \in \mathbb{R}^q$, $W_1 \in \mathbb{R}^{h_1 \times d}$, $\boldsymbol{b}_1 \in \mathbb{R}^{h_1}$, $\sigma_{1,i}$ with $i \in \{1, \cdots, h_1\}$ are non-linear activation functions [76] and $h_1 \in \mathbb{N}^+$ is the number of hidden neurons which is a hyperparameter that regulates the non-linearity of the model (see Section V-C to understand how to tune h_1). All these parameters basically represent the $\mathcal{F}$. Note that many activation functions exist [100] (e.g., the linear LIN, the hyperbolic tangent TANH and the rectified linear unit RELU) and researchers have also developed ways to not select one between the existing ones but to actually learn also the best activation function for a particular problem [101], [102]. Note that, with this formalization, it is easy to understand the relation between a shallow model, i.e., Eq. (16), and kernel methods, i.e., Eq. (13), is that $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\sigma}(W_1 \boldsymbol{x} + \boldsymbol{b}_1)$, namely the $\boldsymbol{\phi}$ is not fixed a-priori but depends on the parameters W_1 and $\boldsymbol{b}_1$ (and possibly $\boldsymbol{\sigma}_1$) that must be learned from the data.

A multi-layer, i.e., deep, multi-output neural network for regression can be defined as follows

$$\boldsymbol{f}(\boldsymbol{x}) = W_0 \boldsymbol{r}_l(\boldsymbol{x}) + \boldsymbol{b}_0,$$
$$\begin{cases} \boldsymbol{r}_i(\boldsymbol{x}) = \boldsymbol{\sigma}_l(W_l \boldsymbol{r}_{i-1}(\boldsymbol{x}) + \boldsymbol{b}_l), & i \in \{2, \cdots, l\} \\ \boldsymbol{r}_1(\boldsymbol{x}) = \boldsymbol{\sigma}_1(W_1 \boldsymbol{x} + \boldsymbol{b}_1) \end{cases} \tag{17}$$

with $W_0 \in \mathbb{R}^{q \times h_l}$ and $\boldsymbol{b}_0 \in \mathbb{R}^q$, $W_i \in \mathbb{R}^{h_i \times h_{i-1}}$ and $\boldsymbol{b}_i \in \mathbb{R}^{h_i}$ with $i \in \{2, \cdots, l\}$, $W_1 \in \mathbb{R}^{h_1 \times d}$ and $\boldsymbol{b}_1 \in \mathbb{R}^{h_1}$, $\sigma_{i,j}$ with $i \in \{1, \cdots, l\}$ and $j \in \{1, \cdots, h_1\}$ are non-linear activation functions and $h_i, l \in \mathbb{N}^+$ with $i \in \{1, \cdots, l\}$. l is the number of layers of the network and h_i with $i \in \{1, \cdots, l\}$ are the number of hidden neurons in each layer. The letters are hyperparameters that regulate the non-linearity of the model (see Section V-C to understand how to tune them). All these parameters basically represent the $\mathcal{F}$. If we set $l = 1$ we get the shallow model presented in Eq. (16). Note also that, in this context, $\boldsymbol{r}_l(\boldsymbol{x})$ is usually called representation while $\boldsymbol{f}(\boldsymbol{x})$ is usually called model [46]. Note that, as proved in [103], deep and shallow networks have the same approximation properties, i.e., there always exists an equivalent shallow architecture to a deep one and vice-versa. Nevertheless, a deep network can, in some cases, achieve better approximation properties with a smaller number of weights [104]. Note that based on the functional form of the matrices W_i with $i \in \{1, \cdots, l\}$ it is possible to define different kind of layer [46], e.g., convolutions, pooling, dropout, and dense. The choice of a particular layer type is again a hyperparameter that defines the final structure of the network and needs to be carefully tuned to achieve the optimal performance [105], [106]. Note that, in our case, simple dense layers are usually enough since there is no particular structure in the data [46].

Once the structure of the $\boldsymbol{f}$ is defined, the method for finding $\boldsymbol{f}^*$ is analogous to the one of the Kernel Methods. We first need to select a metric $\mathtt{M}(\boldsymbol{f})$ and a regularizer $\mathtt{R}(\boldsymbol{f})$ and then we have to solve the following problem

$$\boldsymbol{f}^* : \quad \arg\min_{\boldsymbol{f} \in \mathcal{F}} \quad \mathtt{M}(\boldsymbol{f}) + \lambda \mathtt{R}(\boldsymbol{f}). \tag{18}$$

In Neural Networks the regularizer $\mathtt{R}(\boldsymbol{f})$ can be quite complex [46]. In fact we can enforce a sparsity in the solution of a layer i by including in $\mathtt{R}(\boldsymbol{f})$ the Manhattan norm of the weights in W_i. Or we can simply reduce the overfitting in the last layer by including in $\mathtt{R}(\boldsymbol{f})$ the Euclidean norm of the weights in W_0. Nevertheless, in Neural Networks, there are other ways to enforce regularization. For example a dropout layer acts as a regularizer for the solution [107].

Unfortunately, because of the non-linearity of $\boldsymbol{f}$ Problem (18) is non-convex even if $\mathtt{M}(\boldsymbol{f})$ and $\mathtt{R}(\boldsymbol{f})$ are convex. The first practical and effective workaround to this issue was proposed by [108] with the famous backpropagation algorithm. The latter is basically a gradient descent algorithm, and consequently, it requires Problem (18) to be differentiable. Many evolution of this algorithm have been proposed during the years [109], [110], e.g., Adadelta, Adagrad, Adam and stochastic gradient descent are very effective libraries that are able to perform automatic differentiation [111], [112]. Differentiation for Neural Networks has the same importance of convexity for Kernel Methods [43], [46].

3) ENSEMBLE METHODS

Ensemble Methods [77] implement the wisdom of crowds principle in ML, namely rely on the collective opinion of a group of individuals rather than on a single expert in the group to reach better decisions [113]. In a more formal way, Ensemble Methods, instead of using a single complex model (learner), like the Kernel Methods or Neural Networks, combine many different simple (weak) learners. XGBoost [114] is surely the most efficient and effective one among the Ensemble Methods and is the last evolution in the long history. The original idea was to exploit a series of decision trees [115] combined together to achieve a higher accuracy with respect to a single decision tree. Decision trees are often chosen as weak learners since they are quite easy to tune, computationally efficient, numerically robust and able to naively and easily handle missing values and categorical features. Many strategies have been developed during the years to construct and combine these trees. The first one was the bagging strategy [116], namely subsample, e.g., via bootstrap [117], the original dataset, construct a decision tree on each of the subsample and then combine them together with a majority voting procedure. This approach was then evolved into the famous Random Forests algorithm [118] by adding random subset feature selection, and finally into the Random Forests Rotation algorithm [119] by further adding random feature rotation. The second main strategy was the boosting one [120], namely to sequentially build trees minimising the error of the previous one boosting the influence of high performing models. This approach was then evolved into the famous gradient boosting algorithm [121] where gradient descent algorithm was introduced to minimize the error of sequential models, and finally into the XGBoost (eXtreme Gradient Boosting) algorithm [114] by further adding parallel processing, pruning and regularisation.

Let us more formally define XGBoost. For the sake of readability, we consider the case of $q = 1$, namely the output space is mono-dimensional. The extension to $q > 1$ will be discussed in Section V-D3. In this case, the regression model is defined as a tree ensemble model using $k \in \mathbb{N}^+$ additive functions

$$f(\boldsymbol{x}) = \sum_{i=1}^{k} g_i(\boldsymbol{x}), \tag{19}$$

where $g_i \in \mathcal{T}$ with $i \in \{1, \cdots, k\}$ and $\mathcal{T}$ is the space of all the possible regression trees. Each tree g_i is defined by a structure s_i, a number of leaves $e_i \in \mathbb{N}^+$. Each leaf is characterised by a weight (or score) $w_{i,j} \in \mathbb{R}$ with $i \in \{1, \cdots, k\}$ and $j \in \{1, \cdots, e_i\}$. For a given example $\boldsymbol{x}$, the decision rules r_i with $i \in \{1, \cdots, k\}$ in the trees to classify it into the leaves and calculate the final prediction by summing up the score in the corresponding leaves given by $w_{i,j}$ with $i \in \{1, \cdots, k\}$ and $j \in \{1, \cdots, e_i\}$. To learn the set of functions used in the model, we minimize the usual regularized objective we saw before

$$f^* : \quad \arg\min_{f \in \mathcal{F}} \quad \mathtt{M}(f) + \lambda \mathtt{R}(f), \tag{20}$$

but where, in this case, $\mathtt{M}$ is usually a convex differentiable metric and $\mathtt{R}$ penalises the complexity of the model both in terms of the number of leaves and the norm of the weights.

Since the model of Eq. (19) are trees, Problem (20) cannot be optimized using traditional optimization methods. For this reason, the model is trained in an additive manner. Formally, let g_t^* be the optimal model at the t-th iteration, we will need to find the g_t^* such that

$$g_t^* : \quad \arg\min_{g_t \in \mathcal{T}} \quad \mathtt{M}(g_t + g_{t-1}^*) + \lambda \mathtt{R}(g_t). \tag{21}$$

This means that the g_t that most improves the model of Eq. (19) according to Eq. (20) is greedily added. Since it is usually impossible to enumerate all the possible tree structures, a greedy algorithm that starts from a single leaf and iteratively adds branches to the tree is used instead.

Besides $\mathtt{M}$, two additional techniques are used to further prevent overfitting. The first technique is shrinkage [122], which scales newly added weights by a factor η after each step of the tree boosting to reduce the influence of each individual tree and leaves space for future trees to improve the model. The second technique is feature subsampling, namely using a subset of features during tree branches creation. This technique is used in the Random Forests algorithm [118] and efficiently and effectively reduces overfitting.

XGBoost is not only an algorithm, it is also an open-source library [114]. The library focuses on speed, flexibility, and model performance, not only from the algorithmic perspective but also from all underlying system optimizations (e.g., parallelization, caching and hardware optimization).

XGBoost has many hyperparameters but not all of them have the same effects on the final performance of the model [114]. There are three main classes of hyperparameters: General Parameters (e.g., what booster to use, in our case the tree gradient boosting, and the number of threads), the Booster Parameters (e.g., η, k, the maximum depth of the trees τ, the minimum loss reduction to make a split γ, the fraction of samples and features used train the trees ϱ_s, ϱ_f and the maxim number of leaves $\varkappa$) and the Learning Task Parameters (e.g., $\mathtt{M}$, $\mathtt{R}$ and λ). The latters hyperparameters need to be carefully tuned (see Section V-C).

B. ESTIMATING THE ERROR OF THE MODEL

As mentioned earlier, the first step toward the evaluation of the error of a model is to possess a fresh set of data $\mathcal{T}_m$ since $\mathcal{D}_n$ cannot be safely used to contemporary build the model and estimate its error [47]. Some theoretical approaches have tried to address the problem of using the same data to build and estimate the error of the model which is a fundamental problem when the data is very limited, i.e., the small sample regime [123]. In our case the number of samples is quite big so splitting them in two sets, $\mathcal{D}_n$ and $\mathcal{T}_m$, is not a problem.

Once $\mathcal{T}_m$, we have to decide a metric to assess the quality of the mode. This metric strongly depends on the application. In our case, as described in Section IV, the ML models are used to replace the computational expensive hydrodynamic analysis performed with the NEMOH software to compute the RAO as described in Figure 1. Consequently, our metric $\mathtt{M}$ determines how good we are in predicting the RAO. In particular, we will use the following metrics $\mathtt{M}$ to evaluate noitemsep,topsep=0ex

- Mean absolute percentage error ($\mathtt{MAPE}$),
- Scatter plot,
- Comparison, for different values of ω,

between the estimated RAO when the NEMOH software is used and when NEMOH is replaced by the ML-based surrogate models (Figure 1(b)).

Moreover, since in our work we want to learn a model to contemporary achieve high accuracy and limited computational requirements, we need to define a metric to measure these computational requirements. In particular, we are interested in the computational effort needed to compute the model output given its inputs. For this purpose we will use the following metric $\mathtt{M}$ noitemsep,topsep=0ex

- Average time, in seconds, to predict the RAO ($\mathtt{TIME}$),
- Memory requirement, in megabyte, to store the model that predicts the RAO ($\mathtt{MEM}$).

Finally, note that we do not take into account the time necessary to build the model since our work focuses on designing models that have to be exploited to facilitate the FOWT's complex and iterative design and optimization process. In fact, during this phase, we use models already built and consequently the time needed for this phase is negligible. Nevertheless, we observe that the time needed to build the NEMOH must be measured in years of research and development. Instead, the ML-based models that we propose in this work may require, in the worst case, a few months. In conclusion, the difference is an order of magnitude in favor of the ML model when it comes to considering the time needed to build the model.

C. SELECTING THE BEST HYPERPARAMETERS

The problem of tuning the performance of a ML algorithm, namely, to select the best hyperparameters, is a fundamental issue in ML [47]. Resampling techniques are commonly used by researchers and practitioners since they work well in most situations and this is why we will exploit them in this work. Other alternatives exist, based on the Statistical Learning Theory, but they tend to underperform resampling techniques when the number of samples is large as in our case [47]. Resampling techniques are based on a simple idea: the original dataset $\mathcal{D}_n$ is resampled once or many (n_r) times, with or without replacement, to build two independent datasets called learning and validation sets, respectively $\mathcal{L}_l^r$ and $\mathcal{V}_v^r$ with $r \in \{1, \cdots, n_r\}$, such that

$$\mathcal{L}_l^r \cap \mathcal{V}_v^r = \oslash, \quad \mathcal{L}_l^r \cup \mathcal{V}_v^r = \mathcal{D}_n. \tag{22}$$

Subsequently, to select the best hyperparameters' combination $\mathcal{H}$ in a set of possible ones $\mathfrak{H} = \{\mathcal{H}_1, \mathcal{H}_2, \cdots\}$ for the algorithm $\mathscr{A}_{\mathcal{H}}$ or, in other words, to perform the MS phase, the following procedure has to be applied:

$$\mathcal{H}^* : \quad \arg\min_{\mathcal{H} \in \mathfrak{H}} \sum_{r=1}^{n_r} \mathtt{M}(\mathscr{A}_{\mathcal{H}}(\mathcal{L}_l^r), \mathcal{V}_v^r), \tag{23}$$

where $\mathscr{A}_{\mathcal{H}}(\mathcal{L}_l^r)$ is a model built with the algorithm $\mathscr{A}$ with its set of hyperparameters $\mathcal{H}$ and with the data $\mathcal{L}_l^r$, and where $\mathtt{M}(f, \mathcal{V}_v^r)$ is a desired metric. Since the data in $\mathcal{L}_l^r$ is independent from the data in $\mathcal{V}_v^r$, $\mathcal{H}^*$ will be the set of hyperparameters which allows achieving a small error on a data set that is independent from the training set.

This procedure needs to be slightly modified in order to meet the requirements of our work. In fact, our purpose is to simultaneously optimize different metrics $\mathtt{M}$ (i.e., accuracy, measured with $\mathtt{MAPE}$, and computational requirements, measured with $\mathtt{TIME}$ and $\mathtt{MEM}$). For this reason we will perform a MS phase following the procedure defined as follows:

$$\mathcal{H}^*\!: \arg\min_{\mathcal{H} \in \mathfrak{H}} \sum_{r=1}^{n_r} \iota \mathtt{M}_1(\mathcal{H}, \mathcal{L}_l^r, \mathcal{V}_v^r) + (1-\iota)\mathtt{M}_2(\mathcal{H}, \mathcal{L}_l^r, \mathcal{V}_v^r), \tag{24}$$

where $\iota \in [0, 1]$ and $\mathtt{M}_1(\mathcal{H}, \mathcal{L}_l^r, \mathcal{V}_v^r)$ and $\mathtt{M}_2(\mathcal{H}, \mathcal{L}_l^r, \mathcal{V}_v^r)$ are particular metrics (usually one referring to the accuracy as the $\mathtt{MAPE}$ and the other one referring to the accuracy computational requirements as the $\mathtt{TIME}$ and $\mathtt{MEM}$) computed on $\mathcal{V}_v^r$ for the estimated RAO when the NEMOH software is used and when NEMOH is replaced by the ML-based surrogate models trained on $\mathcal{L}_l^r$ with the set of hyperparameters $\mathcal{H}$. $\iota \in [0, 1]$ balances the trade off between accuracy and computational requirements (e.g., for $\iota \to 1$ we tend to maximize the accuracy not caring at all about the computational requirements while for $\iota \to 0$ we obtain the opposite result). Since the number of hyperparameter combinations may be very large, a random search approach will be adopted [124] to explore this space.

D. REDUCING THE COMPUTATIONAL REQUIREMENTS OF THE SURROGATE MODELS

In this section, we will define the final structure of the different algorithms (and related models) that we propose, together with the associated hyperparameters, to build the different ML-based surrogate models needed to predict the RAO (see Section IV). We will leverage on the properties described in Section V-A, keeping in mind the metrics defined in Section V-B, to define three algorithms (and related models) able to deliver accurate yet computationally aware results.

1) KERNEL METHODS

As we discussed in Section V-A1, the model of Kernel Methods, when using a Gaussian kernel is defined as

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i e^{-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}\|^2}, \tag{25}$$

where γ is a hyperparameter and $\boldsymbol{\alpha} \in \mathbb{R}^n$ is the set of parameters to be tuned during the training phase. Note that, if $q > 1$ as in our case, it is required to build a different model for each of the quantities to predict (see Section IV). We did not insert the parameter b since, thanks to the Gaussian kernel properties of being able alone to learn any possible model [98], it can be neglected. In order to tune $\boldsymbol{\alpha}$ we need to define $\mathtt{M}$ and $\mathtt{R}$.

For what concerns $\mathtt{M}$, as described in Section V-A1, it must be convex and note that we do not have to use the same metric to evaluate the performance of the final model (see Section V-B) and to evaluate the performance of each single surrogate model composing the final RAO predictive model (see Section IV) as discussed in many works [43], [86], [93]. In this work we will use the mean epsilon insensitive error

$$\mathtt{M}(f) = \frac{1}{n} \sum_{i=1}^{n} \max[0, |y_i - f(\boldsymbol{x}_i)| - \epsilon], \tag{26}$$

where $\epsilon \in \mathbb{R}^+$ is a hyperparameter of $\mathtt{M}$. We choose this metric since it increases the sparsity of the model [75], [91], namely the number of α_i with $i \in \{1, \cdots, n\}$ equal to zero. In fact, the more alphas are equal to zero, the less computational effort will require the model of Eq. (25).

For what concerns $\mathtt{R}$, instead, we will use the nowclassical L2 regularizer of the weight of the linear model in the space induced by the kernel which can be formulated as follows [75], [91]

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j e^{-\gamma\|\boldsymbol{x}_i-\boldsymbol{x}\|^2}. \tag{27}$$

To further reduce the number of alphas equal to zero, we will add a L1 regularizer [90], [125] for the alphas and consequently we have that

$$\mathtt{M}(f) = (1-\beta)\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j e^{-\gamma\|\boldsymbol{x}_i-\boldsymbol{x}\|^2} + \beta\sum_{i=1}^{n}|\alpha_i|, \tag{28}$$

where $\beta \in [0, 1]$ is a hyperparameter of $\mathtt{R}$.

Combining these definitions with Problem (10) we have that

$$\begin{aligned}\boldsymbol{\alpha}^* : \arg\min_{\alpha\in\mathbb{R}^n} \quad & \frac{1}{n}\sum_{i=1}^{n}\max[0, |y_i - f(\boldsymbol{x}_i)| - \epsilon] \\ & + \lambda\left[(1-\beta)\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j e^{-\gamma\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2} + \beta\sum_{i=1}^{n}|\alpha_i|\right],\end{aligned} \tag{29}$$

which is a convex problem for which a standard solver can achieve the global minimum efficiently and effectively [92].

2) NEURAL NETWORKS

As we discussed in Section V-A2, the model of Neural Networks is defined as

$$\begin{aligned}&f(\boldsymbol{x}) = W_0\boldsymbol{r}_l(\boldsymbol{x}) + \boldsymbol{b}_0,\\ &\begin{cases}\boldsymbol{r}_i(\boldsymbol{x}) = \boldsymbol{\sigma}_l(W_l\boldsymbol{r}_{i-1}(\boldsymbol{x}) + \boldsymbol{b}_l), & i \in \{2, \cdots, l\}\\ \boldsymbol{r}_1(\boldsymbol{x}) = \boldsymbol{\sigma}_1(W_1\boldsymbol{x} + \boldsymbol{b}_1)\end{cases}\end{aligned} \tag{30}$$

where $W_0 \in \mathbb{R}^{q\times h_l}$ and $\boldsymbol{b}_0 \in \mathbb{R}^q$, $W_i \in \mathbb{R}^{h_i\times h_{i-1}}$ and $\boldsymbol{b}_i \in \mathbb{R}^{h_i}$ with $i \in \{2, \cdots, l\}$, $W_1 \in \mathbb{R}^{h_1\times d}$ and $\boldsymbol{b}_1 \in \mathbb{R}^{h_1}$ are the set of parameters to be tuned during the training phase. The structure of $W_0, \cdots, W_l$ and $l, h_1, \cdot, h_l, \boldsymbol{\sigma}_1, \cdots, \boldsymbol{\sigma}_l$ are instead hyperparameters of the model that need to be carefully tuned (see Section V-C). Note that, contrary to Kernel Methods, models based on Neural Networks are able, by construction, to predict all quantities (see Section IV).

Since all the space of hyperparameters cannot be fully explored it is often required to exploit experience and theoretical properties to reduce the space of exploration [46], [105]. For this reason, we will fix some hyperparameters. We will set $W_0, \cdots, W_l$ to be classical dense layers, so we do not impose any particular structure since input data is unstructured [46], [76]. For what concern the activation functions we will use for $\boldsymbol{\sigma}_l$ the hyperbolic tangent because of its approximation properties [103] while for $\boldsymbol{\sigma}_{l-1}, \cdots, \boldsymbol{\sigma}_1$ we will exploit the rectified linear unit [126], [127] since it improves the ability to train the model counteracting the gradient vanishing problem [128], [129]. Therefore what remains to be tuned as hyperparameters are $l, h_1, \cdot, h_l,$.

Instead, to tune the parameters $W_0, \boldsymbol{b}_0, \cdots, W_l, \boldsymbol{b}_l$ we need to define $\mathtt{M}$ and $\mathtt{R}$.

For what concerns $\mathtt{M}$, as in Kernel Methods, we do not have to use the same metric to evaluate the performance of the final model and to evaluate the performance of each single surrogate model composing the final RAO predictive model. Instead, differently to Kernel Methods, since the training procedure will result in non convex problem no matter the choice of $\mathtt{M}$, we need just to use a differentiable $\mathtt{M}$ (see Section V-A2). Nevertheless, usually a differentiable convex $\mathtt{M}$ is exploited [46]. In this work we will use the mean squared error averaged over the q outputs

$$\mathtt{M}(f) = \frac{1}{q}\sum_{i=1}^{q}\frac{1}{n}\sum_{j=1}^{n}\left[y_{j,i} - f_i(\boldsymbol{x}_j)\right]^2. \tag{31}$$

We choose this metric since it is one of the most effective, differentiable, and convex metrics for regression [82], [86]. Note that this metric suffers if the order of magnitude of the q target is very different. For this reason, we will make use of a simple normalization

$$\mathtt{M}(f) = \frac{1}{q}\sum_{i=1}^{q}\frac{1}{n}\sum_{j=1}^{n}\left[\frac{y_{j,i} - f_i(\boldsymbol{x}_j)}{\frac{1}{n}\sum_{j=1}^{n}y_{j,i}^2 - \left(\frac{1}{n}\sum_{j=1}^{n}y_{j,i}\right)^2}\right]^2. \tag{32}$$

For what concerns $\mathtt{R}$, instead, we will use the Frobenius norm of W_0, namely $\mathtt{R}(\boldsymbol{f}) = \|W_0\|_{\mathrm{F}}$. This is equivalent to the L2 regularizer we adopt in Kernel Methods. Moreover within layer 1 to l, namely $W_1, \cdots, W_l$, we insert a dropout layer, namely a fraction $\theta_i \in [0, 1]$ of the W_i with $i \in \{1, \cdots, l\}$ are randomly put to zero to improve the generalization of the model [107].

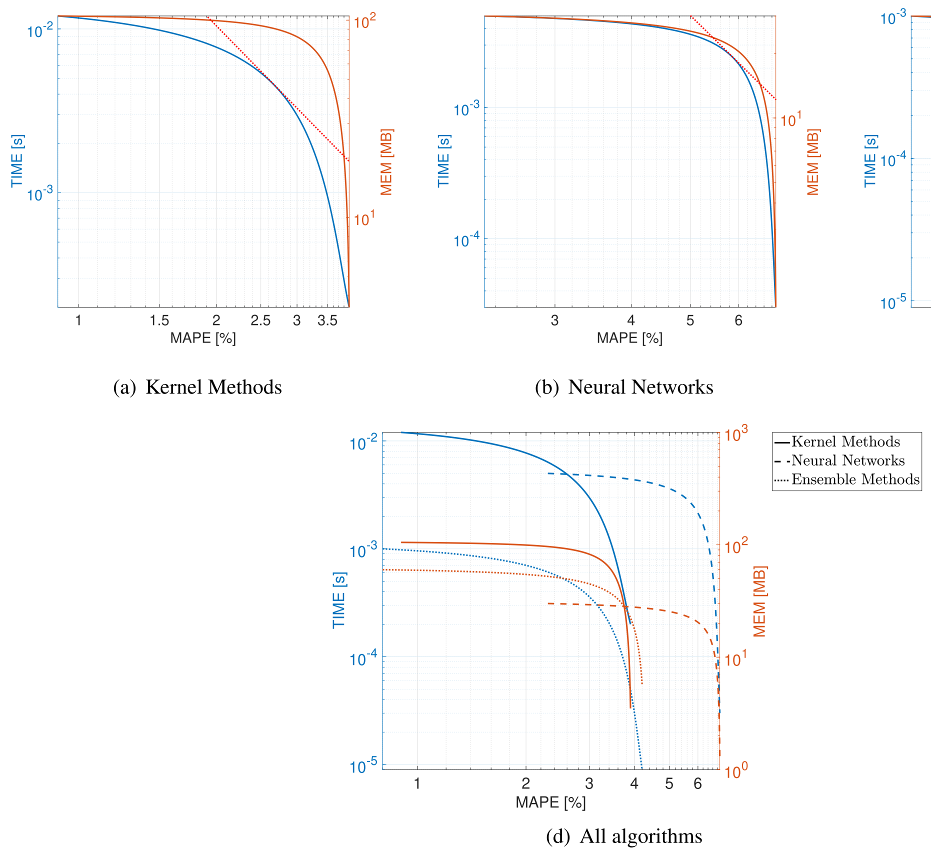
Combining these definitions with Problem (18) we have the final training problem that we will solve using mini batch stochastic gradient descent [130]. Note that this algorithm has two main hyperparameters, the batch size $b_s \in \{1, \cdots, n\}$ and the learning rate $l_r \in \mathbb{R}^+$. These parameters regulate the speed of the optimizer and the effectiveness in reaching a good local minima [46].

3) ENSEMBLE METHODS

As we discussed in Section V-A3, the model XGBoost is defined as follows

$$f(\boldsymbol{x}) = \sum_{i=1}^{k} g_i(\boldsymbol{x}), \tag{33}$$

where k is a hyperparameter and $g_i(\boldsymbol{x})$ is the set of trees to be tuned during the training phase (their structure s_i and their weights $\boldsymbol{w}_i$). Note that, as in Kernel Methods, in our case



(a) Kernel Methods (b) Neural Networks (c) Ensemble Methods

(d) All algorithms

FIGURE 3. Pareto curves of MAPEvsTIME and MAPEvsMEM for the different algorithms.

$q > 1$ and it is required to build a different model for each of the quantities to predict (see Section IV). In order to tune trees, we need to define $\mathtt{M}$ and $\mathtt{R}$.

For what concerns $\mathtt{M}$, similarly to Neural Networks, we will use the mean squared error averaged as it is differentiable and convex

$$\mathtt{M}(f) = \frac{1}{n}\sum_{i=1}^{n}\left[y_i - f(\boldsymbol{x}_i)\right]^2 . \tag{34}$$

For what concerns $\mathtt{R}$, similarly to Kernel Methods and Neural Networks, we will use the L2 norm of $\boldsymbol{w}_i$, namely $\mathtt{R}(\boldsymbol{g}_i) = \|\boldsymbol{w}_i\|^2$. Then the optimal models are found iteratively according to Eq. (21). The hyperparameters λ, k, η, τ, γ, ϱ_s, ϱ_f, and $\varkappa$ need to be carefully tuned (see Section V-C).

Note that each of these hyperparameters have different effects on the performance of the model in terms of accuracy (λ, k, η, τ, γ, ϱ_s, ϱ_f, and $\varkappa$ since they regulate the trade-off between under and over fitting) or computational complexity of the final model (k, τ, γ, and $\varkappa$ since they regulate how much computation and memory are required for the model).

VI. EXPERIMENTAL RESULTS

In this section, we first describe the experimental setting listing all the hyperparameters search spaces used in the MS and EE of the final models.

TABLE 5. Quality in predicting the RAO for best algorithm and ι.

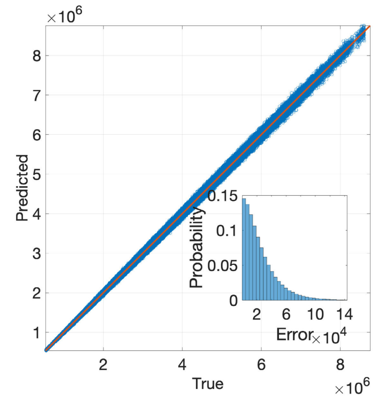
Model	MAPE [%]	TIME [s]	MEM [MB]
Kernel Methods	2.6 ± 0.2	$4.5 \cdot 10^{-3} \pm 0.4 \cdot 10^{-3}$	75 ± 5
Neural Networks	5.8 ± 0.6	$1.8 \cdot 10^{-3} \pm 0.4 \cdot 10^{-3}$	20 ± 3
Ensemble Methods	2.8 ± 0.1	$3.7 \cdot 10^{-4} \pm 0.9 \cdot 10^{-4}$	37 ± 2

In particular, for Eq. (24) two different scenarios have been considered noitemsep,topsep=0ex

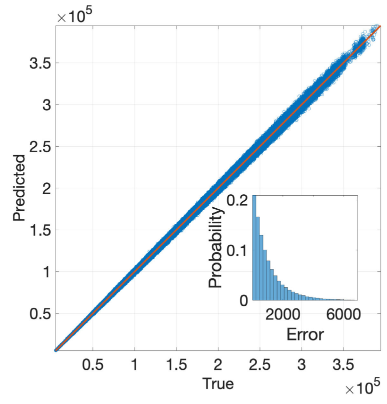
- $\mathtt{M}_1$ equal to $\mathtt{MAPE}$ and $\mathtt{M}_2$ equal to $\mathtt{TIME}$ (MAPEvsTIME);
- $\mathtt{M}_1$ equal to $\mathtt{MAPE}$ and $\mathtt{M}_2$ equal to $\mathtt{MEM}$ (MAPEvsMEM);

namely, we want to find the best trade-off between accuracy (measured with $\mathtt{MAPE}$) and computational resources (measured with $\mathtt{TIME}$ and $\mathtt{MEM}$).

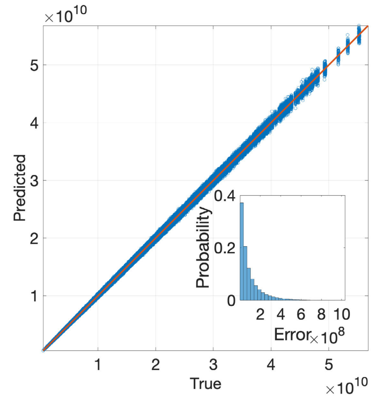
By varying $\iota \in [0, 1]$ it is possible to construct the so called Pareto frontier [131]. The Pareto principle states that a solution belongs to the Pareto Optimal set if there are no other solutions that can improve at least one of the objectives without degrading any other objective [132]. When the Pareto frontier has a simple natural smooth shape (see results in the next section), a good heuristic to choose an optimal point is to use the so called knee of the frontier [133]. In our case, this point represents the value of ι for which increasing



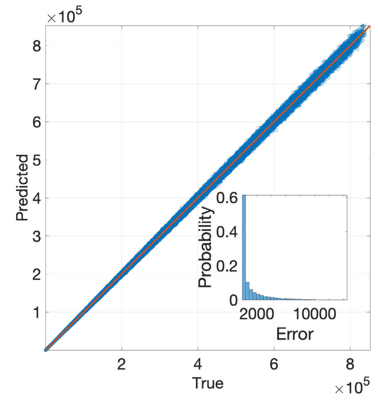
(a) $\mathsf{A}_{1,1}$



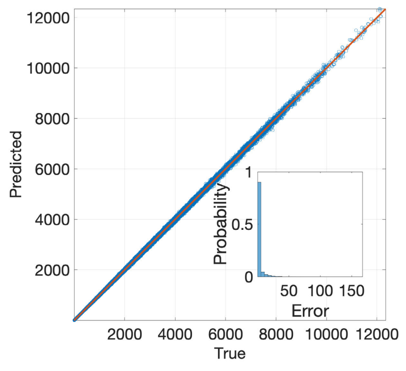
(b) $\mathsf{A}_{3,3}$



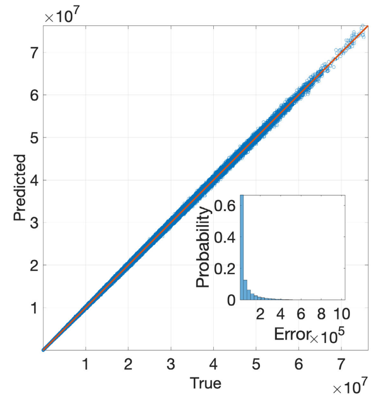
(c) $\mathsf{A}_{5,5}$



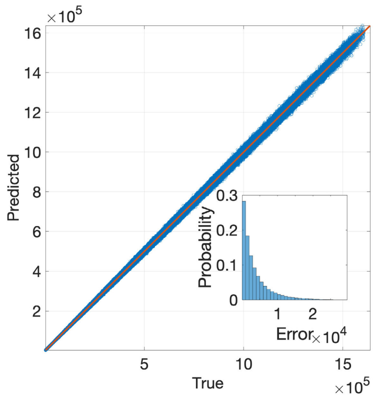
(d) $\mathsf{B}_{1,1}$



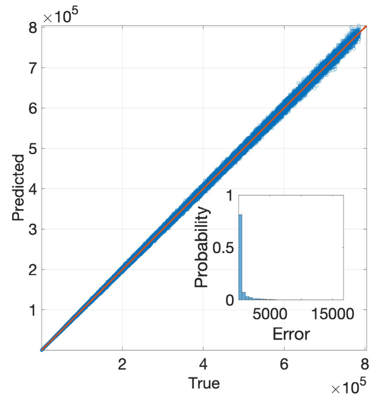
(e) $\mathsf{B}_{3,3}$



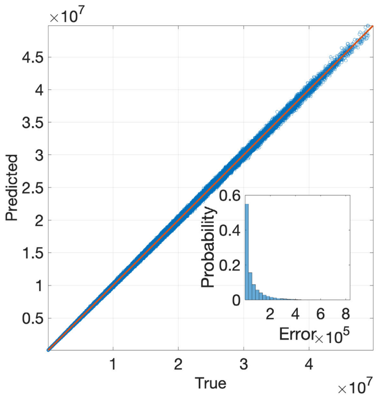
(f) $\mathsf{B}_{5,5}$



(g) $\mathsf{X}_{1,1}$



(h) $\mathsf{X}_{3,3}$



(i) $\mathsf{X}_{5,5}$

FIGURE 4. Scatter True/Predicted and histogram of the absolute error (logarithmic scale) of the surrogate models for the best algorithm and ι.

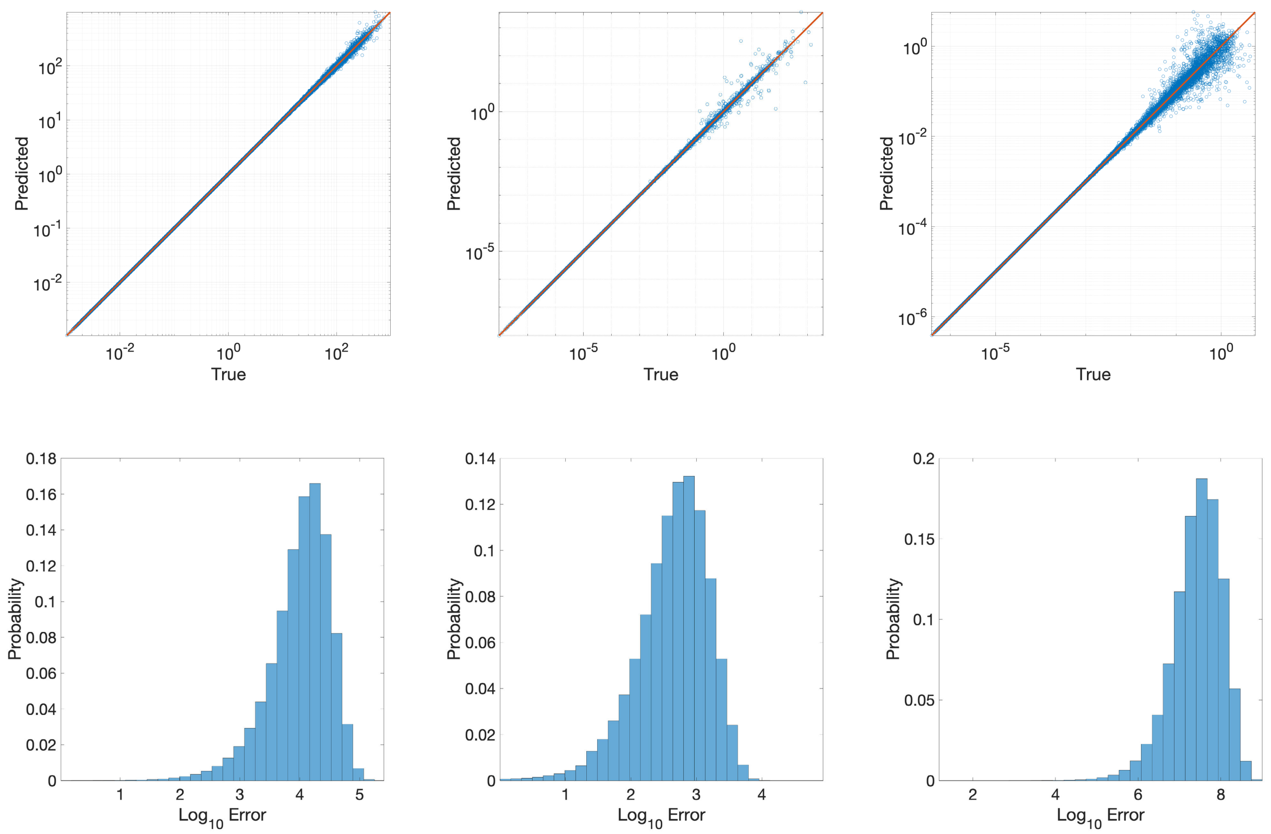
its value leads to an accuracy increase not compensated by enough computational resources decrease and vice versa. The aforementioned principle will be used to identify the optimal ι.

A. EXPERIMENTAL SETTING

This section is devoted to the definition of the experimental setting.

For what concerns the MS phase we set $n_r = 10$ with 80% of the data in $\mathcal{L}_l^r$, 10% in $\mathcal{V}_v^r$, and 10% in $\mathcal{T}_t^r$. Moreover we tested different values of $\iota \in \{0.000001, 0.000005, 0.00001, 0.00005, \cdots, 0.5\} \cup (1 - \{0.000001, 0.000005, 0.00001, 0.00005, \cdots, 0.5\})$.

For what concerns the different models we searched the optimal hyperparameters according to what reported in the following noitemsep,topsep=0ex



(a) RAO 1 (b) RAO 2 (c) RAO 3

FIGURE 5. Scatter True/Predicted and histogram of the absolute error (logarithmic scale) RAO for the best algorithm and ι.

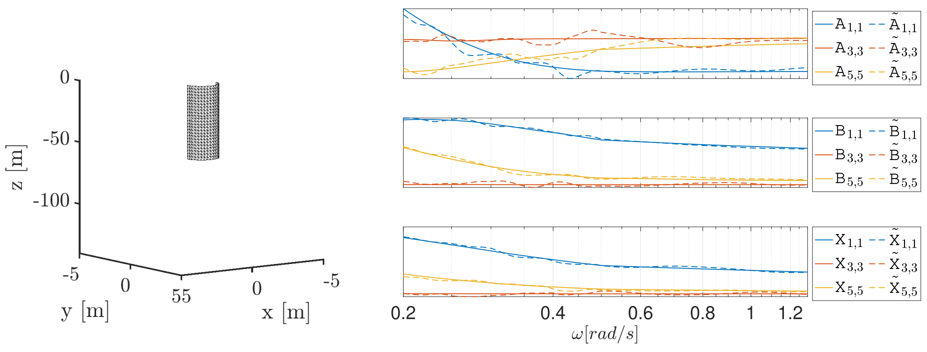
- Kernel Methods: $\lambda \in 10^{\{-6.0,-5.8,\cdots,4.0\}}$, $\gamma \in 10^{\{-6.0,-5.8,\cdots,4.0\}}$, $\epsilon \in \{0, 0.001, 0.005, 0.01, \cdots, 0.05\}$, and $\beta \in \{0.000001, 0.000005, 0.00001, 0.00005, \cdots, 0.5\} \cup (1-\{0.000001, 0.000005, 0.00001, 0.00005, \cdots, 0.5\})$;
- Neural Networks: $l \in \{1, 2, \cdots, 5\}$, $h_1, \cdots, h_l \in 10^{\{}1, 2, 3, 4, 5\}$, $\sigma_1 = \text{LIN}$, $\sigma_2 \cdots, \sigma_l \in \{\text{RELU}, \text{TANH}\}$, $\lambda \in 10^{\{-6.0,-5.8,\cdots,4.0\}}$, $\theta_1, \cdots, \theta_l \in \{0, 0.001, 0.005, 0.01, 0.05\}$, $b_s \in \{512\}$, and $l_r \in \{0.001, 0.002, 0.004, 0.008, 0.01, 0.02, 0.04, 0.08\}$;
- Ensemble Methods: $\lambda \in 10^{\{-6.0,-5.8,\cdots,4.0\}}$, $k \in \{20, 40, 80, 160, 320, 640, 1024\}$, $\eta \in \{0.001, 0.002, 0.004, 0.008, 0.01, 0.02, 0.04, 0.08\}$, $\tau \in \{10, 15, 20, 25, 30\}$, $\gamma \in \{0, 0.001, 0.005, 0.01\}$, $\varrho_s \in \{1, 0.9, 0.7\}$, $\varrho_f \in \{1, 0.5, 0.2, 0.1\}$, and $\varkappa \in \{100, 200, 400, 800, 1000, 2000, 4000, 10000\}$

It is worth remembering that the ML-based surrogate models that replace NEMOH software (see Figure 1) need to estimate 9 different quantities (9 different models to train for Kernel Methods and Ensemble Methods and optimal hyperparameters set). Consequently, all the possible hyperparameters combinations cannot be fully explored. For this reason, we just checked for 10000 random configurations inside the full search space [124].

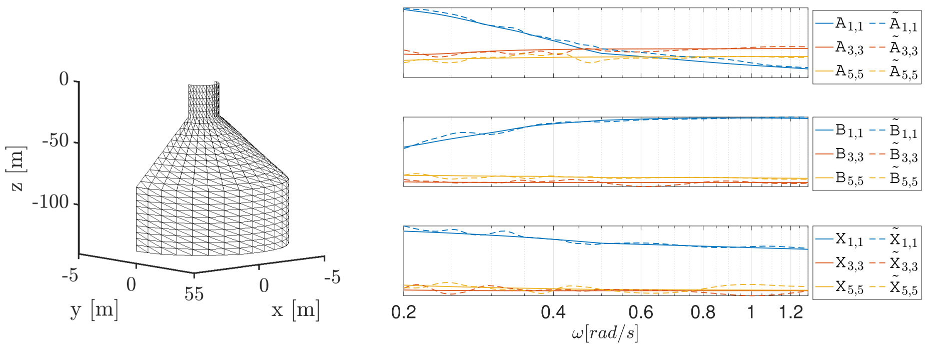
B. EXPERIMENTAL RESULTS

This section reports the results of applying the methodology described in Section V using the experimental setting described in Section VI-A over the data described in Section IV.

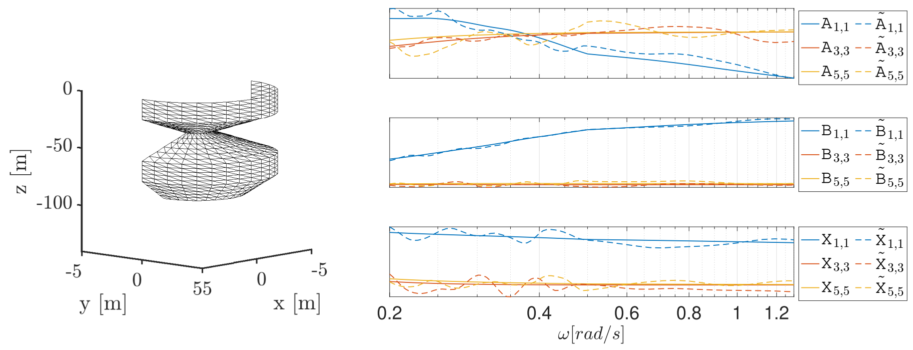
Figure 3 reports the Pareto curves resulting from Kernel Methods, Neural Networks, Ensemble Methods, and all the algorithms together. Note that both axis are presented in logarithmic scale. Blue lines correspond to the MAPEvsTIME scenario and orange to the MAPEvsMEM one. Note that `MAPE` is in percentage, `TIME` in seconds and `MEM` in megabyte. The intercepts between the red line (exploited to find the knee of the frontier) and the blue line represent the optimal ι for the MAPEvsTIME scenario: we give more importance to `TIME` with respect to `MEM` since for design purposes MAPEvsTIME is the most relevant scenario. As a matter of fact, in this framework, `MEM` is not critical as we are dealing with quantity of memory which is negligible for the current generation of computers.



(a) Geometry 1



(b) Geometry 2



(c) Geometry 3

FIGURE 6. Geometries of Figure 2. Comparison of real versus (vs) predicted of the added mass coefficients in surge $A_{1,1}$ vs $\tilde{A}_{1,1}$, heave $A_{3,3}$ vs $\tilde{A}_{3,3}$, and pitch $A_{5,5}$ vs $\tilde{A}_{5,5}$, the radiation damping matrix coefficients in surge $B_{1,1}$ vs $\tilde{B}_{1,1}$, heave $B_{3,3}$ vs $\tilde{B}_{3,3}$, and pitch $B_{5,5}$ vs $\tilde{B}_{5,5}$, and the first order wave load transfer function matrix coefficients in surge $X_{1,1}$ vs $\tilde{X}_{1,1}$, heave $X_{3,3}$ vs $\tilde{X}_{3,3}$, and pitch $X_{5,5}$ vs $\tilde{X}_{5,5}$. Best algorithm and ι.

From Figure 3 it is possible to observe how Kernel Methods tend to lose a small amount of accuracy (measured with MAPE) in order to reduce the computational requirements (measured with TIME), with respect to Neural Networks and Ensemble Methods. For what concerns the loss in MAPE to reduce the computational requirements in terms of MEM there is an unfavorable behavior for all the algorithms: performance tends to decrease significantly in order to reduce the MEM requirements. It is necessary to pay a big price in terms of MAPE in order to reduce MEM. Nonetheless, this does not represent a limitation. In fact, for all the models the actual requirements in terms of MEM is quite small (hundreds of megabyte). In this respect, it is possible to note that the best behavior is showed by Kernel Methods for which the Pareto curves are pretty smooth, in the Lipschitz sense. Instead, when it comes to look at the absolute values of MAPE, TIME, and MEM, Kernel and Ensemble Methods tend to be the ones showing the best performances in terms of MAPE while Neural Networks is the one which shows the best performance in terms of TIME and MEM.

Ensemble Methods appears to be the method showing the best compromise between the different approaches (see Figure 3(d)), with the best MAPE overall and

TIME and MEM close to the Neural Networks, which proved to be the top performing model on these metrics.

Table 5 reports a comparison between the optimal point on the Pareto curves (in particular with respect to the MAPEvsTIME one since we observed that `MEM` is not a problem in this application for all methods) for the different algorithms. Note that we did not report Table 5 for other points since this would not add any additional value to the paper.

From Table 5, it is possible to derive the same observation drawn for the Pareto curves. Ensemble Methods result to be the algorithm with the best trade-off between `MAPE`, `TIME`, and `MEM`. Note that also Kernel Methods are quite effective while Neural Networks tend to under-perform in terms of `MAPE`. Note also that Ensemble Methods and Kernel Methods demonstrate both errors below 3% and Ensemble Methods can achieve these results with a model that requires a fraction of millisecond.

Since Ensemble Methods resulted to be the model with the best trade-off, we reported, for the sake of completeness, the scatter plot of the surrogate models and the relative distribution of the absolute errors, built exploiting the latter methods, in Figure 4 and the RAO predicted based on these surrogates in Figure 5 against the one predicted using the NEMOH code (see Figure 1 for reference). We did not report the same figures for all the other models for space constraints, and we also believe that those figures would not add any additional value to the paper.

From Figures 4 and 5, it is possible to further observe the high predictive capabilities of the surrogate models in comparison to the ground truth, which in our case is the prediction obtained employing the NEMOH software. In the scatter plots presented in Figures 4, one can observe a strong qualitative agreement between the true and predicted values across various parameters. Specifically, this alignment is evident in the coefficients of the added mass matrix (for surge, heave, and pitch motions), the radiation damping matrix (for surge, heave, and pitch motions), and the 1st-order wave load transfer function matrix (also for surge, heave, and pitch motions). In the scatter plots depicted in Figures 5, the high predictive capabilities are clearly demonstrated for the RAO in surge, heave, and pitch motions. Additionally, the Gaussian-like distribution of errors on a logarithmic scale further corroborates the accuracy and quality of the learned methods. It's important to note that the models do not perform uniformly across all quantities; they excel in predicting RAO 1 while forecasting RAO 3 proves to be more challenging. This discrepancy may be attributable to the pitch dynamics (RAO 3) complexity, which is harder for the models to learn [134].

For the readers that have more expertise on the application than on the data science part, we reported, for the example geometries of Figure 2, the comparison between the RAO predictor based on NEMOH and the ones reported in Figure 6 (see Figure 1 for reference).

Figure 2 clearly demonstrates a robust agreement—both qualitative and quantitative—between the developed models' predictions and domain experts' expectations.

VII. CONCLUSION

FOWTs hold promise in addressing the increasing global energy demands and environmental concerns by expanding their deployment to areas with stronger winds in deeper waters. However, their design and construction face substantial engineering challenges due to the harsh offshore conditions and their relative immaturity compared to fixed bottom structures.

Presently, CFD stands as the primary viable solution due to its accuracy, aiding in the design and optimization processes. Unfortunately, the computational demands of CFD hinder its integration into automated workflows, leading to predominantly manually crafted pipelines.

To overcome this limitation, our study introduces and evaluates a range of Artificial Intelligence-based surrogate models. These models achieve accuracy levels closely aligned with CFD, demonstrating less than a 3% discrepancy while requiring only a fraction of the computational resources, reducing computation times from minutes to milliseconds.

Our work focuses on two key contributions. First, we utilize state-of-the-art software to predict the hydrodynamic response characteristics of floating spar-type offshore wind turbines, specifically the RAO. This software generates a dataset comprising various geometries and their associated RAOs, using an efficient method of geometry parameterization. This dataset serves to evaluate the performance of the surrogate models.

To the best of our knowledge, this study represents the first demonstration of Machine Learning-based models outperforming state-of-the-art CFD-based tools in terms of computational efficiency. Furthermore, it is the first to create a real dataset from a state-of-the-art CFD code for testing the quality of the proposed models. Consequently, we believe that our work could have a profound impact as it marks the initial step towards a framework for optimizing FOWT geometries, automating the design process, minimizing human intervention, and enabling the exploration of unconventional and previously uncharted geometries.

ACKNOWLEDGMENT

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

[1] I. Komusanac, G. Brindley, D. Fraile, and L. Ramirez, "Wind energy in Europe: 2021 statistics and the outlook for 2022–2026," Wind Europe, Brussels, Belgium, Tech. Rep. 1, 2021.

[2] T. Pignolet, "A policy framework for climate and energy in the period from 2020 to 2030," Eur. Wind Energy Assoc., Brussels, Belgium, Tech. Rep. 52014DC0015, 2015.
[3] H. Díaz and C. G. Soares, "Review of the current status, technology and future trends of offshore wind farms," *Ocean Eng.*, vol. 209, Aug. 2020, Art. no. 107381.
[4] L. Eatough, "Floating offshore wind technology and operations review," Catapult, Melbourne, VIC, Australia, Tech. Rep. 1, 2021.
[5] *FloatingWind Joint Industry Project*, Carbon Trust, London, U.K., 2018.
[6] M. Hannon, E. Topham, J. Dixon, D. McMillan, and M. Collu, "Offshore wind, ready to float? Global and U.K. trends in the floating offshore wind market," Univ. Strathclyde, Glasgow, U.K., Tech. Rep., 2019, doi: 10.17868/69501.
[7] H. Yang and Y. Zhu, "Robust design optimization of supporting structure of offshore wind turbine," *J. Mar. Sci. Technol.*, vol. 20, no. 4, pp. 689–702, Dec. 2015.
[8] M. Borg, A. Shires, and M. Collu, "Offshore floating vertical axis wind turbines, dynamics modelling state of the art. Part I: Aerodynamics," *Renew. Sustain. Energy Rev.*, vol. 39, pp. 1214–1225, Nov. 2014.
[9] M. Borg and M. Collu, "Offshore floating vertical axis wind turbines, dynamics modelling state of the art. Part III: Hydrodynamics and coupled modelling approaches," *Renew. Sustain. Energy Rev.*, vol. 46, pp. 296–310, Jun. 2015.
[10] Z. Ren, A. S. Verma, Y. Li, J. J. E. Teuwen, and Z. Jiang, "Offshore wind turbine operations and maintenance: A state-of-the-art review," *Renew. Sustain. Energy Rev.*, vol. 144, Jul. 2021, Art. no. 110886.
[11] D. Ilardi, "Data-driven solutions to enhance planning, operation and design tools in Industry 4.0 context," Ph.D. thesis, Dept. DIBRIS, Genoa, Italy, Univ. Genoa, 2023.
[12] M. Wang, C. Wang, A. Hnydiuk-Stefan, S. Feng, I. Atilla, and Z. Li, "Recent progress on reliability analysis of offshore wind turbine support structures considering digital twin solutions," *Ocean Eng.*, vol. 232, Jul. 2021, Art. no. 109168.
[13] S. Tao, Q. Xu, A. Feijóo, G. Zheng, and J. Zhou, "Nonuniform wind farm layout optimization: A state-of-the-art review," *Energy*, vol. 209, Oct. 2020, Art. no. 118339.
[14] S. Khatir, D. Boutchicha, C. Le Thanh, H. Tran-Ngoc, T. N. Nguyen, and M. Abdel-Wahab, "Improved ANN technique combined with Jaya algorithm for crack identification in plates using XIGA and experimental analysis," *Theor. Appl. Fract. Mech.*, vol. 107, Jun. 2020, Art. no. 102554.
[15] D. H. Nguyen-Le, Q. B. Tao, V.-H. Nguyen, M. Abdel-Wahab, and H. Nguyen-Xuan, "A data-driven approach based on long short-term memory and hidden Markov model for crack propagation prediction," *Eng. Fract. Mech.*, vol. 235, Aug. 2020, Art. no. 107085.
[16] H. Tran-Ngoc, S. Khatir, H. Ho-Khac, G. De Roeck, T. Bui-Tien, and M. A. Wahab, "Efficient artificial neural networks based on a hybrid metaheuristic optimization algorithm for damage detection in laminated composite structures," *Compos. Struct.*, vol. 262, Apr. 2021, Art. no. 113339.
[17] H. Tran-Ngoc, S. Khatir, T. Le-Xuan, G. De Roeck, T. Bui-Tien, and M. A. Wahab, "A novel machine-learning based on the global search techniques using vectorized data for damage detection in structures," *Int. J. Eng. Sci.*, vol. 157, Dec. 2020, Art. no. 103376.
[18] S. Khatir, S. Tiachacht, C. Le Thanh, E. Ghandourah, S. Mirjalili, and M. A. Wahab, "An improved artificial neural network using arithmetic optimization algorithm for damage assessment in FGM composite plates," *Compos. Struct.*, vol. 273, Oct. 2021, Art. no. 114287.
[19] S. Wang, H. Wang, Y. Zhou, J. Liu, P. Dai, X. Du, and M. A. Wahab, "Automatic laser profile recognition and fast tracking for structured light measurement using deep learning and template matching," *Measurement*, vol. 169, Feb. 2021, Art. no. 108362.
[20] K.-H. Lee, K.-H. Kim, D.-H. Lee, K.-T. Lee, and J.-P. Park, "Two-step optimization for wind turbine blade with probability approach," *J. Sol. Energy Eng.*, vol. 132, no. 3, pp. 034503–034508, Aug. 2010.
[21] K. Müller, M. Dazer, and P. W. Cheng, "Damage assessment of floating offshore wind turbines using response surface modeling," *Energy Proc.*, vol. 137, pp. 119–133, Oct. 2017.
[22] M. Muskulus, "Designing the next generation of computational codes for wind-turbine simulations," in *Proc. 21st Int. Offshore Polar Eng. Conf.*, 2011, pp. 86–98.
[23] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annu. Rev. Fluid Mech.*, vol. 52, no. 1, pp. 477–508, Jan. 2020.
[24] A. Pollard, L. Castillo, L. Danaila, and M. Glauser, *Whither Turbulence and Big Data in the 21st Century?* Cham, Switzerland: Springer, 2016.
[25] C. W. Rowley and S. T. M. Dawson, "Model reduction for flow analysis and control," *Annu. Rev. Fluid Mech.*, vol. 49, no. 1, pp. 387–417, Jan. 2017.
[26] J. Ling and J. Templeton, "Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty," *Phys. Fluids*, vol. 27, no. 8, pp. 1–32, Aug. 2015.
[27] A. P. Singh, S. Medida, and K. Duraisamy, "Machine-Learning-Augmented predictive modeling of turbulent separated flows over airfoils," *AIAA J.*, vol. 55, no. 7, pp. 2215–2227, Jul. 2017.
[28] R. M. M. Slot, J. D. Sørensen, B. Sudret, L. Svenningsen, and M. L. Thøgersen, "Surrogate model uncertainty in wind turbine reliability assessment," *Renew. Energy*, vol. 151, pp. 1150–1162, May 2020.
[29] S. Zhou, C. Li, Y. Xiao, and P. W. Cheng, "Importance of platform mounting orientation of Y-shaped semi-submersible floating wind turbines: A case study by using surrogate models," *Renew. Energy*, vol. 156, pp. 260–278, Aug. 2020.
[30] S. Li and L. Caracoglia, "Surrogate model Monte Carlo simulation for stochastic flutter analysis of wind turbine blades," *J. Wind Eng. Ind. Aerodyn.*, vol. 188, pp. 43–60, May 2019.
[31] J. P. Murcia, P.-E. Réthoré, N. Dimitrov, A. Natarajan, J. D. Sørensen, P. Graf, and T. Kim, "Uncertainty propagation through an aeroelastic wind turbine model using polynomial surrogates," *Renew. Energy*, vol. 119, pp. 910–922, Apr. 2018.
[32] A. D. V. Carrasco, D. J. Valles-Rosales, L. C. Mendez, and M. I. Rodriguez, "A site-specific design of a fixed-pitch fixed-speed wind turbine blade for energy optimization using surrogate models," *Renew. Energy*, vol. 88, pp. 112–119, Apr. 2016.
[33] M. Sessarego, N. Ramos-García, H. Yang, and W. Z. Shen, "Aerodynamic wind-turbine rotor design using surrogate modeling and three-dimensional viscous–inviscid interaction technique," *Renew. Energy*, vol. 93, pp. 620–635, Aug. 2016.
[34] R. Bourguet, G. Martinat, G. Harran, and M. Braza, "Aerodynamic multi-criteria shape optimization of VAWT blade profile by viscous approach," in *Wind Energy*. London, U.K.: Hindawi Limited, 2007.
[35] Z. Han, K. Zhang, W. Song, and J. Liu, "Surrogate-based aerodynamic shape optimization with application to wind turbine airfoils," in *Proc. 51st AIAA Aerosp. Sci. Meeting Including New Horizons Forum Aerosp. Expo.*, Jan. 2013, pp. 1–47.
[36] A. F. P. Ribeiro, A. M. Awruch, and H. M. Gomes, "An airfoil optimization technique for wind turbines," *Appl. Math. Model.*, vol. 36, no. 10, pp. 4898–4907, Oct. 2012.
[37] M. Atcheson, A. Garrad, L. Cradden, A. Henderson, D. Matha, J. Nichols, D. Roddier, and J. Sandberg, *Floating Offshore Wind Energy*. Cham, Switzerland: Springer, 2016.
[38] T. T. Tran and D.-H. Kim, "The coupled dynamic response computation for a semi-submersible platform of floating offshore wind turbine," *J. Wind Eng. Ind. Aerodynamics*, vol. 147, pp. 104–119, Dec. 2015.
[39] T. T. Tran and D.-H. Kim, "Fully coupled aero-hydrodynamic analysis of a semi-submersible FOWT using a dynamic fluid body interaction approach," *Renew. Energy*, vol. 92, pp. 244–261, Jul. 2016.
[40] S. Netzband, C. W. Schulz, U. Göttsche, D. F. González, and M. Abdel-Maksoud, "A panel method for floating offshore wind turbine simulations with fully integrated aero- and hydrodynamic modelling in time domain," *Ship Technol. Res.*, vol. 65, no. 3, pp. 123–136, Sep. 2018.
[41] A. Coraddu, L. Oneto, M. Kalikatzarakis, D. Ilardi, and M. Collu, "Floating spar-type offshore wind turbine hydrodynamic response characterisation: A computational cost aware approach," in *Proc. Global Oceans*, Singapore, Oct. 2020, pp. 1–8.
[42] J. N. Newman, *Marine Hydrodynamics*. Cambridge, MA, USA: MIT Press, 2018.
[43] S. Shalev-Shwartz and S. Ben-David, *Machine Learning: From Theory to Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[44] W. Zhao, "Research on the deep learning of the small sample data based on transfer learning," in *Proc. AIP Conf.*, 2017, pp. 020018–020027.
[45] Y. Wang, Q. Yao, J. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," 2019, *arXiv:1904.05046*.
[46] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[47] L. Oneto, *Model Selection and Error Estimation in a Nutshell*. Cham, Switzerland: Springer, 2020.
[48] A. Babarit and G. Delhommeau, "Theoretical and numerical aspects of the open source BEM solver NEMOH," in *Proc. Eur. Wave Tidal Energy Conf.*, 2015, pp. 01198800–01198812.
[49] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev, "Why the Monte Carlo method is so important today," *WIREs Comput. Statist.*, vol. 6, no. 6, pp. 386–392, Nov. 2014.
[50] J. M. Jonkman and K. Shaler, "Fast.Farm user's guide and theory manual," Nat. Renew. Energy Lab., Golden, CO, USA, Tech. Rep. NREL/TP-5000-78485, 2018.
[51] B. Jonkman and N. Kelley, "TurbSim: A stochastic, full-field, turbulence simulator primarialy for use with inflowwind/aerodyn-based simulation tools," Nat. Renew. Energy Lab. (NREL), Golden, CO, USA, Tech. Rep. v2, 2016.
[52] J. Jonkman, "The new modularization framework for the FAST wind turbine CAE tool," in *Proc. 51st AIAA Aerosp. Sci. Meeting Including New Horizons Forum Aerosp. Expo.*, Jan. 2013, pp. 1–28.
[53] C. Bak, F. Zahle, R. Bitsche, T. Kim, A. Yde, L. C. Henriksen, M. H. Hansen, J. Blasques, M. Gaunaa, and A. Natarajan, "The DTU 10-MW reference wind turbine," in *Proc. Danish Wind Power Res.*, 2013, pp. 1–22.
[54] F. Sandner, W. Yu, D. Matha, J. Azcona, X. Munduate, E. Grela, S. Voutsinas, and A. Natarajan, "INNWIND. EU D4. 33: Innovative concepts for floating structures," Univ. Stuttgart, Stuttgart, Germany, Tech. Rep. 4.3.3, 2014.
[55] T. J. Larsen and A. M. Hansen, "How 2 HAWC2, the user's manual," Tech. Univ. Denmark, Kongens Lyngby, Denmark, Tech. Rep. 1597, 2007.
[56] N. Ramos-García, J. N. Sørensen, and W. Z. Shen, "Three-dimensional viscous-inviscid coupling method for wind turbine computations," *Wind Energy*, vol. 19, no. 1, pp. 67–93, Jan. 2016.
[57] Siemens. (2021). *Star CCM + User's Manual*. [Online]. Available: http://www.cd-adapco.com/products/star-ccm/documentation
[58] Ansys. (2011). *Ansys Fluent User's Manual*. [Online]. Available: http://www.ansys.com/products/fluids/ansys-fluent
[59] P. A. Berthelsen and I. Fylling, "Optimization of floating support structures for deep water wind turbines," in *Proc. Conf. EWEA*, 2012, pp. 1618–1630.
[60] M. Hall, B. Buckham, and C. Crawford, "Evolving offshore wind: A genetic algorithm-based support structure optimization framework for floating wind turbines," in *Proc. MTS/IEEE Oceans*, Jun. 2013, pp. 1–10.
[61] M. Leimeister, A. Kolios, M. Collu, and P. Thomas, "Design optimization of the OC3 phase IV floating spar-buoy, based on global limit states," *Ocean Eng.*, vol. 202, Apr. 2020, Art. no. 107186.
[62] J. M. Hegseth, E. E. Bachynski, and J. R. R. A. Martins, "Integrated design optimization of spar floating wind turbines," *Mar. Struct.*, vol. 72, Jul. 2020, Art. no. 102771.
[63] L. Chinmoy, S. Iniyan, and R. Goic, "Modeling wind power investments, policies and social benefits for deregulated electricity market—A review," *Appl. Energy*, vol. 242, pp. 364–377, May 2019.
[64] J. Jonkman, "Definition of the floating system for phase IV of OC3," Nat. Renew. Energy Lab., Golden, CO, USA, Tech. Rep. NREL/TP-500-47535, 2010.
[65] M. Borg and M. Collu, "A comparison between the dynamics of horizontal and vertical axis offshore floating wind turbines," *Phil. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 373, no. 2035, Feb. 2015, Art. no. 20140076.
[66] O. Anaya-Lara, J. O. Tande, K. Uhlen, and K. Merz, *Offshore Wind Energy Technology*. Hoboken, NJ, USA: Wiley, 2018.
[67] J. A. Myers, *Handbook of Equations for Mass and Area Properties of Various Geometrical Shapes*. China Lake, CA, USA: Nav. Ordnance Test Station, 1962.
[68] J. M. Journée and W. W. Massie, "Offshore hydromechanics," Delft Univ. Technol., Delft, The Netherlands, Tech. Rep. 1, 2000.
[69] Y. Liu, S. Li, Q. Yi, and D. Chen, "Developments in semi-submersible floating foundations supporting wind turbines: A comprehensive review," *Renew. Sustain. Energy Rev.*, vol. 60, pp. 433–449, Jul. 2016.
[70] X. Wang, X. Zeng, J. Li, X. Yang, and H. Wang, "A review on recent advancements of substructures for offshore wind turbines," *Energy Convers. Manage.*, vol. 158, pp. 103–119, Feb. 2018.
[71] C. C. Aggarwal, *Data Mining: The Textbook*. Cham, Switzerland: Springer, 2015.
[72] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3133–3181, 2014.
[73] M. Wainberg, B. Alipanahi, and B. J. Frey, "Are random forests truly the best classifiers?" *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 3837–3841, 2016.
[74] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
[75] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[76] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.
[77] L. Rokach, *Ensemble Learning: Pattern Classification Using Ensemble Methods*. Singapore: World Scientific, 2019.
[78] D. H. Wolpert, "The supervised learning no-free-lunch theorems," in *Soft Computing and Industry: Recent Applications*, 2002.
[79] F. Bargagli-Stoffi, G. Cevolani, and G. Gnecco, "Should simplicity be always preferred to complexity in supervised machine learning?" in *Proc. Int. Conf. Mach. Learn., Optim., Data Sci.* Cham, Switzerland: Springer, 2020, pp. 55–59.
[80] D. Lauc, "Machine learning and the philosophical problems of induction," in *Guide to Deep Learning Basics: Logical, Historical and Philosophical Perspective*. Cham, Switzerland: Springer, 2020.
[81] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Cham, Switzerland: Springer, 2013.
[82] J. Franklin, "The elements of statistical learning: Data mining, inference and prediction," *Math. Intelligencer*, vol. 27, no. 2, pp. 83–85, Mar. 2005.
[83] A. Doan, "Human-in-the-loop data analysis: A personal perspective," in *Proc. Workshop Hum.-Loop Data Anal.*, Jun. 2018, pp. 1–6.
[84] F. Girardin and N. Lathia, "When user experience designers partner with data scientists," in *Proc. AAAI Spring Symp. Ser.*, 2017, pp. 376–381.
[85] B. Scholkopf, "The kernel trick for distances," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 301–307.
[86] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?" *Neural Comput.*, vol. 16, no. 5, pp. 1063–1076, May 2004.
[87] A. Bakushinsky and A Goncharsky, *Ill-Posed Problems: Theory and Applications*. Cham, Switzerland: Springer, 2012.
[88] A. N. Tikhonov and V. Y. Arsenin, *Methods for Solving Ill-Posed Problems*. Moscow, Russia: Nauka, 1979.
[89] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, Feb. 1970.
[90] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc., B, Methodol.*, vol. 58, no. 1, pp. 267–288, Jan. 1996.
[91] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statist. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004.
[92] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[93] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *J. Amer. Stat. Assoc.*, vol. 101, no. 473, pp. 138–156, Mar. 2006.
[94] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Proc. Int. Conf. Comput. Learn. theory*, 2001, pp. 416–426.
[95] A. Zheng and A. Casari, *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. New York, NY, USA: O'Reilly Media, Inc., 2018.
[96] P. Duboue, *The Art of Feature Engineering: Essentials for Machine Learning*. Cambridge, U.K.: Cambridge Univ. Press, 2020.
[97] M. G. Genton, "Classes of kernels for machine learning: A statistics perspective," *J. Mach. Learn. Res.*, vol. 2, pp. 299–312, Dec. 2001.
[98] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with Gaussian kernel," *Neural Comput.*, vol. 15, no. 7, pp. 1667–1689, Jul. 2003.
[99] 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.
[100] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," in *Proc. Int. Conf. Comput. Sci. Technol.*, 2021, pp. 124–133.
[101] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," 2014, *arXiv:1412.6830*.
[102] G. Marra, D. Zanca, A. Betti, and M. Gori, "Learning activation functions by means of kernel based neural networks," in *Proc. Int. Conf. Italian Assoc. Artif. Intell.*, 2019, pp. 418–430.

[103] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. control, signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.

[104] H. N. Mhaskar and T. Poggio, "Deep vs. Shallow networks: An approximation theory perspective," *Anal. Appl.*, vol. 14, no. 6, pp. 829–848, Nov. 2016.

[105] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowl.-Based Syst.*, vol. 212, Jan. 2021, Art. no. 106622.

[106] H. Jin, Q. Song, and X. Hu, "Auto-Keras: An efficient neural architecture search system," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1946–1956.

[107] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[108] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[109] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3668–3681, Aug. 2020.

[110] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On empirical comparisons of optimizers for deep learning," 2019, *arXiv:1910.05446*.

[111] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[112] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, pp. 1–12, Mar. 2018.

[113] F. Galton, "Vox populi (the wisdom of crowds)," *Nature*, vol. 75, no. 7, pp. 450–451, 1907.

[114] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

[115] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[116] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[117] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. Boca Raton, FL, USA: CRC Press, 1994.

[118] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[119] R. Blaser and P. Fryzlewicz, "Random rotation ensembles," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 126–151, 2016.

[120] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[121] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[122] J. H. Friedman, "Stochastic gradient boosting," *Comput. Statist. Data Anal.*, vol. 38, no. 4, pp. 367–378, Feb. 2002.

[123] A. Vabalas, E. Gowen, E. Poliakoff, and A. J. Casson, "Machine learning algorithm validation with a limited sample size," *PLoS ONE*, vol. 14, no. 11, Nov. 2019, Art. no. e0224365.

[124] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. 2, 2012.

[125] F. Schöpfer, A. K. Louis, and T. Schuster, "Nonlinear iterative methods for linear ill-posed problems in Banach spaces," *Inverse Problems*, vol. 22, no. 1, pp. 311–329, Feb. 2006.

[126] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units," 2016, *arXiv:1611.01491*.

[127] A. Fred Agarap, "Deep learning using rectified linear units (ReLU)," 2018, *arXiv:1803.08375*.

[128] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, *Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies*. IEEE Press, NJ, USA: IEEE Press, 2001.

[129] G. B. Goh, N. O. Hodas, and A. Vishnu, "Deep learning for computational chemistry," *J. Comput. Chem.*, vol. 38, no. 16, pp. 1291–1307, Jun. 2017.

[130] H. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*. Cham, Switzerland: Springer, 2003.

[131] A. Ishizaka and P. Nemery, *Multi-Criteria Decision Analysis: Methods and software*. Hoboken, NJ, USA: Wiley, 2013.

[132] P. Ngatchou, A. Zarei, and A. El-Sharkawi, "Pareto multi objective optimization," in *Proc. Int. Conf. Intell. Syst. Appl. Power Syst.*, 2005, pp. 1–8.

[133] X. Zhang, Y. Tian, and Y. Jin, "A knee point-driven evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 761–776, Dec. 2015.

[134] R. A. Ibrahim and I. M. Grace, "Modeling of ship roll dynamics and its coupling with heave and pitch," *Math. Problems Eng.*, vol. 2010, pp. 1–32, Jan. 2010.

DAVIDE ILARDI received the bachelor's and master's degrees in electronic engineering and the Ph.D. degree in computer science and systems engineering. His interests have evolved during his academic experience. He has developed a real passion for AI and machine learning fields, focused on the development of data-driven solutions to enhance planning, operation and design tools in industry 4.0 context to explore machine learning solutions and, at the same time, to increase the computational performances of those industrial tools that necessitates to keep up-to-date to bring further value.

MILTIADIS KALIKATZARAKIS received the M.Eng. degree in naval architecture and marine engineering from the National Technical University of Athens, Athens, Greece, in 2013, and the M.Sc. degree in marine technology from the Delft University of Technology, Delft, The Netherlands, in 2017. He is currently pursuing the Ph.D. degree with the Department of Naval Architecture, Ocean and Marine Engineering (NAOME), University of Strathclyde, Glasgow, U.K. He joined the Research and Development Department of Damen Naval, Vlissingen, The Netherlands, in 2017, and he has been involved in data analytics, modeling, and design optimization in the maritime industry.

LUCA ONETO (Senior Member, IEEE) was born in Rapallo, Italy, in 1986. He received the B.Sc. and M.Sc. degrees in electronic engineering from the University of Genoa, Italy, in 2008 and 2010, respectively, and the Ph.D. degree from the School of Sciences and Technologies for Knowledge and Information Retrieval, University of Genoa, in 2014. His Ph.D. dissertation was titled, "Learning Based On Empirical Data." In 2017, he obtained the Italian National Scientific Qualification for the role of an Associate Professor in computer engineering, and in 2018, he obtained the one in computer science. He was an Assistant Professor in computer engineering with the University of Genoa, from 2016 to 2019. In 2018, he was the Co-Founder of the spin-off ZenaByte s.r.l. In 2019, he obtained the Italian National Scientific Qualification for the role of a Full Professor in computer science and computer engineering. In 2019, he became an Associate Professor in computer science with the University of Pisa. He is currently an Associate Professor in computer engineering with the University of Genoa. He has been involved in several H2020 projects (S2RJU, ICT, and DS) and he has been awarded with the Amazon AWS Machine Learning and Somalvico (a Best Italian Young AI Researcher) Awards. His first main topic of research is the statistical learning theory with particular focus on the theoretical aspects of the problems of (semi) supervised model selection and error estimation. His research interests include data science with particular reference to the problem of trustworthy AI and the solution of real world problems by exploiting and improving the most recent learning algorithms and theoretical results in the fields of machine learning and data mining.

MAURIZIO COLLU is currently a Professor in offshore renewable energy engineering with the University of Strathclyde, U.K., and the Chair of the ITTC Specialist Committee on Ocean Renewable Energy (2021–2024). His area of expertise is applied mechanics, focusing on multidisciplinary model of dynamics. He is internationally recognized as a world-level leading expert in offshore floating wind systems and offshore multipurpose platforms, specializing in their design and analysis—having directly managed as lead £ 1.7M of research in this field and contributing to the management of projects for a total of £ 17.8M.

ANDREA CORADDU (Member, IEEE) was born in Pietrasanta, in 1979. He received the Laurea degree in naval architecture and marine engineering from the University of Genoa, Italy, in 2006, and the Ph.D. degree from the School of Fluid and Solid Mechanics, University of Genoa, in 2012. His Ph.D. dissertation was titled, "Modeling and Control of Naval Electric Propulsion Plants." Currently, he is an Associate Professor in intelligent and sustainable energy systems with the Maritime and Transport Technology Department, Delft University of Technology, Delft, The Netherlands. He has been an Associate Professor with the Department of Naval Architecture, Ocean and Marine Engineering, University of Strathclyde, from October 2020 to August 2021. His relevant professional and academic experiences, include working as an Assistant Professor with the University of Strathclyde, a Research Associate with the School of Marine Science and Technology, Newcastle University, a Research Engineer as part of the DAMEN Research and Development Department based in Singapore. He is also a Postdoctoral Research Fellow with the University of Genoa. He has been involved in a number of successful grant applications from research councils, industry, and international governmental agencies focusing on the design, integration, and control of complex marine energy and power management systems enabling the development of next-generation complex and multi-function vessels that can meet the pertinent social challenges regarding the environmental impact of human-related activities.