Machine Learning Based Modelling and Control of Wind Turbine Structures and Wind Farm Wakes

by

Jincheng Zhang

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

School of Engineering

July 2021

Contents

List of Tables

List of Figures

Acknowledgements

First of all, I would like to thank my PhD supervisor, Professor Xiaowei Zhao for his full support and guidance throughout my PhD studies. His vision in this emerging interdisciplinary field involving machine learning, fluid dynamics, control theory and renewable energy has guided me through this exciting research journey. Also, I acknowledge that all the works presented in this thesis have received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 765579. I also acknowledge the Scientific Computing Research Technology Platform (SCRTP) at the University of Warwick for providing High-Performance Computing resources.

I would like to thank my dear colleagues at the Engineering lunch group. They are Alfi, Eleanor, Jacqui, Joe, Lorenzo, Matt, Natalie, Punit, Sarah, Sam, Sam 2.0, and Rhys. The nice lunch and pub time we spent together has been great. To Alfi, thanks for the mobile games nights we have spent together during the pandemic.

I would like to thank my dear colleagues in Intelligent Control & Smart Energy (ICSE) Research Group. They are Chang, Hongyang, Jingjie, Mohamed, Pengyuan, Shuyue, Xing, Yangming, Yexin, Yinan, and Zhaolong. To Hongyang, Pengyuan, Xing and Yinan, thanks for the great time we spent together playing Mahjong.

I would like to thank my dear colleagues in ConFlex project. They are Andrea, Arturo, Borjan, Charlotte, Gaston, Juan, Marc, Mir, Nathanael, Pei, Pietro, Sahar, Shantanu, and Vikram. It has been a great pleasure to join the same project with you guys. The time we spent together at Bilbao and San Sebastian has been great.

I would like to thank my parents Juntao and Liqin, and my wife Meng. It is your love that keeps supporting and motivating me throughout this exciting journey. To Meng, mon amour, looking forward to sharing the challenges and joys with you in the journeys ahead.

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree.

Parts of this thesis have been published by the author:

Journal (Peer Reviewed)

(1) **J. Zhang**, X. Zhao and X. Wei, Reinforcement learning-based structural control of floating wind turbines, **IEEE Transactions on Systems, Man, and Cybernetics: Systems**(2020), DOI: 10.1109/TSMC.2020.3032622.

(2) **J. Zhang** and X. Zhao, Quantification of parameter uncertainty in wind farm wake modeling, **Energy** 196 (2020) 117065.

(3) **J. Zhang** and X. Zhao, A novel dynamic wind farm wake model based on deep learning, **Applied Energy**, 277 (2020) 115552.

(4) **J. Zhang** and X. Zhao, Machine-learning-based surrogate modeling of aerodynamic flow around distributed structures, **AIAA Journal** 59 (3) (2021) 868–879.

(5) **J. Zhang** and X. Zhao, Spatiotemporal wind field prediction based on physics-informed deep learning and LIDAR measurements, **Applied Energy** 288 (2021) 116641.

(6) **J. Zhang** and X. Zhao, Three-dimensional spatiotemporal wind field reconstruction based on physics-informed deep learning, **Applied Energy** 300 (2021) 117390.

(7) **J. Zhang** and X. Zhao, Wind farm wake modeling based on deep convolutional conditional generative adversarial network, journal paper draft (2021), under re-

view.

Conference (Peer Reviewed)

(1) **J. Zhang**, X. Zhao, and X. Wei, Data-driven structural control of monopile wind turbine towers based on machine learning, Proceedings of the 21st IFAC World Congress, Berlin, Germany, July 2020.

All the works in the above papers are originated, conducted, and investigated by me, including conceptualization, data curation, formal analysis, investigation, methodology, project administration, software, validation, visualization, and writing of the original draft & review.

During my PhD study, the collaborations with co-workers have resulted in the following publications, which are not included in this thesis. As these works are closely related to this thesis, they are listed below for references.

Journal (Peer Reviewed)

(1) H. Dong, **J. Zhang**, and X. Zhao, Intelligent Wind Farm Control via Deep Reinforcement Learning and High-Fidelity Simulations, **Applied Energy** 292 (2021) 116928.

(2) H. Dong, X. Zhao, B. Luo, and **J. Zhang**, Robust Deep Reinforcement Learning with Application in Structural Control of Floating Wind Turbines, journal paper draft (2021), under review.

Abstract

With the fast development of wind energy, new technological challenges emerge, which calls for new research efforts to further reduce the cost of wind power. A lot of efforts have been spent to tackle the modelling and control of wind turbines and wind farms. However, big research gaps still exist due to the complexity and strong nonlinearity of the underlying structural and fluid systems. On the other hand, machine learning (ML), which is very powerful in handling complex and nonlinear systems, is developing very fast in the past years. Therefore, this thesis aims to tackle the modelling and control issues arising from the fast-developing wind industry, based on both traditional methods (including structural mechanics, control engineering, fluid dynamics, and scientific computing) and ML (including reinforcement learning, supervised ML, dimensionality reduction, generative adversarial network, and physics-informed deep learning).

First, at the turbine level, mitigation of dynamic response of a floating wind turbine using active tuned mass dampers is investigated, where a reinforcement learning algorithm is employed and a neural network structure is designed to realize the employed algorithm. Second, at the farm level, novel static and dynamic wind farm wake models are developed by proposing novel ML-based surrogate modelling methods for distributed fluid systems and then training the model based on high-fidelity CFD database generated by large eddy simulations. Third, the prediction of the spatiotemporal wind field in the whole domain in front of a wind turbine is investigated by combining data (i.e. LIDAR measurements at sparse locations) and physics (i.e. Navier-Stokes equations) in a unified manner via physics-informed deep learning. The results presented in this thesis fully demonstrate the great performance of the proposed structural controllers, the great accuracy, efficiency & robustness of the developed wind farm models, and the great accuracy of the full spatiotemporal wind field predictions.

Abbreviations

ADM	Actuator Disk Method
ADP	Adaptive Dynamic Programming
AE	Auto-Encoder
ALM	Actuator Line Method
BTLCD	Bidirectional Tuned Liquid Column Damper
CFD	Computational Fluid Dynamics
CGAN	Conditional Generative Adversarial Network
DC-CGAN	Deep Convolutional Conditional Generative Adversarial Network
DC-GAN	Deep Convolutional Generative Adversarial Network
DNS	Direct Numerical Simulation
DoF	Degree of Freedom
DWM	Dynamic Wake Meandering
FAST	Flow Analysis Software Toolkit
FLORIS	FLOw Redirection and Induction in Steady-state
FSTI	Free-Stream Turbulence Intensity
FTCS	Forward-Time Central-Space
GAN	Generative Adversarial Network
GPR	Gaussian Process Regression
HMD	Hybrid Mass Damper
HPC	High-Performance Computing
ICA	Independent Component Analysis
LES	Large Eddy Simulation

LIDAR	LIght Detection And Ranging
LoS	Line-of-Sight
LSTM	Long Short-Term Memory
MAP	Maximum A Posteriori
MCMC	Markov Chain Monte Carlo
ML	Machine Learning
MRMSE	Mean value of the Root Mean-Squared Errors
MSE	Mean-Squared Error
NN	Neural Network
NREL	National Renewable Energy Laboratory
NS	Navier-Stokes
PCA	Principal Component Analysis
PDE	Partial Differential Equation
PDF	Probability Density Function
PI-ADP	Policy Iteration Approximate Dynamic Programming
PINN	Physics-Informed Neural Network
POD	Proper Orthogonal Decomposition
QoI	Quantity of Interest
RANS	Reynold-Averaged Navier-Stokes
RL	Reinforcement Learning
RMSE	Root Mean-Squared Error
RNN	Recurrent Neural Network
SD	Standard Deviation
SOWFA	Simulator fOr Wind Farm Applications
SVD	Singular Value Decomposition
TLD	Tuned Liquid Damper
TMD	Tuned Mass Damper
UKF	Unscented Kalman Filter
UQ	Uncertainty Quantification
VI-ADP	Value Iteration Approximate Dynamic Programming

Chapter 1

Introduction

As one of the most important sustainable energy resources, wind energy has been investigated extensively all over the world. The recent decades have seen substantial growth of wind power, e.g. the global wind power installation has surpassed 651 GW at the end of 2019 [1] while this figure was just 74GW back in 2006 [2]. In order to achieve the net-zero target in the coming decades, sustained and accelerated growth of wind power is expected.

With the fast development of wind energy (such as the design of larger wind turbine blades to further increase the power capture efficiency and the development of floating wind technology for deep water sites), new technological challenges emerge. First, at the individual turbine level, due to the installation limitations of the land-based wind turbines and on the other hand the high quality of the offshore wind, more and more wind turbines are being constructed offshore [3], including both the monopile wind turbines in shallow water site and the floating wind turbines in deep water sites where the fix-bottom structures become economically infeasible. However, the strong wind and wave conditions in offshore environment have a great impact on the fatigue loads on the wind turbine structures [3]. Therefore their load mitigation is of great importance for reducing maintenance costs and increasing lifespan. Second, at the wind farm level, the wake interactions between individual wind turbines within a wind farm have a large impact on the farm's overall performance [4]. As the turbine rotor wakes are characterized by reduced wind speed and an increased turbulence level & wind shear, the downstream turbines operating in the wakes usually generate less power and experience more severe structural loads than the ones operating in freestream wind. Therefore the modelling and control of wind farm wake flows are important for achieving the optimal power generations of wind farms. Third, the spatiotemporal variability of the intermittent and chaotic

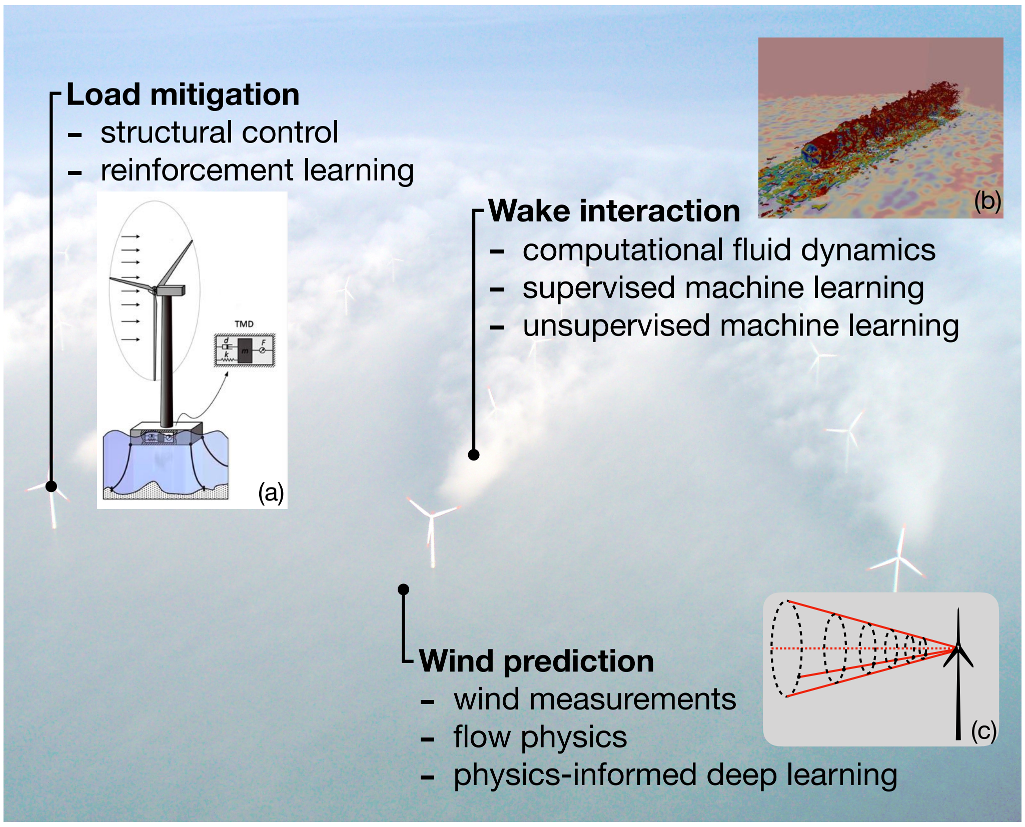


Figure 1.1: The illustration of the modelling and control issues investigated in this thesis and the overview of the proposed methods. Background: the Horns Rev Offshore Wind Farm (photo by Christian Steiness); (a) the illustration of a floating turbine structural system (figure adapted from [9]); (b) the illustration of wind turbine wake flows, generated by CFD simulations; (c) the illustration of wind measurements by turbine-mounted LIDAR.

wind poses great challenges for wind industry, in the scenarios such as the aforementioned load mitigation of turbine structures and the wind farm modelling & control, as well as in other research scenarios such as the integration of wind power into the power grid [5,6] and the wind resource assessment [7,8]. Therefore, detailed and accurate wind field predictions are of great importance. The aforementioned challenges, which are the main focus of this thesis, are summarized in Figure 1.1.

A lot of research efforts have been spent to tackle these challenges, based on the advancements in structural mechanics, fluid dynamics, control engineering, and scientific computing. However, due to the complexity and strong nonlinearity of the underlying structural and fluid systems, a big gap still exists and new modelling & control methods are still urgently required. On the other hand, the field of machine learning (ML), including supervised ML, unsupervised ML and reinforcement learning (RL), is developing very fast and has seen great progress in the past few years. The advancement of ML is also revolutionizing other fields including both engineering applications and fundamental research, such as in renewable energy systems [10], transportation systems [11], fluid mechanics [12], and biology & medicine research [13]. As ML (in particular deep learning [14]) is very powerful in handling complex and nonlinear systems, it brings new opportunities in tackling the technological challenges arising from wind industry.

Therefore, this thesis aims to tackle the modelling and control issues emerging from the fast-developing wind industry, as shown in Figure 1.1, by employing and designing new methods based on the recent progress in ML research while taking advantage of the traditional approaches in structural mechanics, control engineering, fluid dynamics, and scientific computing. Specifically, in this thesis, (a) a novel structural control design approach is proposed for the load mitigation of floating turbine structures based on traditional structural control and RL; (b) novel wind farm wake models are developed based on computational fluid dynamics (CFD) and supervised & unsupervised ML; and (c) spatiotemporal wind field prediction approaches are developed based on sparse measurements, flow physics and physics-informed deep learning. The overview of the proposed methods is summarized in Figure 1.1.

In the rest part of this chapter, the literature review on the modelling and control of wind turbine structures and wind farm wakes is given in 1.1. Then the literature review on the related ML approaches is given in 1.2. Based on these literature reviews, the motivations and the original contributions of this thesis are described in Section 1.3. Finally the thesis layout is presented in Section 1.4.

1.1 Modelling and Control of Wind Turbine Structures and Wind Farm Wakes

1.1.1 Wind Turbine Control

With the global wind sector moving further towards offshore [1], the modelling and control of offshore wind turbines is becoming a hot topic. The foundations of offshore wind turbines can be classified as the fix-bottom and the floating ones. The fix-bottom turbines are installed in shallow water sites while the floating turbines are constructed in deep water sites where the fix-bottom structures become economically infeasible [3]. The fatigue loads on floating wind turbines are much more severe than the fix-bottom ones, due to the platform motions caused by the significant external disturbances (i.e. strong wind and waves) [3].

One approach to mitigate the structural load of wind turbines is to make use of the rotor thrust as the restoring force to stabilize the turbine structures, which can be achieved through either turbine blade pitch control [15] (including individual pitch control [16] and collective pitch control [17, 18]) or torque control [19]. However, this approach may interfere with the nominal power generation. Another approach is to install additional control devices, such as Tuned Mass Dampers (TMDs) [20] and Tuned Liquid Dampers (TLDs), to dampen the platform/tower vibrations directly without interfering with power generation. In [21], passive TMDs were investigated for the structural control of both monopile turbines and floating turbines. Further study considered the use of multiple TMDs [22] and different modelling approaches for monopile turbines [23] and floating turbines [24]. As for TLDs, they were investigated for the structural control of wind turbines in [25]. Further studies investigated the modelling and optimal design of TLDs [26], the semi-active control approach [27], and the use of bidirectional tuned liquid column damper (BTLCD) [28].

The performance of a TMD can be further improved by adding active force control to it, which is referred as Hybrid Mass Dampers (HMDs) [29]. The existing works on active structural control of floating wind turbines by using HMDs are rather limited. In [30], structural control of a floating barge-type wind turbine was investigated, where an HMD was positioned in the turbine nacelle to reduce the loads. A limited degree of freedom (DoF) model was constructed through the system identification procedure, then a H_∞ multivariable loop shaping controller was designed. The paper [31] further investigated the effects of both actuator dynamics and control-structure interaction on the active control of floating wind turbines. In [9], load mitigation of floating wind turbines by an HMD installed on the platform was investigated. A linear design model was first identified, then a generalized H_∞ method was employed to optimize control gains, which achieved good performance under normal wind and wave conditions. However, this method was not able to work on extreme wind and wave conditions. In [32], a contact nonlinear modelling method for barge-type floating wind turbines was presented, where a stroke-limited HMD was included. The HMD was installed in the turbine's nacelle and a state-feedback linear quadratic regulator controller was proposed for the active structural control.

The control designs in the aforementioned works were all based on linear models, whether they were formulated based on physical principles or identified by synthetic data. However, the motion of floating wind turbines in the offshore environment can be quite complex and the turbine position can be quite far away from equilibrium, due to the extreme wind/waves loads. These control approaches thus work mainly in the normal wind/wave conditions. Therefore, a control strategy that can take account of the nonlinear dynamics of floating wind turbines is urgently required.

1.1.2 Wind Farm Wake Modelling

Wake interactions have a great impact on the overall performance of wind farms. For example, the experimental investigation [4] showed that the downstream turbine's power loss due to the wake effects could be up to 46% compared to the power generation in the designed wind condition. In order to mitigate the wake effects, turbine layout is usually optimized during the design phase [33, 34] while various control techniques are proposed for the operation phase to steer the wake away from the downstream turbines, which include turbine yaw control, individual pitch control, and tilt-based control [35]. Therefore, wake modelling is of great importance in order to take wake interactions into account in the optimal design and control of wind farms.

Wake Models

A range of wake models have been developed in the literature [36], including the low-fidelity analytical models, the medium-fidelity dynamic models, and the high-fidelity CFD models.

The most widely used wake models are the low-fidelity analytical models, such as the Jensen Park model [37,38], the Frandsen model [39], the FLOw Redirection and Induction in Steady-state (FLORIS) model [40], the 3D wake model [41,42], and the models developed in [43,44]. These models are formulated analytically and are very fast to evaluate, which makes them suitable for wind farm layout optimization. The further development of analytical models is still an active area, such as taking into account the turbulence effect [33, 45], modelling yaw effects [46–48], modelling background flow effects [49], considering the expansion of physical wake boundary [50], and incorporating uncertainty based on high-fidelity data [51]. However, as these models are static, they are mainly used for optimizing static quantities such as mean power generations. The dynamic features of wind turbine wakes are missing in these models.

Most investigations of unsteady wakes are carried out using high-fidelity CFD models, such as the studies of the Lillgrund wind farm in [52] and the Horns Rev offshore wind farm in [53]. The high-fidelity CFD models solve the filtered Navier-Stokes (NS) equations using numerical methods with the turbine modelled by the actuator disk method (ADM) [52, 54, 55] or the actuator line method (ALM) [40, 56, 57]. The comparisons of the ADM and ALM methods in wind farm simulations were also investigated, by using the PALM model [58], the UTDWF model [59], and the model developed in [60]. The further development of wind farm solvers is still an active area, such as the Nalu-Wind solver in [61]. Although these high-fidelity models can capture the detailed 3D wake dynamics, such as wake recovery and wake meandering, they are time-consuming and expensive to run. For instance, in [40] about 60 hours of distributed computation with 512 processors on high-performance computing (HPC) clusters were used for 1000s large eddy simulation (LES) of a 3km $\times$ 3km wind farm with 6 turbines. The requirement of long simulation time and enormous computing resources makes high-fidelity models not suitable for control design purposes. In the existing literature, there are also a few medium-fidelity dynamic models, such as the dynamic wake meandering (DWM) in [62], WFSim in [63], and the continuous-time dynamic wake model in [64]. The development of such control-oriented dynamic wake models is becoming very active now.

Uncertainty Quantification of Wake Models

Currently, the low-fidelity wake models are still the main tool in wind industry for wake predictions, though they can not predict the detailed flow dynamics and careful calibration of the empirical model parameters is often needed to increase the prediction accuracy [65]. The underlying uncertainty of analytical wake models needs to be rigorously quantified in order to achieve reliable wake predictions.

The input uncertainty and model uncertainty are the two main sources of uncertainty in wind turbine wake predictions. The former has been investigated in the literature. In [66], wind direction uncertainty was investigated and its impact on predicting turbine power generation was evaluated. The results showed the inclusion of direction uncertainty improved the agreement between the power predictions of analytical wake models and measurement data. In [67], Jensen wake model was employed with the consideration of the inflow direction uncertainty to predict the wake profile behind wind turbines and the results showed a better match between the predicted wake profiles and measurement data. Recently, the inclusion of uncertainty in active wake control is also receiving attention [68, 69]. However, all these work only considered the input uncertainty and did not consider the model uncertainty.

The model uncertainty for a general computer model can be classified into parameter uncertainty, model inadequacy, residual variability and code uncertainty according to the Bayesian uncertainty quantification (UQ) framework [70]. Among them, the parameter uncertainty and model inadequacy are the two most important uncertainty sources. The former arises from the lack of knowledge of the heuristic model parameters and the latter represents the discrepancy between the true physical values and the model output at the optimal model parameters. In recent years, a lot of research attention has been paid to the parameter uncertainty of fluid dynamics such as boundary layer flows [71–73], channel flows [74], transitional flows [75], compressible jet-in-crossflow [76], etc. The parameter uncertainty of wake models in wind farm simulations has not yet been investigated in the literature, which will be rigorously quantified in this thesis. This UQ study of traditional wake model is also very useful to demonstrate the need of developing novel wake models for improved wake predictions.

1.1.3 Wind Field Predictions

The detailed wind field information is of great interest for wind applications, such as in developing strategies on the wind resource assessment and the monitoring & control of wind turbine/farm. For example, the control of wind turbines based on detailed flow structures was studied in [77] which showed the consideration of detailed wind information (i.e. the coherent flow structures in the incoming wind) could improve the control performance significantly. In order to obtain such information, wind measurement technologies and wind prediction approaches have been investigated extensively.

Wind measurement technologies, such as light detection and ranging (LIDAR) [78], have been developed in the past decades. Extensive research efforts have since then been spent in the measurement analysis of LIDAR [79,80] and their applications in wind turbine control [81,82] and wind resource assessment [7,83]. However, LIDAR can only provide wind speed measurements at sparse spatial locations along the laser beam. Also, it can only measure the line-of-sight (LoS) wind speed in the laser beam direction [84], so the wind speed magnitude and direction have to be estimated (Cyclops' dilemma). The measurement of the whole spatiotemporal wind field is still not feasible by the current sensor technologies.

On the other hand, numerous wind prediction approaches have been proposed and recent advancements include the deep learning ensemble model with data denoising [85], the recurrent neural networks based approach with error correction [86], and the variational Bayesian deep learning based approach [87]. These studies showed very promising results in wind speed predictions. However, as the measurement data is usually only available at sparse spatial locations, the whole flow field in front of the wind turbines can not be predicted with these approaches. Numerical approaches have been investigated to obtain the detailed spatiotemporal wind field information [8, 88], by numerically solving the NS equations. However, numerical models were mainly designed for forward simulations of fluid flows thus cannot take real-time wind measurements into account. In order to achieve the accurate prediction of the spatiotemporal wind field information, both flow physics and real-time wind measurements need to be considered.

Currently there are very limited studies that can achieve wind field predictions based on scattered wind measurements. In [89], a wind field reconstruction method was proposed, where a simplified dynamic model of the atmospheric boundary layer was derived and then an unscented Kalman filter (UKF) was used to estimate the model state from LIDAR measurements. The sensitivity study of the developed method was also carried out, where different beam half-angles, look directions, atmosphere conditions, and measurement noise levels were considered. In [90], a velocity and pressure field estimation framework was proposed, where a reduced order dynamic model was built based on NS equations, by uniquely employing a pressure Poisson equation formulation in conjunction with a basis function decomposition method. Then a modified UKF algorithm was used for the state estimation. The proposed method was validated by both numerical experiments and real-world LIDAR measurements. In [91], a wind field reconstruction method was developed based on CFD and proper orthogonal decomposition (POD), where CFD simulations were carried out to generate a database of wind fields and POD was employed to extract the low-dimensional basis vectors. The flow field reconstruction was then carried out through these basis vectors based on measurements from the optimally-placed sensors. The authors of [91] further investigated the optimal design of the sensor arrangement in order to improve the reconstruction performance [92]. To summarize, all these studies showed very promising results by combining flow physics (either through dynamic wind model, NS equations or CFD simulations) with wind measurements in the wind field prediction process. However, due to the complexity (e.g. the strong nonlinearity and the multi-scale characteristics) of the wind dynamics, the aforementioned studies all included an explicit model reduction process in reconstructing the flow field (i.e. low-order wind model [89], reduced-order model from NS equations [90], dimensionality reduction by POD [91, 92]), therefore their prediction performance was limited by the explicit model reduction error (i.e. the error due to the limited number of reduced basis used for the flow reconstruction).

1.2 Machine Learning Based Modelling and Control Approaches

This section reviews the ML approaches that are closely related to the modelling and control works presented in this thesis. They include RL, supervised ML, dimensionality reduction, generative adversarial network (GAN) and physics-informed deep learning. It is worth noting that this is a very broad research field and the literature review in this section is not exhaustive i.e. only the most relevant literature is included.

1.2.1 Reinforcement Learning for Optimal Control

RL, also called approximate or adaptive dynamic programming (ADP) in control communities, is a powerful tool for optimal control of complex systems. Originally proposed by Webos [93, 94], ADP has caught extensive attention recently on the optimal control of both continuous-time and discrete-time systems [95–102]. It is specifically designed and developed to tackle the control of complex nonlinear systems. Typical examples include coal gasification [103], energy management systems [104], hypersonic vehicle tracking [105], microgrid system [106] and optimal tracking [107].

There are mainly two types of iterative ADP algorithms: Value Iteration Approximate Dynamic Programming (VI-ADP) and Policy Iteration Approximate Dynamic Programming (PI-ADP) [108–110]. For VI-ADP approaches, the iteration begins with an initial value function, and then the policy improvement is carried out according to the iterative value function. This approach does not require an initial admissible control law. However, the initialization of the value function needs to be designed in order to guarantee the stability and convergence of the iteration process. The paper [111] proposed a VI-ADP approach with a value function initialization technique, where the convergence property was also proved. In [112] a generalized VI-ADP algorithm was proposed, which only requires an arbitrary positive semi-definite function to initialize the value function in order to guarantee the convergence property of the algorithm. On the other hand, for the PI-ADP approaches, an admissible initial control law is required for the iteration process. In [113], a discrete-time PI-ADP algorithm was proposed for nonlinear systems with convergence and stability analysis, and an effective method to obtain the initial admissible control law was given. A generalized PI-ADP approach was proposed in [114], which relaxed the requirement of obtaining the initial admissible policy [99]. Though very powerful in tackling complex nonlinear control problems, the work on the RL-based control of wind energy systems is still in its infancy and more works are expected to emerge in the future.

1.2.2 Machine Learning for Surrogate Modelling of Fluid Flows

CFD is an important tool for investigating complex flow problems in many engineering and scientific applications, e.g. aircraft design, weather forecasting, and turbulence research. However, despite the fast development of HPC technology, its use for repetitive and real-time tasks, such as optimization and real-time control, is still highly challenging due to the requirement of enormous computational resources and long simulation time. For instance, the 3D Reynold-averaged Navier-Stokes (RANS) simulation of complex flows typically requires several hundred/thousand CPU hours, not to say the LES and the direct numerical simulations (DNS) where the unsteady flow details are resolved. Therefore, surrogate modelling, which aims at constructing an efficient yet accurate approximation to the full CFD model, has attracted a lot of attention, such as in aeroelastic computations [115], aerodynamic load evaluations [116], aerodynamic simulations with consideration of multiple operating conditions [117], UQ of turbulence models [73], combustion modelling [118], and the simulations of single-injector combustion process [119].

ML-based surrogate modelling of fluid flows is attracting more and more research attention recently. One approach is to directly formulate the surrogate modelling of fluid systems as a supervised ML problem to train a model with the flow parameters as training input and the full flow field as training output [120,121]. This kind of approach makes use of the-state-of-art ML algorithms, which can thus mimic the fluid system to high accuracy if there are enough training data available and the model generalisation issue is carefully addressed. Another approach is to first reduce the high-dimensional flow field into its low-dimension representation by a dimensionality reduction technique, and then formulate a supervised ML problem to predict the reduced coefficients instead of the full flow field, with the flow parameters as input. In this way, the trained model can capture the main features of the fluid systems while alleviating the need for a large amount of high-fidelity training data.

The combination of POD for dimensionality reduction [122–124] and supervised ML for flow field prediction has been investigated recently. In [125], a surrogate modelling technique called POD-NN was proposed, where POD was used to extract the reduced basis and neural network (NN) was used for approximating the map between flow parameters and the POD coefficients. The combination of POD and other ML techniques, i.e. Gaussian process regression (GPR), was investigated in [126,127] and it was tested on a set of numerical examples. The potential to incorporate physical constraints was investigated in [128]. They proposed the use of particular solutions in the POD expansion to enforce certain constraints, e.g. boundary conditions. In addition, a set of supervised ML techniques were considered, including NN, multivariate polynomial regression, k-nearest-neighbours, and decision trees.

To avoid the explicit dimensionality reduction errors and at the same time mitigate the overfitting issues in predicting high-dimensional target, GAN, an ML framework extremely powerful in generating realistic high-dimensional content, may offer improved performance for the surrogate modelling of fluid flows. Different from explicitly fitting the training target in the supervised ML, GAN is trained implicitly to produce realistic high-dimensional content. It consists of two networks, a generator and a discriminator. The generator is designed to generate high-dimensional content while the discriminator aims to distinguish the content generated by the generator from the true content. The generator and the discriminator are trained in an adversarial way such that the generator learns to produce more realistic content so that it can 'fool' the discriminator, while the discriminator learns better ways to distinguish fake from true. Since its first publication, GAN has attracted extensive attention rapidly in the ML field. Its developments have led a lot of exciting successes, such as the conditional generative adversarial network (CGAN) [129], deep convolutional generative adversarial network (DC-GAN) in image representations [130], image-to-image translation with CGAN [131], Wasserstein GAN [132], and cycle-consistent GAN [133]. The applications of GAN in fluid problems are still rare. A few examples include the super-resolution problems of fluid flows [134], super-resolution turbulent flow reconstructions [135], modelling of fluid flows using DC-GAN [136], and turbulence enrichment [137]. It is expected that more works based on GAN will emerge in the future on various fluid applications, as they share an important feature - the high dimensionality of the model output (e.g. high-resolution images vs high-fidelity flow snapshots).

1.2.3 Machine Learning Incorporating Data and Physics

Most works on the application of ML in physical systems treat ML models as 'black-box' and the physical knowledge is usually neglected during the training of the ML model. In order to take full advantage of both data and the physical knowledge, the works on the incorporation of physical laws in the training of deep learning models are emerging lately, such as the data-based turbulence modelling [138,139], the discovery of governing equations [140,141], solving high-dimension partial differential equations (PDEs) [142] and the surrogate modelling of physical systems [143,144]. A versatile ML framework for solving forward and inverse problems involving PDEs, called physics-informed neural networks (PINNs), was proposed in [145] recently. The main idea of PINNs is to encode PDEs in terms of loss functions, which are then used for NN training together with the available labelled data. Specifically, automatic differentiation [146] is employed to take the derivatives of the NN output with respect to the NN input (i.e. space and time coordinates). These derivatives are then used to form the loss functions that represent the residues of the PDEs. The development of PINNs is becoming very active. Recent studies include both method development (such as the UQ of PINNs [147], the use of adaptive activation functions [148] and the learning from multi-fidelity [149] and noisy data [150]) and various applications (such as vortex-induced vibrations [151], high-speed flow [152], and hidden-physics inference from flow visualizations [153]).

For the ML-based surrogate modelling of fluid flows, a set of flow field data, which is usually generated by CFD simulations, is available for training the ML model. However, there are other research scenarios that the flow field data is not available at all. For example, in the spatiotemporal wind field prediction problems considered in this thesis, only sparse LIDAR measurements are available. In such scenarios, the physics-informed deep learning is particularly powerful as it does not need the flow field data to train the deep learning model. Therefore, by incorporating physics and data in a unified framework, it opens new opportunities in the scenarios of ‘small’ data for physical systems.

1.3 Motivations and Research Contributions

With the main technological challenges identified and the related literature described in the previous sections, the motivations and the research contributions of this thesis are described in this section.

- At the turbine level, the structural control of floating wind turbines using active TMD is investigated based on the RL approach. Different from previous works which are based on linear design models, the proposed approach takes account of the nonlinear dynamics of the structural system in the control design process. To the best of the author’s knowledge, this is the first time that RL-based approach is employed to this type of application. Specifically, an ADP algorithm is used to derive the optimal control law based on the nonlinear structural dynamics, and the large-scale ML platform Tensorflow is employed for the design and implementation of the NN structure.

- Then the research moves to the farm level. For the first time, the parameter uncertainty of wind farm wake models is investigated. As previous works in the literature mainly focus on the input uncertainty, this work further advances the accurate characterization of uncertainties in wind farm wake predictions. Specifically, the parameter uncertainty of the widely-used FLORIS model is quantified rigorously in the Bayesian UQ framework, based on the high-fidelity flow field data generated by the LES flow solver Simulator fOr Wind Farm Applications (SOWFA).

- After investigating traditional analytical wake models, novel wake models are then developed. An ML-based surrogate modelling method for distributed fluid systems is first proposed, where a dimensionality reduction technique, such as POD, independent component analysis (ICA), auto-encoder (AE) in this work, is employed to reduce the flow field dimension and a regression model (such as the fully-connected NN in this work) is employed to predict the reduced representation of the flow field with the flow parameters as the input. The proposed method is specifically designed to tackle the fluid systems involving distributed aerodynamic structures, by first decomposing the whole fluid domain into subdomains, then carrying out surrogate modelling for each subdomain by treating both the boundary information and the distributed flow parameters as the input parameters, and finally combining the flow field of each subdomain with the consideration of the matching condition at the subdomain interface. The proposed method is then applied to wind farm wake modelling, based on the flow field data generated by high-fidelity simulations.

- The above surrogate modelling method is based on the supervised ML framework and explicit dimensionality reductions are employed to reduce the flow field dimension. To avoid the explicit dimensionality reduction errors and at the same time mitigate the overfitting issues in predicting high-dimensional target, a novel surrogate modelling method is proposed, which follows the GAN framework [154], in particular deep convolutional conditional generative adversarial network (DC-CGAN). A novel wind farm wake model is then developed using the proposed method, based on the data generated by high-fidelity simulations.

- The above works focus on the static wind farm wake modelling. Thus they can not be used for the simulation of unsteady wind farm wakes. In order to capture the dynamic wake behaviours, a deep learning based dynamic wake model is developed. First, a set of LES simulations are carried out to generate a series of flow field data for wind turbines operating in different conditions. Then a surrogate modelling method, called POD-LSTM, is proposed to build this novel wake model, where POD is employed to reduce the flow field dimension and the long short-term memory (LSTM) network is employed to predict the reduced representation of the flow field at a future time step based on historical flow fields. After training, the model can be used for the fast simulation of unsteady wind farm wakes. An illustrative example is given based on a 9-turbine wind farm to demonstrate the capability of the method and evaluate the computational cost.

- The load mitigation of individual wind turbines and the modelling & control of wind farms both depend greatly on the accurate characterization of wind field information, which is still not available with the current sensor technology. To fill the gap, a deep learning method which can combine sparse LIDAR measurements and flow physics is developed to predict the time-varying onset flow upstream of a wind turbine. Specifically, a deep NN is constructed and the NS equations are incorporated in the deep NN by employing the physics-informed deep learning technique. The training of this physics-incorporated deep learning model only requires the sparse LIDAR measurement data while the spatiotemporal wind field in the whole domain (which cannot be measured) can be predicted after training. This study can discover complex wind patterns that do not present in the training dataset, thus is totally distinct from previous ML-based wind prediction studies which treat ML models as 'black-box' and require the corresponding input and target values to learn complex relations.

- All the previous works focused on the predictions of the 2D wind field. For the first time, the prediction of the full 3D spatiotemporal wind field in front of a wind turbine is investigated in this thesis. Specifically, a deep learning model is developed which combines the 3D NS equations and the scanning LIDAR measurements via physics-informed deep learning. The turbulent viscosity is also taken into account to further improve the prediction performance.

1.4 Thesis Outline

In Chapter 2, the RL-based structural control of floating wind turbines is presented. This chapter is derived based on the following peer-reviewed journal paper published by the author:

- **J. Zhang**, X. Zhao and X. Wei, Reinforcement learning-based structural control of floating wind turbines, **IEEE Transactions on Systems, Man, and Cybernetics: Systems**(2020), DOI: 10.1109/TSMC.2020.3032622.

In Chapter 3, the quantification of parameter uncertainty in wind farm wake modelling is investigated. This chapter is derived based on the following peer-reviewed journal paper published by the author:

- **J. Zhang** and X. Zhao, Quantification of parameter uncertainty in wind farm wake modeling, **Energy** 196 (2020) 117065.

In Chapter 4, an ML-based surrogate modelling method is proposed for aerodynamic flow around distributed structures, which is then applied to wind farm wake modelling. This chapter is derived based on the following peer-reviewed journal paper published by the author:

- **J. Zhang** and X. Zhao, Machine-learning-based surrogate modeling of aerodynamic flow around distributed structures, **AIAA Journal** 59 (3) (2021) 868–879.

In Chapter 5, a surrogate modelling method is proposed based on DC-CGAN and then applied to wind farm wake modelling. This chapter is derived based on the following journal paper draft:

- **J. Zhang** and X. Zhao, Wind farm wake modeling based on deep convolutional conditional generative adversarial network, journal paper draft (2021), under review.

In Chapter 6, a novel dynamic wind farm wake model is developed which can capture dynamic wake characteristics in real-time. This chapter is derived based on the following peer-reviewed journal paper published by the author:

- **J. Zhang** and X. Zhao, A novel dynamic wind farm wake model based on deep learning, **Applied Energy**, 277 (2020) 115552.

In Chapter 7, the spatiotemporal wind field prediction based on physics-informed deep learning and LIDAR measurements is presented. This chapter is derived based on the following peer-reviewed journal paper published by the author:

- **J. Zhang** and X. Zhao, Spatiotemporal wind field prediction based on physics-informed deep learning and LIDAR measurements, **Applied Energy** 288 (2021) 116641.

In Chapter 8, the full 3D spatiotemporal wind field reconstruction is investigated based on the 3D NS equation and scanning LIDAR measurements. This chapter is derived based on the following peer-reviewed journal paper published by the author:

- **J. Zhang** and X. Zhao, Three-dimensional spatiotemporal wind field reconstruction based on physics-informed deep learning, **Applied Energy** 300 (2021) 117390.

In Chapter 9, the conclusions are drawn and future perspectives are discussed.

I declare that all the works in the above papers are originated, conducted, and investigated by me, including conceptualization, data curation, formal analysis, investigation, methodology, project administration, software, validation, visualization, and writing of the original draft & review.

Chapter 2

Reinforcement Learning Based Structural Control of Floating Wind Turbines

2.1 Introduction

This chapter investigates the RL-based control design for the load mitigation of floating wind turbines, by using active TMD. The barge type platform is considered in this work, and the TMD is installed on the platform instead of in the turbine nacelle. Then the PI-ADP approach for discrete-time systems [99] is employed for the structural control design. Because the considered open-loop system (the floating wind turbine with passive TMD) is stable, there exists a natural admissible control policy (i.e. active control force set to be zero). This control policy is used for the initialisation of the employed PI-ADP approach which can ensure the stability of the closed-loop system [99, 113]. The employed algorithm includes three networks, i.e. an action network, a critic network, and a plant network. The training of the plant network and critic network is carried out in supervised manner while the training of the action network aims to minimise the critic network output, which requires the gradient information flowing through all the three networks. The NN structures in the existing literature are usually very simple where only weight vectors outside the NN activations were included in the NN structure or the weight matrix in the hidden layer were included but not used for training, which undermined the ability of NNs in approximating the plant's complex nonlinear behaviour in practical applications. This allows the gradients being derived analytically, but it is infeasible to do so for complex network structures. In this work, the automatic differentiation is employed to calculate the gradients in all the NN training including the training of the hidden layer and this method is independent of specific applications, which greatly extends the PI-ADP algorithm's ability in solving complex practical problems and simplifies its implementation. The large-scale ML platform Tensorflow [155] is used for the implementation of the proposed NN structure. Its highly parallel computing environment makes the proposed NN implementation even more powerful, especially with the use of GPU.

The National Renewable Energy Laboratory (NREL) 5-MW baseline ITI Energy barge wind turbine model [3] (upwind, three blades, 126m rotor, variable speed) is used in this study. An HMD (i.e. a TMD with an additional active force control) is installed on the platform and designed to suppress the vibration in the fore-aft direction. It is worth noting that the side-to-side vibration of the platform can be well tackled by passive TMDs. The NREL Flow Analysis Software Toolkit (FAST) code [156] is employed to simulate the structural system, and the plant network is trained based on the data generated by FAST. After training the plant network, a series of ADP controllers are obtained by varying the penalty term in the action-critic network training, which considers the trade-off between the control performance and power consumption.

The remaining part of this chapter is organised as follows: the structural control of floating wind turbines is formulated in Section 2.2. The PI-ADP algorithm and its implementation with the proposed NN structure are described in Section 2.3, where the training of the plant network, the critic network, and the action network is presented in detail. The structural control design based on PI-ADP is described in Section 2.4. The control performance is evaluated in Section 2.5, where a set of wind/wave conditions are considered. Finally the conclusions are drawn in Section 2.6.

2.2 Problem Formulation

The structural control of a floating wind turbine is described in this section. Here the turbine's structural dynamics with HMD and the control objective are given.

2.2.1 Floating Turbine System with HMD

The structural control of an NREL 5-MW floating wind turbine model within FAST code [3] is investigated here. An HMD is coupled to the floating platform of this model, which moves in the fore-aft direction to suppress the structural vibration of the floating turbine in this direction, see Figure 2.1. An illustration of the floating

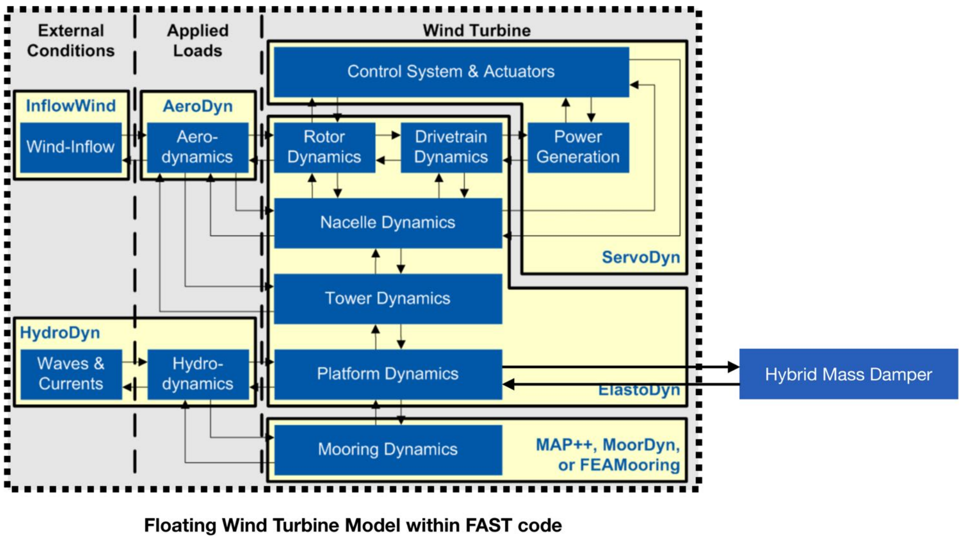


Figure 2.1: A schematic illustration of the floating wind turbine model within FAST [157] coupled with an HMD.

turbine structural system is also given in Figure 1.1(a). A stroke limit of ±17m is imposed for the HMD since the length of the platform is 40m. The damping coefficient, stiffness coefficient and mass of the HMD are set as 60393N/(m/s), 103019N/m and 400000kg respectively, which are the optimal values used in [9]. All the DoFs are enabled except the nacelle yaw DoF as the yaw control is not considered in this work. Among all the DoFs, the main structural dynamics of this turbine-HMD system can be characterized by the platform pitch angle, the tower-top displacement, and the HMD displacement. Therefore, the structural system can be approximated by a discrete system F:

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k, u_k), \;\; k = 0, 1, 2, ..., \tag{2.1}$$

where k is the discret time step, u is the control variable (i.e. the active HMD force), and $\mathbf{x}$ is the state variable which is defined as

$$\mathbf{x} = [x_{hmd}, u_{hmd}, x_{plfm}, u_{plfm}, x_{tt}, u_{tt}]. \tag{2.2}$$

Here x_{hmd}, u_{hmd}, x_{plfm}, u_{plfm}, x_{tt}, u_{tt} represent the HMD displacement, HMD velocity, platform pitch angle, platform pitch angular velocity, tower top displacement and tower-top velocity, respectively.

2.2.2 Control Objective

The active structural control aims at reducing the vibrations of the turbine's platform and tower in the fore-aft direction with a minimum amount of power consumption. Denote the sequence of active HMD forces as $\bar{u}_k = \{u_k, u_{k+1}, u_{k+2}, ...\}$, then the cost function for the state $\mathbf{x}_0$ under the control $\bar{u}_0$ is defined as

$$J(\mathbf{x}_0, \bar{u}_0) = \sum_{k=0}^{\infty} U(\mathbf{x}_k, u_k) \tag{2.3}$$

where the utility function $U(\mathbf{x}_k, u_k)$ is defined as

$$U(\mathbf{x}_k, u_k) = (\mathbf{x}_k)^T \cdot A_u \cdot (\mathbf{x}_k) + B_u(u_k)^2, \tag{2.4}$$

where $\cdot$ represents the dot product, the superscript T represents the matrix transpose, and the empirical parameters A_u and B_u are used to investigate the trade-off between the active control force and the control performance. Equation 2.4 allows the utility function $U(\mathbf{x}_k, u_k)$ to be positive definite [99, 113] and to take account of the costs from both the structural vibrations (which is described by the first term) and the active power consumption (which is described by the second term).

Here this work focuses on state-feedback control thus an arbitrary control law can be expressed as

$$u_k = \mu(\mathbf{x}_k). \tag{2.5}$$

The cost function for the state $\mathbf{x}_0$ under the control law μ can then be expressed as

$$J^{\mu}(\mathbf{x}_0) = \sum_{k=0}^{\infty} U(\mathbf{x}_k, \mu(\mathbf{x}_k)). \tag{2.6}$$

The structural control objective is to find an optimal control policy $\mu^*(\mathbf{x}_k)$ such that

$$\mu^*(\mathbf{x}_k) = \arg\min_{u_k}\{U(\mathbf{x}_k, u_k) + J^*(F(\mathbf{x}_k, u_k))\}, \tag{2.7}$$

where

$$J^*(\mathbf{x}_k) = \min_{u_k}\{U(\mathbf{x}_k, u_k) + J^*(F(\mathbf{x}_k, u_k))\}. \tag{2.8}$$

2.3 PI-ADP Algorithm and its Implementation

The PI-ADP algorithm begins with an admissible control law μ_0 and then obtains the optimal cost function and the optimal control law iteratively through the policy evaluation and policy improvement procedure. During policy evaluation, the value function V_i is constructed based on the corresponding control law μ_i such that it satisfies the following equation

$$V_i(\mathbf{x}_k) = U(\mathbf{x}_k, \mu_i(\mathbf{x}_k)) + V_i(F(\mathbf{x}_k, \mu_i(\mathbf{x}_k))). \tag{2.9}$$

Then during the policy improvement, the control law μ_{i+1} is updated based on the value function V_i according to

$$\mu_{i+1}(\mathbf{x}_k) = \arg\min_{u_k}\{U(\mathbf{x}_k, u_k) + V_i(F(\mathbf{x}_k, u_k))\}. \tag{2.10}$$

Through the iteration process ($\mu_0 \rightarrow V_0 \rightarrow \mu_1 \rightarrow V_1 \rightarrow \mu_2 \rightarrow ...V_{N-1} \rightarrow \mu_N$), the optimal cost function J^* is approximated by V_N and the optimal control law μ^* is approximated by μ_N. The properties of the PI-ADP algorithm have been proved in [99, 113], where an admissible initial control law is required to guarantee the convergence and stability of the algorithm. The main contribution of this chapter is a novel NN realisation of the employed PI-ADP algorithm and its application on structural control of floating wind turbine (which is also applicable to other complex industrial systems). The interested reader may refer to [99, 113] for the detailed proof of stability. The proposed NN structure and its training details are described in the following subsections.

2.3.1 Neural Network Structure

The whole NN structure proposed in this work is illustrated in Figure 2.2. The plant network, the action network, and the critic network in Figure 2.2 are all fully-connected NNs with one-hidden layer as illustrated in Figure 2.3.

Here the network is designed in order to feed standardized data for all the NN trainings. The standard scaler (denoted as Scaler 1, Scaler 2, Scaler 3 and Scaler 4 in Figure 2.2) is employed, which normalizes the data by their mean value and standard deviation

$$\mathbf{d}^{std} = \frac{\mathbf{d} - m(\mathbf{d})}{s(\mathbf{d})} \tag{2.11}$$

where $m(\mathbf{d})$ and $s(\mathbf{d})$ represent the mean and standard deviation of the dataset

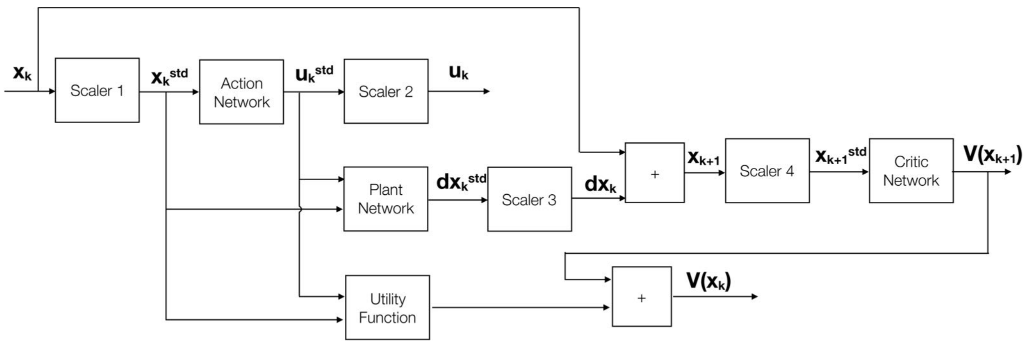


Figure 2.2: The whole NN structure including the plant network, the action network and the critic network.

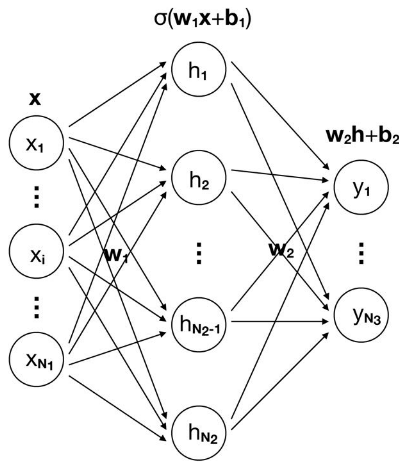


Figure 2.3: The illustration of a fully-connected NN with one hidden layer. $\mathbf{x}$ is the N_1-dimension input variable, $\mathbf{h}$ is the N_2-dimension hidden layer output, and $\mathbf{y}$ is the N_3-dimension output variable. $\mathbf{w}_1$, $\mathbf{b}_1$, $\mathbf{w}_2$, and $\mathbf{b}_2$ are the training variables of the NN and σ is the activation function.

d. The utility function is redefined in terms of the standardized state and action variables as

$$U(\mathbf{x}_k^{std}, u_k^{std}) = (\mathbf{x}_k^{std})^T \cdot A_u^* \cdot (\mathbf{x}_k^{std}) + B_u^*(u_k^{std})^2, \tag{2.12}$$

so that the costs arising from structural vibration and the power consumption are comparable. The forward and inverse mappings of Scaler 1 - Scaler 4 are denoted as $\mathscr{S}_1$ - $\mathscr{S}_4$ and $\mathscr{S}_1^{-1}$ - $\mathscr{S}_4^{-1}$, respectively. The forward mappings of the plant network, the action network, and the critic network are denoted as $\mathscr{P}$, $\mathscr{A}$ and $\mathscr{C}$ respectively.

The plant network is designed to approximate the structural system such that

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathscr{S}_3^{-1} \circ \mathscr{P}\big(\mathscr{S}_1(\mathbf{x}_k), \mathscr{S}_2(u_k)\big), \tag{2.13}$$

where $\circ$ represents the composition of functions. The plant network is designed to predict the state change instead of the state variable because this way can greatly increase the prediction accuracy — the state change is usually subtle compared to the state variable. The action network is designed to approximate the structural controller such that

$$u_k = \mathscr{S}_2^{-1} \circ \mathscr{A} \circ \mathscr{S}_1(\mathbf{x}_k). \tag{2.14}$$

The critic network is designed to approximate the value function such that

$$V(\mathbf{x}_k) = \mathscr{C} \circ \mathscr{S}_4(\mathbf{x}_k). \tag{2.15}$$

Thus, with this NN structure, Equation 2.9 is approximated by

$$\begin{aligned} \mathscr{C}_i \circ \mathscr{S}_4(\mathbf{x}_k) = {} & U\big(\mathscr{S}_1(\mathbf{x}_k), \mathscr{A}_i \circ \mathscr{S}_1(\mathbf{x}_k)\big) \\ & + \mathscr{C}_i \circ \mathscr{S}_4\Big(\mathbf{x}_k + \mathscr{S}_3^{-1} \circ \mathscr{P}\big(\mathscr{S}_1(\mathbf{x}_k), \mathscr{A}_i \circ \mathscr{S}_1(\mathbf{x}_k)\big)\Big), \end{aligned} \tag{2.16}$$

and Equation 2.10 is approximated by

$$\begin{aligned} \mathscr{A}_{i+1} \circ \mathscr{S}_1(\mathbf{x}_k) = {} & \underset{\mathscr{A} \circ \mathscr{S}_1(\mathbf{x}_k)}{\arg\min} \Big\{ U\big(\mathscr{S}_1(\mathbf{x}_k), \mathscr{A} \circ \mathscr{S}_1(\mathbf{x}_k)\big) \\ & + \mathscr{C}_i \circ \mathscr{S}_4\Big(\mathbf{x}_k + \mathscr{S}_3^{-1} \circ \mathscr{P}\big(\mathscr{S}_1(\mathbf{x}_k), \mathscr{A} \circ \mathscr{S}_1(\mathbf{x}_k)\big)\Big)\Big\}. \end{aligned} \tag{2.17}$$

2.3.2 Neural Network Training

The training of the proposed NN structure is detailed here and the overall training process is summarised in **Algorithm 1**.

Equation 2.13 can be reformulated as

$$\mathscr{S}_3(\mathbf{x}_{k+1} - \mathbf{x}_k) = \mathscr{P}\big(\mathscr{S}_1(\mathbf{x}_k), \mathscr{S}_2(u_k)\big), \tag{2.18}$$

thus the plant network is trained by minimising the mean-squared error (MSE) loss

$$l_p = MSE\Big(\mathscr{S}_3(\mathbf{x}_{k+1} - \mathbf{x}_k), \mathscr{P}\big(\mathscr{S}_1(\mathbf{x}_k), \mathscr{S}_2(u_k)\big)\Big). \tag{2.19}$$

The critic network training can be done in two different approaches. In the

Algorithm 1 The training of the proposed NN structure

```
1: Load the training data $\{x_k^*\}$, $\{u_k^*\}$ and $\{x_{k+1}^*\}$, and preprocess them to obtain $\{x_k^{std*}\}$, $\{u_k^{std*}\}$ and $\{dx_k^{std*}\}$.
2: Set the hidden layer neuron number $N_2$; Set the learning rate $lr$.
3: Train the plant network by feeding $\{x_k^{std*}\}$, $\{u_k^{std*}\}$ and $\{dx_k^{std*}\}$ into $\mathbf{x}_k^{std}$, $\mathbf{u}_k^{std}$, and $\mathbf{dx}_k^{std}$.
4: Set the maximum iteration number $N_{iter}$.
5: Initialise $i = 1$; Set the convergence criterion $\epsilon$.
6: Initialise the action network such that the initial control law is admissible.
7: while $i < N_{iter} + 1$ do
8:     Compute the $\{V_i(x_{k+1}^*)\}$ according to Equation 2.20 by feeding $\{x_{k+1}^*\}$ into $\mathbf{x}_k$.
9:     Train the critic network by feeding $\{x_{k+1}^*\}$ and $\{V_i(x_{k+1}^*)\}$ into $\mathbf{x}_{k+1}$ and $V(\mathbf{x}_{k+1})$.
10:    Train the action network by feeding $\{x_k^*\}$ into $\mathbf{x}_k$.
11:    if $i > 1$ then:
12:        Compute $e_{conv}$: the MSE between $\{V_{i-1}(\mathbf{x}_k)\}$ and $\{V_i(\mathbf{x}_k)\}$.
13:        if $e_{conv} < \epsilon$ then:
14:            The whole process is convergent.
15:            Break.
16:        end if
17:    end if
18:    $i \leftarrow i + 1$
19: end while
```

first approach, the critic network is trained such that Equation 2.9 is satisfied. In the second approach, the value function is approximated directly by

$$V(\mathbf{x}_k) = \sum_{j=0}^{N} U(\mathbf{x}_{k+j}^{std}, u_{k+j}^{std}) \tag{2.20}$$

where N is a large number. In this work the second approach is employed, as it converges faster than the first one. Thus the critic network is trained by minimising

$$l_c = MSE\Big(\sum_{j=0}^{N} U\big(\mathscr{S}_1(\mathbf{x}_{k+j}), \mathscr{A} \circ \mathscr{S}_1(\mathbf{x}_{k+j})\big), \mathscr{C} \circ \mathscr{S}_4(\mathbf{x}_k)\Big). \tag{2.21}$$

where $\mathbf{x}_{k+j}$ is obtained by evaluating the plant network iteratively.

According to Equation 2.17, the action network is trained by minimising

$$
\begin{aligned}
l_a = MSE\Bigg(&0, U\big(\mathscr{S}_1(\mathbf{x}_k), \mathscr{A} \circ \mathscr{S}_1(\mathbf{x}_k)\big) + \\
&\mathscr{C} \circ \mathscr{S}_4\Big(\mathbf{x}_k + \mathscr{S}_3^{-1} \circ \mathscr{P}\big(\mathscr{S}_1(\mathbf{x}_k), \mathscr{A} \circ \mathscr{S}_1(\mathbf{x}_k)\big)\Big)\Bigg).
\end{aligned} \tag{2.22}
$$

The critic network and the action network are trained iteratively until the whole process converges which is described in detail in Algorithm 1. All the NN trainings are carried out with Adam optimization algorithm [158]. Automatic differentiation [146], which calculates the gradients of complex graphs automatically based on the chain rule, is employed here for deriving $\partial l_p/\partial w_p$ ($\partial l_c/\partial w_c$ or $\partial l_a/\partial w_a$) for the plant (critic or action) network training, where w_p (w_c or w_a) represents the training variables in the plant (critic or action) network.

The relationship between the three sub-networks is further explained here. Plant network is trained alone in supervised manner, as only $\mathscr{P}$ is involved in Equation 2.19. Critic network is also trained alone in supervised manner as only forward evaluation of $\mathscr{A}$ and $\mathscr{P}$ is needed in deriving $\partial l_c/\partial w_c$ in Equation 2.21. The training of action network, however, requires the gradient information flowing through the whole network, as $\mathscr{P}$, $\mathscr{C}$ and $\mathscr{A}$ are all involved in deriving $\partial l_a/\partial w_a$ in Equation 2.22.

2.4 HMD Controller Design of a Floating Wind Turbine

In this section, the design of an ML-based HMD controller for a floating wind turbine is investigated, using the NN structure and training algorithm proposed in the previous section.

2.4.1 Plant Network Training

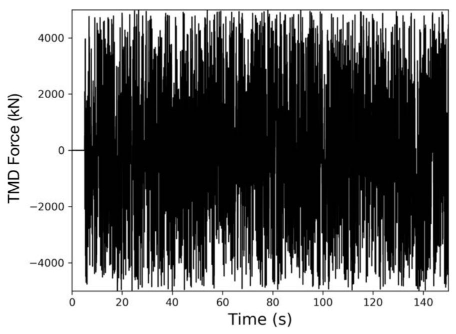
The plant network is trained by using the data generated by the floating wind turbine simulation model shown in Figure 2.1. The training dataset is a set of training samples with each sample consisting of the state variable at the current time step, the action variable at the current time step and the state variable at the next time step. The time step is set as 0.06s, which is also the time step of the trained plant network. To generate the training dataset, 100 random initial conditions for $[x_{hmd}, x_{plfm}, x_{tt}]$ in the parameter space $[-15, 15]$m $\times$ $[-15, 15]$deg $\times$ $[-3, 3]$m are first obtained by Latin hypercube sampling method, then a 15-second simulation with a time step of 0.01s is carried out for each initial condition, under the excitation of random HMD force with a time step of 0.1s and within the interval $[-5000, 5000]$kN. Next, the first 5-second time series is eliminated and the data sample is extracted with a time interval of 0.1s. All the data samples are collected together to form the final training dataset.

After generating the training dataset, a plant network with the hidden-layer neuron number $N_2 = 20$ is constructed. The learning rate is set as 0.001 and the training error is set as 10^{-3}. The plant network is then trained to mimic the structural system. In order to assess the accuracy of the trained network, a comparison of the FAST simulation results and the plant network calculations is given in Figure 2.4, with the test HMD force time series shown in Figure 2.4(a) and the comparison of the structural response under this HMD force excitation shown in 2.4(b)-(g). Both calculations are based on the same initial condition at $t = 5$s, and the test HMD force time series shown in Figure 2.4(a) has not been used during training. Figures 2.4(b)-(g) show a perfect match between the FAST simulation and the plant network calculation for the whole simulation period. This demonstrates that the plant network has captured the nonlinear dynamics of the structural system.

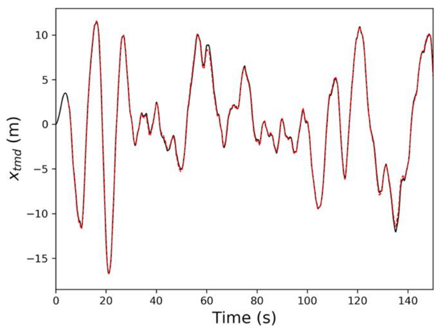
2.4.2 Action-Critic Network Training

The action-critic network training is conducted iteratively according to Algorithm 1. The hidden-layer neuron number of both networks is set as 20, and the learning rate is set as 0.001. The bias term is not used for the action network, imposing the condition $\mu(0) = 0$. The training error of critic network is set as 10^{-3} and the training of action network is deemed completed when the training loss l_a drops less than a prescribed threshold with further training, which is set as 10^{-5} here. The action network is initialised by very small random weights as $\mu_0 = 0$ is an admissible control law for the structural system. N in Equation 2.20 is set as 5000. A_u^* in the utility function is set as $10^{-4} \times diag\{(1, 1, 25, 25, 1, 1)\}$ and a number of values are chosen for B_u^*, which are reported in Table 2.1. Each chosen B_u^* results in a different action network after the training process. These trained action networks are the state-feedback controllers that will be used for the structural control in the next section (denoted as ADP1-ADP8 hereafter).

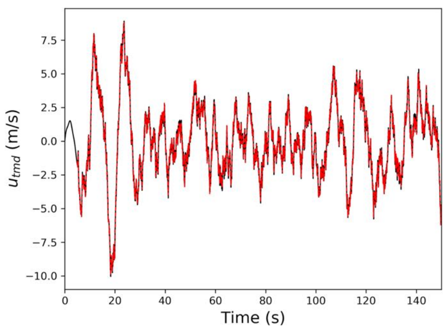
All the NN trainings are carried out with one NVIDIA Tesla K80 GPU card to take advantage of Tensorflow's efficiency with GPU backend. For all the 8 (ADP1-ADP8) controller design, on average, the plant network training requires about 88s, the critic network training requires about 33s, and the action network training requires about 245s. The action network training takes much longer than other NNs



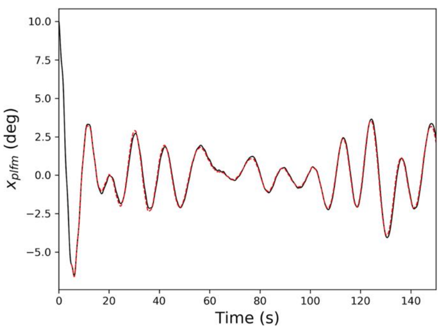
(a) The test HMD force



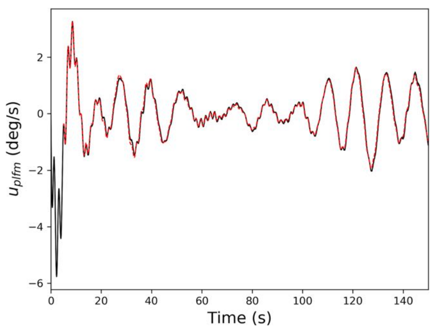
(b) HMD displacement



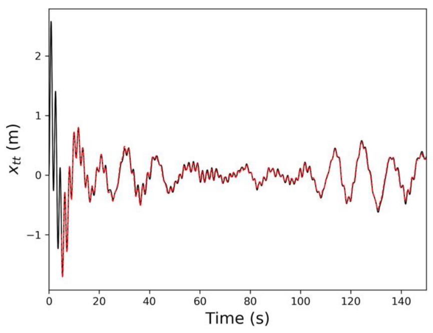
(c) HMD velocity



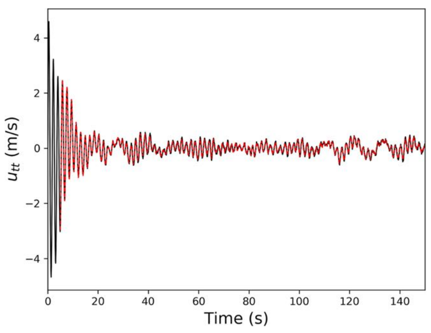
(d) Platform pitch angle



(e) Platform pitch angular velocity



(f) Tower top displacement



(g) Tower top velocity

Figure 2.4: The comparison of the FAST simulation results (solid line) and the plant network calculations (dashed line).

as it requires the gradient information flowing through the whole network (all three sub-networks are involved). The same training process is also tested out with 4 INTEL Xeon CPUs (2.40GHz) and it is more than three times slower than training with GPU. This clearly demonstrates the advantage of the current implementation on Tensorflow with GPU backend.

2.5 Simulation Study

With the converged plant network capturing the dynamics of the structural system and the converged action network approximating the optimal control law, which are developed above, this section is devoted to simulation tests.

2.5.1 Wind and Wave Conditions

The turbulent wind is generated based on the IEC Kaimai Spectral Model with NTM in TurbSim [159], and the wave condition is generated by the HydroDyn module in FAST based on JONSWAP spectrum. Two extreme and two normal environmental conditions are considered first to analyse the control performance. Then a range of wind/wave conditions are included to further demonstrate the effectiveness of the ADP controllers. For the two extreme events (Event E1 and Event E2), which were recorded in the report [3], the main hub-height longitudinal wind speeds are respectively 22m/s and 24m/s, the turbulence intensity is category B, and the peak-spectral periods of the incident waves are 13.4s and 15.5s with the significant wave heights of 4.7m and 5.5m respectively. For the two normal events (Event N1 and Event N2), the main hub-height longitudinal wind speeds are respectively 9m/s and 18m/s, the turbulence intensity is category A, and the peak-spectral periods of the incident waves are 12s and 11s with the significant wave heights of 2m and 4.5m respectively. For the remaining cases (ranging from normal to extreme conditions), the wind speed increases from 9m/s to 24m/s with an interval of 3 m/s, and B level turbulence intensity is used for all the cases. The corresponding significant wave heights increase linearly from 2m to 5.5m and peak-spectral periods increase linearly from 12s to 15.5s.

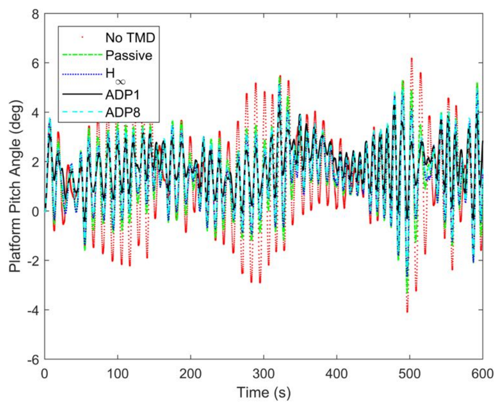
2.5.2 Performance Evaluation and Discussions

The simulation results are given here, including the calculations with no TMD, passive TMD, HMD using H_∞ controller, and HMD using a series of ADP controllers (ADP1-ADP8).

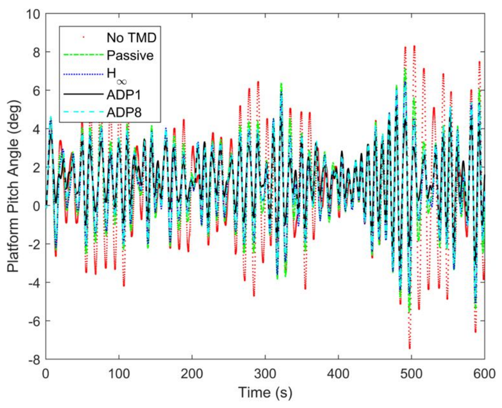
The standard deviation (SD) of the platform pitch angle and the corresponding HMD power consumption for the two normal events (N1 and N2) and the two extreme events (E1 and E2) are given in Table 2.1. Figure 2.5 shows the corresponding time responses. First, the performances of the ADP8 controller and the H_∞ controller are compared because their active power consumption are similar. In the two normal events, the SDs of the platform pitch displacement are reduced by 12.34% and 11.69% using the controller ADP8, compared with the passive case, while they are reduced by 10.83% and 9.64% using the H_∞ controller. It is concluded that the ADP controller and the H_∞ controller perform similarly in normal events. However, in the two extreme events, as can be seen from Figure 2.5(c-d) and Table 2.1, the platform pitch displacement with the controller ADP8 is much smaller than the ones with the H_∞ controller. Compared with the passive case, the SDs of the pitch displacement are reduced by 14.64% and 10.15% using the controller ADP8 while they are reduced by 8.72% and 2.67% using the H_∞ controller. In conclusion, a clear advantage of the ADP controller over H_∞ controller is observed in the extreme events but the ADP controller performs only slightly better in the normal events. This observation is reasonable because the H_∞ controller is expected to perform well in the linear range of the dynamic systems and the ADP controller shows its advantages in strongly nonlinear situations.

The simulation results of ADP1, the controller with the most effective vibration suppression performance, are also given in Figure 2.5, with the SDs of the platform pitch displacement reduced by 38.53% and 35.56% in the events N1 and N2, and by 41.01% and 40.43% in the events E1 and E2. In addition, the HMD power consumptions are $719.25kW$ and $863.59kW$ in E1 and E2 and $236.73kW$ and $657.62kW$ in N1 and N2. In [9], an average reduction of 18.1% for the platform pitch root-mean-square by the generalized H_∞ controller was reported with an average power consumption of $684kW$. In addition, they stated that their generalized H_∞ controller was not able to work under the extreme wind and wave conditions. The results here clearly demonstrate the great advantage of the ML-based approach over the H_∞ control approach in the structural control of floating wind turbines.

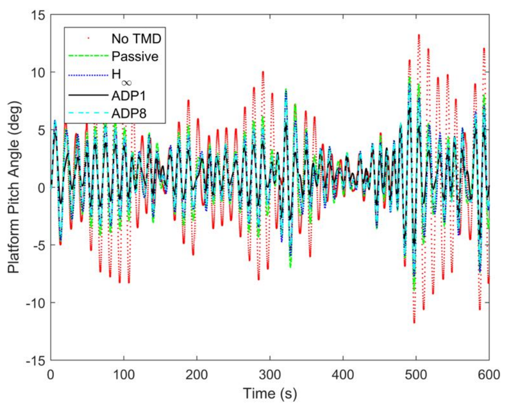
By changing the penalty coefficient B_u^* related to the HMD force magnitude in the utility function, a set of controllers have been obtained which consider the trade-off between the control performance and power consumption. Figure 2.6 shows the reduction of the SDs of the platform pitch displacements against the power consumptions for the events N1, N2, E1, and E2. It shows that ADP1 is the suitable choice if the ability of the wind turbines to withstand extreme conditions is the primary concern, while ADP5 may be more suitable if the cost of the active control (i.e.



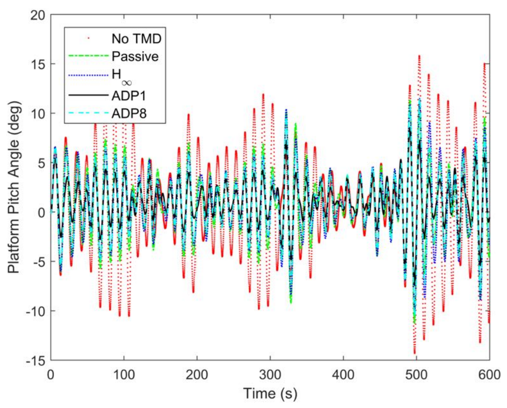
(a) N1



(b) N2



(c) E1



(d) E2

Figure 2.5: The simulation results including the ones with no TMD, passive TMD, HMD using H_∞ controller, HMD using ADP1 controller, and HMD using ADP8 controller, for the normal event N1 and N2, and the extreme event E1 and E2. Among ADP controllers, ADP1 controller is the most effective one in terms of vibration suppression and the ADP8 controller uses similar amount of HMD power as the H_∞ controller.

the HMD power consumption) becomes more concerned. It should be kept in mind that the active power consumption should be much less than the power generation of wind turbines in order to make the active control approach economically feasible.

To further evaluate the ADP controllers' performance, the simulations ranging from normal conditions to extreme conditions (from 9m/s to 24m/s with an interval of 3 m/s) are carried out and the results are given in Table 2.2. As can be seen, the proposed ADP controllers perform very well for all the cases.

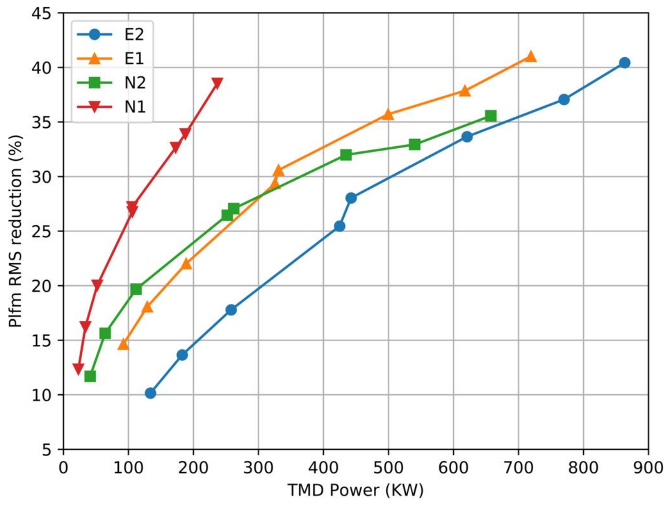


Figure 2.6: The reduction of the SDs of the platform pitch displacements against the HMD power, for the events E1, E2, N1, and N2. The symbols in each line (from left to right) represent the results obtained by ADP8, ADP7,..., ADP1.

Controller	No TMD	Passive	ADP1	ADP2	ADP3	ADP4	ADP5	ADP6	ADP7	ADP8	H_∞
B_u^* ($\times 10^{-4}$)	-	-	5	10	25	50	100	250	500	1000	-
N1:HMD Power	-	-	236.73	187.56	172.82	105.82	106.10	52.011	33.842	23.046	20.88
N1:Pitch SD	1.8533	1.4127	0.8684	0.9339	0.9515	1.0283	1.0346	1.1299	1.1837	1.2384	1.2597
N2:HMD Power	-	-	657.62	540.32	434.83	261.96	251.99	111.84	64.230	41.115	40.47
N2:Pitch SD	2.7134	2.0970	1.3513	1.4065	1.4263	1.5297	1.5423	1.6845	1.7693	1.8519	1.8949
E1:HMD Power	-	-	719.25	617.67	499.50	330.88	325.59	188.93	129.03	92.358	114.91
E1:Pitch SD	4.4224	2.9271	1.7267	1.8189	1.8821	2.0319	2.0678	2.2826	2.3984	2.4987	2.6718
E2:HMD Power	-	-	863.59	770.04	620.92	442.48	424.88	258.03	182.69	134.00	177.83
E2:Pitch SD	5.4536	3.5648	2.1236	2.2438	2.3652	2.5654	2.6575	2.9308	3.0783	3.2028	3.4697

Table 2.1: The results for the two normal events and the two extreme events, including the simulation with no TMD, passive TMD, HMD using H_∞ controller, and HMD using a series of ADP controllers. The standard deviation of the platform pitch angle (in degree) and the corresponding HMD power(in kW) are reported.

Controller	No TMD	Passive	ADP1	ADP2	ADP3	ADP4	ADP5	ADP6	ADP7	ADP8	H_∞
B_u^* ($\times 10^{-4}$)	-	-	5	10	25	50	100	250	500	1000	-
9m/s:HMD Power	-	-	237.16	186.46	172.36	106.02	105.54	51.92	33.69	22.84	20.58
9m/s:Pitch SD	1.8369	1.4015	0.8571	0.9207	0.9350	1.0122	1.0172	1.1144	1.1698	1.2253	1.2473
12m/s:HMD Power	-	-	274.50	215.01	205.30	126.28	124.71	59.46	37.83	24.55	22.48
12m/s:Pitch SD	1.9335	1.5401	1.0318	1.1230	1.1905	1.2437	1.2690	1.3259	1.3734	1.3960	1.4147
15m/s:HMD Power	-	-	291.43	236.96	224.31	135.74	140.73	66.73	42.61	28.46	30.97
15m/s:Pitch SD	2.1640	1.7273	1.0793	1.1398	1.1656	1.2584	1.2713	1.3955	1.4636	1.5298	1.5930
18m/s:HMD Power	-	-	424.41	360.42	325.27	211.48	217.04	115.21	78.39	54.36	64.94
18m/s:Pitch SD	3.1292	2.3123	1.3559	1.4225	1.4653	1.5963	1.6209	1.7979	1.8935	2.0019	2.1246
21m/s:HMD Power	-	-	613.38	532.42	454.32	311.65	313.37	183.77	128.66	92.36	115.01
21m/s:Pitch SD	4.2646	2.9511	1.7295	1.8237	1.8980	2.0484	2.0971	2.3410	2.4476	2.5410	2.7425

Table 2.2: The results for a range of wind/wave conditions (wind speed from 9m/s to 24m/s), including the simulation with no TMD, passive TMD, HMD using H_∞ controller, and HMD using a series of ADP controllers. The standard deviation of the platform pitch angle (in degree) and the corresponding HMD power(in kW) are reported. The 24m/s case is exactly the same as E2 in Table 2.1 thus omitted here.

2.6 Conclusions

The RL-based structural control of floating wind turbines has been investigated. An HMD was installed on the floating platform in order to reduce the platform vibration, and the ADP approach was employed to obtain the optimal control law. The design of NN structure and its implementation on the modern large-scale ML platform Tensorflow were proposed. Three networks were included in the whole NN structure, including a plant network, a critic network, and an action network. After training, the approximate optimal controller was obtained, based on the plant network which captured the nonlinear dynamics of the structural system. The simulation results showed that the designed controller performed very well in both normal and extreme conditions, with the standard deviation of the platform pitch displacement being reduced by around 40%. A clear advantage of the ADP controllers over the H_∞ controller was observed, especially for extreme wind/wave conditions - the scenarios that must be considered seriously in offshore wind technology.

In addition, the developed algorithm allows to consider the trade-off between the control performance (i.e. the reduction of the structural vibration) and the power consumption. A series of ADP controllers were obtained by varying the penalty term in the network training. As expected, the control performance increased with the increase of power consumption. It is worth noting that in practice, the passive TMD is expected to work alone most of the time and the active part is only activated when the vibration is above a certain limit.

Chapter 3

Quantification of Parameter Uncertainty in Wind Farm Wake Modelling

3.1 Introduction

This chapter investigates the parameter uncertainty of analytical wake models in wind farm wake predictions. In the literature, the empirical parameters in analytical wake models were usually calibrated against high-fidelity simulations or measurement data [40] but the underlying uncertainty was usually ignored. Thus only fixed-point predictions can be carried out for the quantities of interest (QoIs). In this work, the high-fidelity CFD data, generated by the LES flow solver SOWFA [160, 161], is used to calibrate the analytical wake model, using FLORIS [40] as an example, in terms of the empirical parameters' probability density functions (PDFs) in the Bayesian UQ framework. The resulting model with parameter uncertainty (called stochastic FLORIS) can predict the statistics of the QoIs which include the information for both the mean value and the corresponding uncertainty while the fixed-point calibration can only predict a single value. An apparent advantage of this work is that it can be used to not only maximise the average power but also minimise the power fluctuation while the fixed-point prediction of the QoIs cannot be used for the latter.

The main contribution of this work is the first application of UQ method in wind farm wake modelling and the detailed analysis of the resulting stochastic model. This work paves the way for wind farm predictions with quantified uncertainty and the proposed framework can be easily applied to other wake models. The remaining part of this chapter is organised as follows: the Bayesian UQ approach is described in Section 3.2. The application of the UQ approach in the wake model FLORIS is described in Section 3.3, where the formulation of FLORIS and the procedure of high-fidelity data generation are given. The results are given in Section 3.4, including the parameter uncertainty of FLORIS inferred from the high-fidelity data and the evaluation of the prediction performance of the developed FLORIS model with parameter uncertainty. The conclusions are drawn in Section 3.5.

3.2 Bayesian Uncertainty Quantification Framework

The UQ approach used in this work is briefly described in this section and further details can be found in [72,73]. In Bayesian UQ framework, various forms of uncertainties are represented through random variables, which are usually characterized by their PDFs. Here for the parameter uncertainty, model parameters are treated as random variables. According to Bayes' rule, the posterior distributions of model parameters can be obtained by

$$p(\mathbf{z}|\mathbf{d}) \propto p(\mathbf{d}|\mathbf{z})p(\mathbf{z}), \tag{3.1}$$

where p($\mathbf{z}$) represents the prior distribution of the model parameters $\mathbf{z}$ and p($\mathbf{d}|\mathbf{z}$) represents the likelihood of the experimental observation $\mathbf{d}$ given $\mathbf{z}$. A stochastic model needs to be constructed in order to obtain the likelihood. In this work it is constructed by simply modelling the model inadequacy through a multiplicative Gaussian random variable:

$$\tilde{\mathbf{d}} =(1+\boldsymbol{\eta})\mathscr{M}(\mathbf{x},\mathbf{z}), \tag{3.2}$$

where $\tilde{\mathbf{d}}$ is the true value of the experimental observation, $\mathscr{M}(\mathbf{x},\mathbf{z})$ is the prediction of $\tilde{\mathbf{d}}$ by the computer model which depends on the explanatory variable $\mathbf{x}$ (e.g. turbine yaw angle) and the model parameter $\mathbf{z}$, and $\boldsymbol{\eta}$ is a random vector with each component η_i being zero-mean, independent and identically distributed Gaussian, i.e. $\eta_i \sim \mathscr{N}(0,\sigma^2)$. $\tilde{\mathbf{d}}$ can be related to the experimental observation $\mathbf{d}$ as:

$$\mathbf{d} = \tilde{\mathbf{d}} + \mathbf{e}. \tag{3.3}$$

Here $\mathbf{e}$ represents the measurement error, which is modelled as a zero mean, independent and identically distributed Gaussian, i.e. $e_i \sim \mathscr{N}(0,\sigma_e^2)$. σ_e is determined from the corresponding experiments. From Equation 3.2 and 3.3, the model output can be related to the experimental observation as

$$\mathbf{d} =(1+\boldsymbol{\eta})\mathscr{M}(\mathbf{x},\mathbf{z})+\mathbf{e}. \tag{3.4}$$

Thus,

$$\mathbf{d}|\sigma,\mathbf{z} \sim \mathscr{N}(\boldsymbol{\mu},\boldsymbol{\lambda}), \tag{3.5}$$

where

$$\boldsymbol{\mu}=\mathscr{M}(\mathbf{x},\mathbf{z}) \text{ and } \boldsymbol{\lambda}=\mathscr{M}^T(\mathbf{x},\mathbf{z})\sigma^2\mathscr{M}(\mathbf{x},\mathbf{z})+\sigma_e^2\boldsymbol{I}. \tag{3.6}$$

Equation 3.1 can then be recast as:

$$p(\boldsymbol{\theta}_M|\mathbf{d}) \propto \frac{1}{\sqrt{(2\pi)^{N_d}|\boldsymbol{\lambda}|}}exp(-\frac{1}{2}\boldsymbol{\delta}^T\boldsymbol{\lambda}^{-1}\boldsymbol{\delta})p(\boldsymbol{\theta}_M) \tag{3.7}$$

where $\boldsymbol{\theta}_M$ denotes $\{\sigma,\mathbf{z}\}$, N_d is the dimension of the experimental observation, $|\boldsymbol{\lambda}|$ represents the determinant of $\boldsymbol{\lambda}$, and $\boldsymbol{\delta}=\boldsymbol{d}-\mathscr{M}(\mathbf{x},\mathbf{z})$.

Then a sampler is employed to obtain the posterior samples according to Equation 3.7 and the kernel estimation is used to evaluate the posterior PDFs of the model parameters. In this work the adaptive Metropolis-Hastings Markov chain Monte Carlo (MCMC) sampler [162], as implemented in the R [163] package MHadaptive [164], is employed. Once the calibration is completed, the model prediction can be carried out by propagating the posterior PDFs of model parameters through the computer model to obtain the PDFs of the QoIs. The so-obtained PDFs are in fact the posterior distribution of the QoIs given the experimental observation:

$$\begin{aligned} p(\tilde{\mathbf{q}}|\mathbf{d}) &= \int p(\tilde{\mathbf{q}},\mathbf{z}|\mathbf{d})d\mathbf{z} \\ &= \int p(\tilde{\mathbf{q}}|\mathbf{d},\mathbf{z})p(\mathbf{z}|\mathbf{d})d\mathbf{z} \\ &= \int p(\tilde{\mathbf{q}}|\mathbf{z})p(\mathbf{z}|\mathbf{d})d\mathbf{z}, \end{aligned} \tag{3.8}$$

where $\tilde{\mathbf{q}}$ represents the QoIs, which can be the same quantity as the experimental observation or other flow quantities. The last step in Equation 3.8 follows by assuming $\tilde{\mathbf{q}}$ and $\mathbf{d}$ are conditionally independent given $\mathbf{z}$.

3.3 Wind Farm Wake Modelling with Quantified Uncertainty

The Bayesian UQ approach described above can be used for general fluid systems and is thus applied to wind farm wake modelling in this section. For this purpose, the computer model $\mathscr{M}(\mathbf{x}, \mathbf{z})$ and the experimental observation $\mathbf{d}$ in Equation 3.7 need to be specified. Here the computer model $\mathscr{M}(\mathbf{x}, \mathbf{z})$ is specified as the analytical wake model FLORIS, with $\mathbf{x}$ being the turbine yaw angle, $\mathbf{z}$ being the empirical parameters in FLORIS, and the output $\mathscr{M}$ being the flow field prediction. The detailed formulation of $\mathscr{M}(\mathbf{x}, \mathbf{z})$ is given in Subsection 3.3.1. The experimental observation $\mathbf{d}$ is generated by high-fidelity numerical experiments with the simulation details given in Subsection 3.3.2.

3.3.1 An Analytical Wake Model - FLORIS

The analytical wake model FLORIS is briefly described in this section, including the modelling of wake deflection, wake expansion and wake velocity. Further details can be found in [40].

For wake deflection, the turbine wake center $y_w(x)$ is determined by

$$y_w(x) = Y + \delta y_{w,rotation}(x) + \delta y_{w,yaw}(x), \tag{3.9}$$

where x is the downwind coordinate, Y is the turbine's crosswind location, and $\delta y_{w,rotation}(x)$ and $\delta y_{w,yaw}(x)$ represent the rotation-induced and yaw-induced wake lateral offset. They are formulated as:

$$\delta y_{w,rotation}(x) = a_d + b_d[x - X], \tag{3.10}$$

and

$$\begin{aligned}\delta y_{w,yaw}(x) = &\frac{\xi_{init}(a,\gamma)\left[15\left[\frac{2k_d[x-X]}{D}+1\right]^4 + \xi_{init}(a,\gamma)^2\right]}{\frac{30k_d}{D}\left[\frac{2k_d[x-X]}{D}+1\right]^5} \\ &- \frac{\xi_{init}(a,\gamma)D[15+\xi_{init}(a,\gamma)^2]}{30k_d},\end{aligned} \tag{3.11}$$

where X is the turbine's downwind location, D is the turbine rotor diameter, a is the axial induction factor, γ is the yaw angle, and $\xi_{init}(a, \gamma) = 2cos^2(\gamma)sin(\gamma)a[1 - a]$. Three empirical parameters, a_d, b_d, and k_d, are involved in this deflection model.

Model parameter	k_d	a_d	b_d	k_e	$m_{e,1}$	$m_{e,2}$	$m_{e,3}$
Left boundary	0.05	-50.0	-0.05	0.04	-1.0	-0.3	0.0
Right boundary	0.25	50.0	0.05	0.14	-0.2	0.3	4.0
Continued	a_U	b_U	$M_{U,1}$	$M_{U,2}$	$M_{U,3}$		
Left boundary	0.0	0.66	0.2	0.7	3.0		
Right boundary	10.0	2.66	0.8	1.3	15.0		

Table 3.1: The prior range of model parameters in the FLORIS model.

After determining the center of the wake location, the wake region behind the turbine is divided into three zones and the diameters of each zone are given by

$$D_{w,q}(x) = max(D + 2k_e m_{e,q}[x - X], 0), x > X, \tag{3.12}$$

where the index q represents the different zones. The three zones are the 'near wake' ($q = 1$), 'far wake' ($q = 2$), and 'mixing zone' ($q = 3$). k_e, $m_{e,1}$, $m_{e,2}$ and $m_{e,3}$ are the four empirical parameters involved in this wake expansion model. The velocity profile within zone q is then modelled as

$$U_{w,q}(x) = U[1 - 2ac_q(x)] \tag{3.13}$$

where U is the freestream wind speed, and

$$c_q(x) = \left[\frac{D}{D + 2k_e[x - X]M_{U,q}/cos(a_U + b_U\gamma)}\right]^2. \tag{3.14}$$

Five empirical parameters, a_U, b_U, $M_{U,1}$, $M_{U,2}$, and $M_{U,3}$, are used in calculating the wake profile. From the formulation above, it can be seen that the FLORIS model predicts the velocity profile at a specific downwind location as a piecewise constant function.

In total, there are 12 empirical model parameters in the FLORIS model. For Bayesian calibration, the prior distributions of these parameters, i.e. p($\mathbf{z}$), need to be specified. In this work, the uniform distribution is used and their prior ranges are given in Table 3.1. They are determined by trial and error and are kept as large as possible so that the posterior is mainly determined by the likelihood.

3.3.2 High-Fidelity Data Generation

SOWFA is employed here to generate high-fidelity CFD data for Bayesian calibration of the FLORIS model described in Subsection 3.3.1. SOWFA is a numerical solver developed based on OpenFOAM for the 3D LES of wind flow around wind turbine array in the atmospheric boundary layer, where the turbine rotors are represented by ADM or ALM. The detailed implementations and validations of SOWFA can be found in [57, 161]. First, a precursor simulation of neutral atmospheric boundary layer is carried out to obtain the initial flow field and inflow boundary conditions. The employed turbulent inflow has a mean hub-height freestream wind velocity of around 8m/s and a freestream turbulence intensity (FSTI) of 6%. For the subsequent wind farm simulations, wind turbines are modelled using ALM and the baseline pitch and torque control are defined as in [165]. A top view of the simulation domain at hub height is shown in Figure 3.1. The size of the simulation domain is $3000 \times 3000 \times 1000$m, with the inflow wind coming from southwest direction. For the mesh generation, a two-level local mesh refinement is used, as is suggested in [35]. The outer mesh dimension is $12 \times 12 \times 12$m, the inner mesh dimension is $3 \times 3 \times 3$m, and the dimension of the mesh in-between is $6 \times 6 \times 6$m. The total number of cells is 1.8×10^7. In this way, the mesh size around the turbine rotors is 3 meters so that the simulation can capture the detailed turbine wake dynamics. An NREL 5-MW baseline turbine is positioned in the simulation domain. For each turbine yaw angle, 1000-second simulations are carried out with a time step of 0.02s. The mean velocity field is then obtained by averaging the instantaneous flow field from 400s to 900s. Each high-fidelity simulation by SOWFA requires around 30 hours using 256 processors. The generated LES data is then used to carry out the Bayesian calibration in the next section.

3.4 Results

3.4.1 Model Calibration

In order to calculate the likelihood, the stochastic model is constructed according to Section 3.2. Since the parameter uncertainty is the main focus of the current investigation, the model inadequacy term is ignored in the remaining part of this chapter. SOWFA is employed for generating high-fidelity flow field data for three scenarios, i.e. $\gamma = -40°$, $\gamma = 0°$, and $\gamma = 40°$. Then the hub-height velocity profile at 5 rotor diameters downstream behind the wind turbine is extracted to calibrate the FLORIS model, as the wind farm with downwind spacing of 5 rotor diameters is of great practical interests and has been studied previously for wake modelling and wind farm control [40]. The measurement error is estimated as the zero-mean Gaussian with $\sigma_e = \text{FSTI} \times U_\infty$. 12000 MCMC samples of the model parameters are generated with a burn-in length (i.e. the length of the samples that

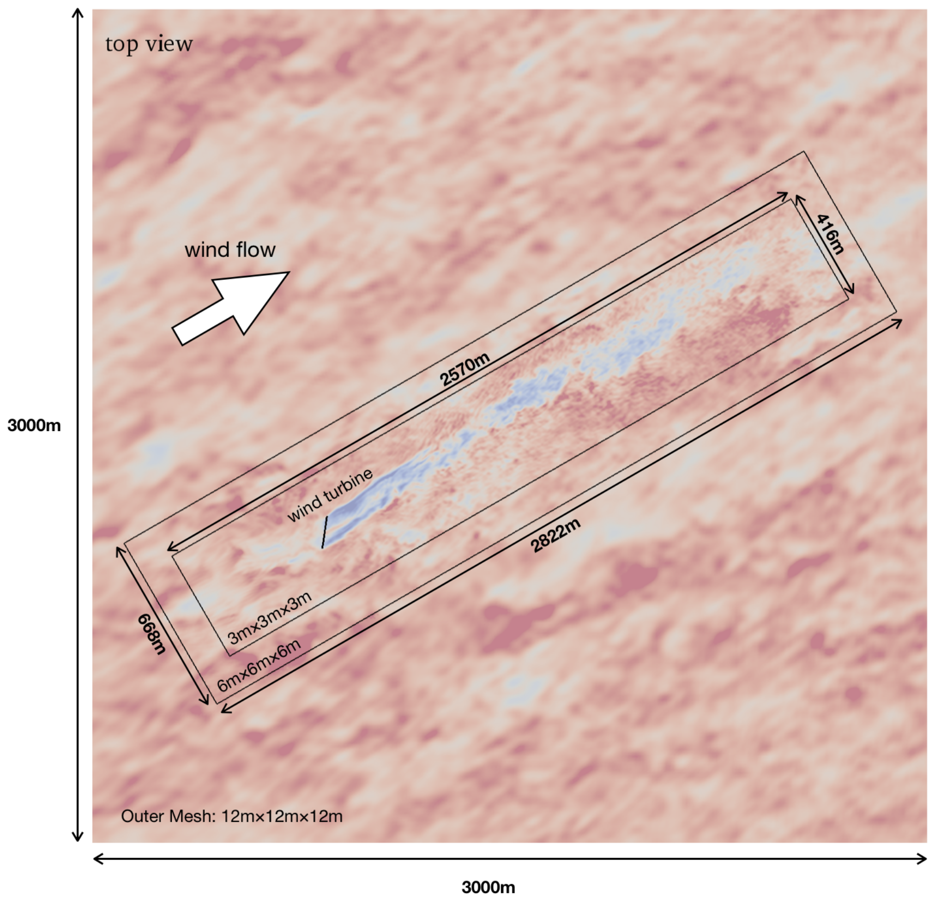


Figure 3.1: A top view of the simulation domain at turbine hub height. The contour shows the instantaneous flow field at 400s for the case with turbine yaw angle γ equal to $-40°$.

are discarded) set to 2000. The kernel estimation is used to estimate the posterior PDF from the MCMC samples. The results are shown in Figure 3.2, 3.3, and 3.4. For simplicity, the posterior distributions of the model parameters arising from modelling wake deflection, wake expansion and wake velocity are shown in different figures. As can be seen from Figure 3.2, 3.3, and 3.4, most of the model parameters are well identified. Among them, the correlation between a_d and b_d is strong, which agrees with the design of the wake deflection. In fact, both parameters intend to capture the same aspect of the wake flow, i.e., the rotation-induced lateral offset of the turbine wake. The nominal values of the model parameters reported in [40] are also shown in Figure 3.2, 3.3, and 3.4. They are the optimal values determined by matching the turbine power with the SOWFA results for a wind farm with 7

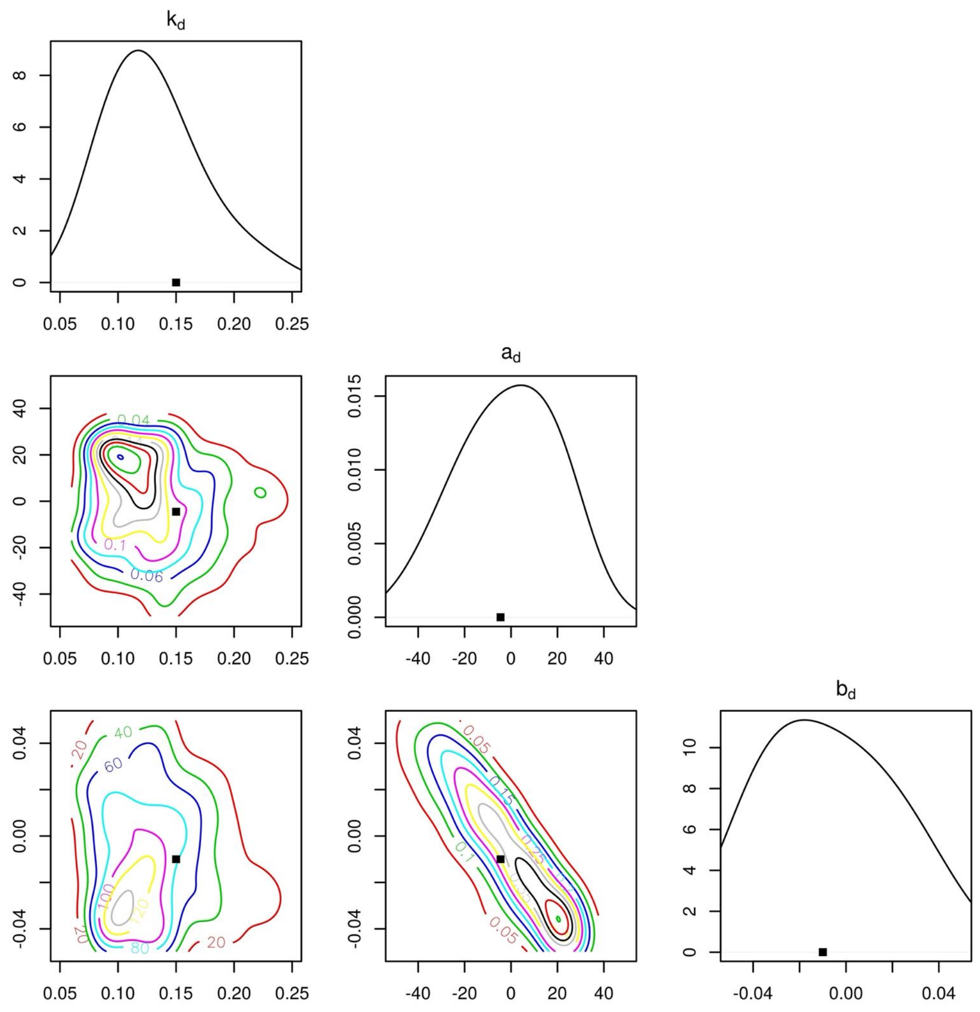


Figure 3.2: The posterior distribution of the model parameters arising from modelling wake deflection. The square points represent the nominal values of the model parameters reported in [40].

rotor diameters' downwind spacing. The Maximum A Posteriori (MAP) values of the calibration in this work are slightly different from their reported values, which is reasonable as the downwind spacing of the calibration cases in this work (5 rotor diameters) is slightly different from theirs. More importantly, the yaw angles of the calibration cases in this work are different from the ones in [40].

The posterior model check is then carried out by propagating the posterior PDFs of the model parameters through the FLORIS model. The predictions with quantified uncertainty for the velocity profiles at 5 rotor diameters downstream are shown in Figure 3.5. The results with nominal and MAP model parameters are also shown for comparison. As can be seen, the predicted velocity profile with quantified

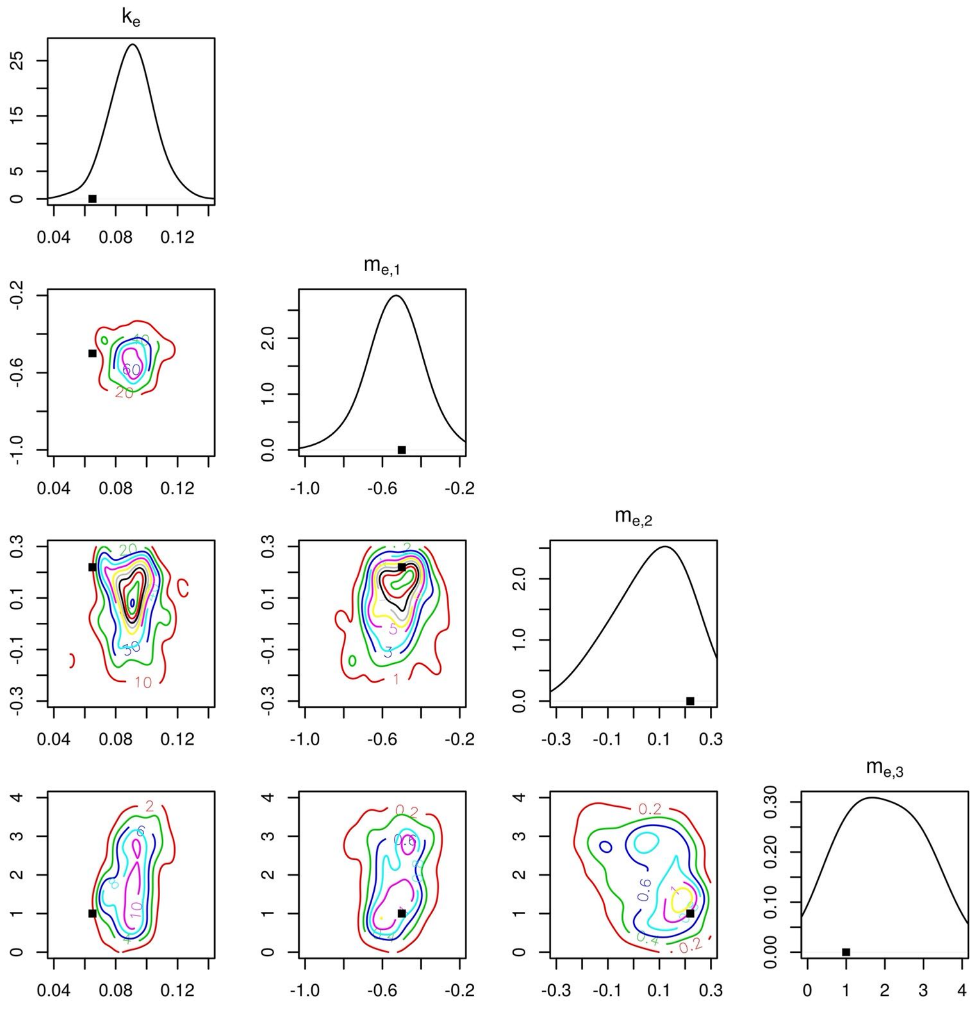


Figure 3.3: The posterior distribution of the model parameters arising from modelling wake expansion. The square points represent the nominal values of the model parameters reported in [40].

uncertainty matches well with SOWFA results, indicating the Bayesian calibration is done successfully. It is observed that the predictions match with SOWFA results much better in yawed cases than the non-yaw case (where a moderate discrepancy is observed), which indicates that the calibration focuses more on yawed conditions, as more high-fidelity data is used at yawed cases (including $\gamma = -40^\circ$ and $\gamma = 40^\circ$). In addition, since the FLORIS model divides the wake into three distinct zones and the velocity is determined separately in each zone as a constant function of the crosswind coordinate y, the velocity profile predicted with nominal/MAP model parameters is discontinuous at the zone boundary, while in reality the wake profile should be continuous. In this sense, the predicted mean value of the velocity profile

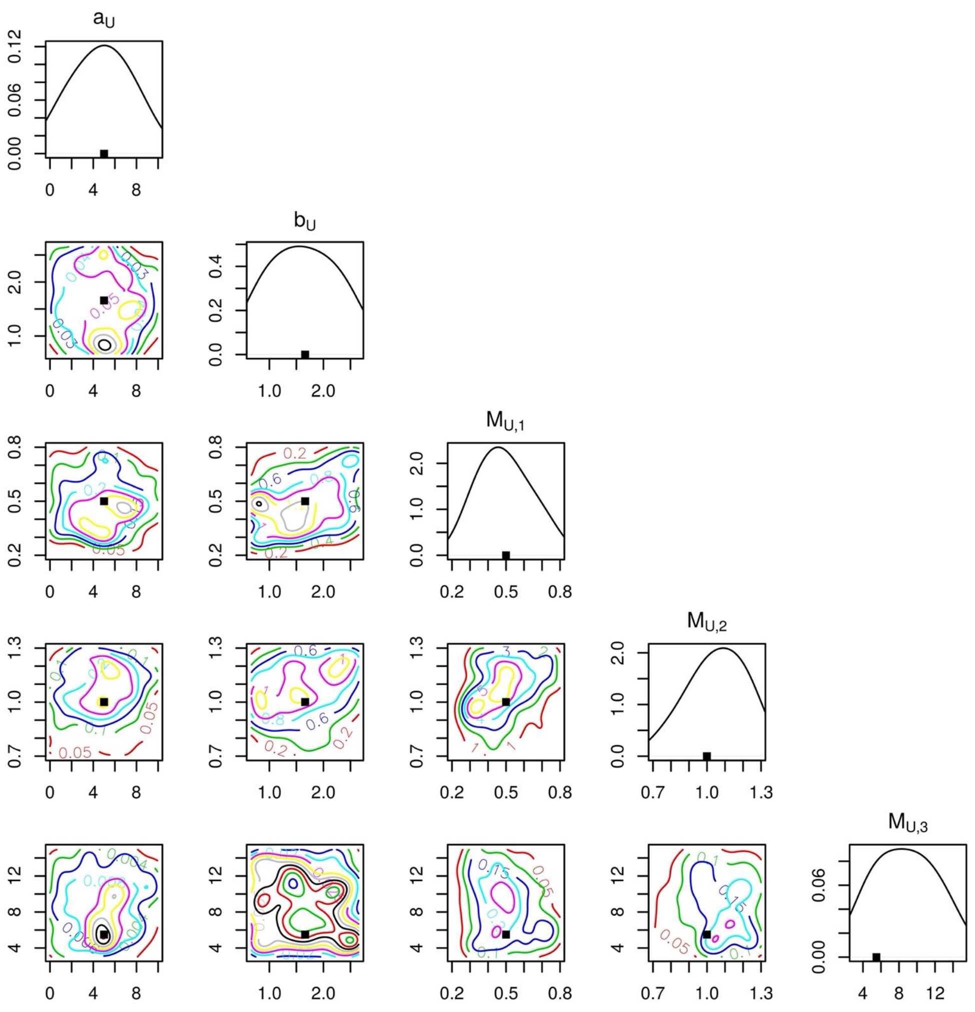
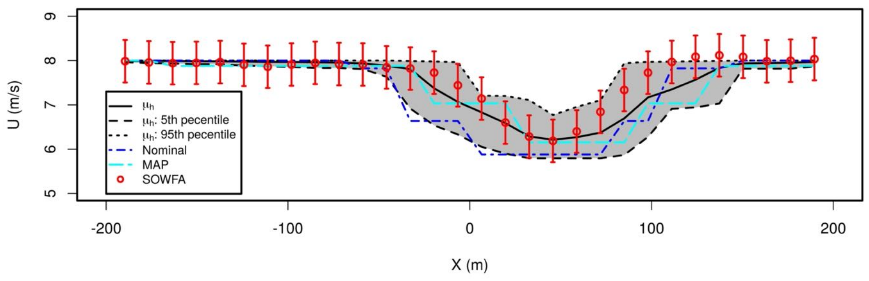


Figure 3.4: The posterior distribution of the model parameters arising from modelling wake velocity. The square points represent the nominal values of the model parameters reported in [40].

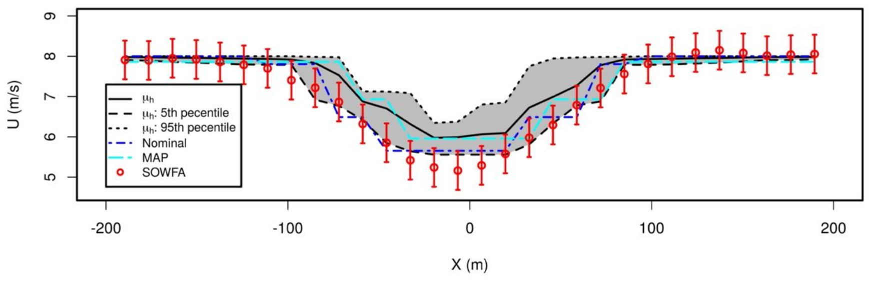
matches much better with SOWFA results than the nominal/MAP predictions.

3.4.2 Evaluation of the Stochastic FLORIS Model

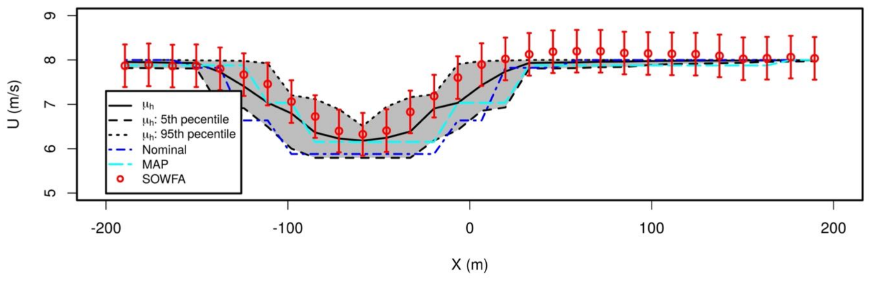
After model calibration, the FLORIS model with its parameters specified by their posterior distributions can be used to predict the statistics of the flow field, turbine power generation, turbine torque, etc. Hereby the FLORIS with uncertain model parameters is denoted as the stochastic FLORIS model, as the Bayesian calibration relies on the construction of the stochastic model described by Equation 3.2. However, it is worth noting that the stochastic FLORIS model is based on FLORIS (which is deterministic) and the prediction uncertainty is taken into ac-



(a) $\gamma = -40°$



(b) $\gamma = 0°$



(c) $\gamma = 40°$

Figure 3.5: The predictions of the velocity profiles at 5 rotor diameters downstream for three different yaw angles ($\gamma = -40°, 0°, 40°$). The high-fidelity SOWFA results are also included.

count only through parameter uncertainty. In the following parts, the stochastic FLORIS model's performance is evaluated in terms of predicting wind farm flow field and turbine power generation.

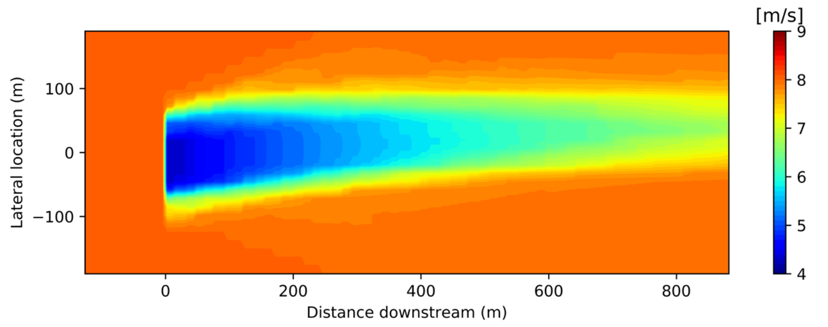
Flow Field Prediction

The posterior distributions of the 2D flow field at hub height are obtained by propagating the PDFs of the model parameters through FLORIS model. The mean value and standard deviation of the flow field are given in Figure 3.6 for the case of turbine yaw equal to -40°. The analysis for 0° and 40° turbine yaws are similar, thus omitted. For comparison, the FLORIS prediction with MAP model parameters is also included in Figure 3.6. The mean and standard deviation of the unsteady SOWFA results are given in Figure 3.7. As can be seen, the mean flow field given by stochastic FLORIS matches better with SOWFA than the MAP results. Furthermore, the standard deviation of SOWFA and stochastic FLORIS results shares a similar qualitative feature: the largest unsteadiness/uncertainty is in the 'mixing zone'.

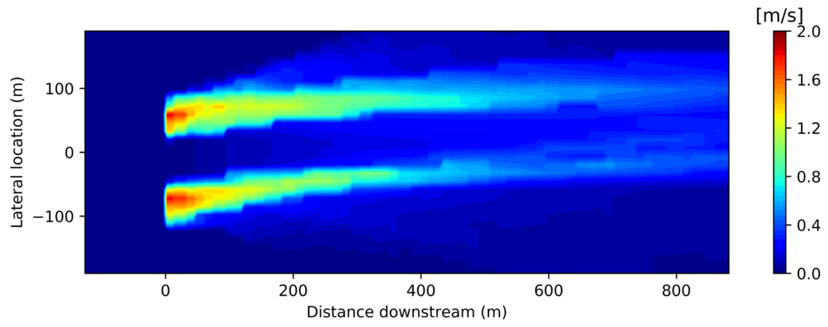
Turbine Power Generation Prediction

In order to evaluate the stochastic FLORIS model's performance in predicting turbine power generation, a series of high-fidelity simulations are carried out for the case of two turbines operating in a row. The simulation domain and mesh configuration of these two-turbine cases are the same as the one-turbine cases shown in Figure 3.1. The only difference is that another NREL 5-MW turbine is added in the flow domain located at 5 rotor diameter downstream of the first turbine. 30 samples of turbine yaw angles are generated by Latin Hypercube Sampling and the yaw angles are reported in Table 3.2. SOWFA is then employed to generate the high-fidelity data for the 30 cases and each SOWFA simulation requires around 30 hours using 256 processors. The turbine power generation is recorded for the simulation period of 500 seconds (from 400s to 900s). The turbine power generation is one of the primary concerns in wind farm design and control, thus the following parts focus on the evaluation of FLORIS and stochastic FLORIS in terms of predicting turbine power generation.

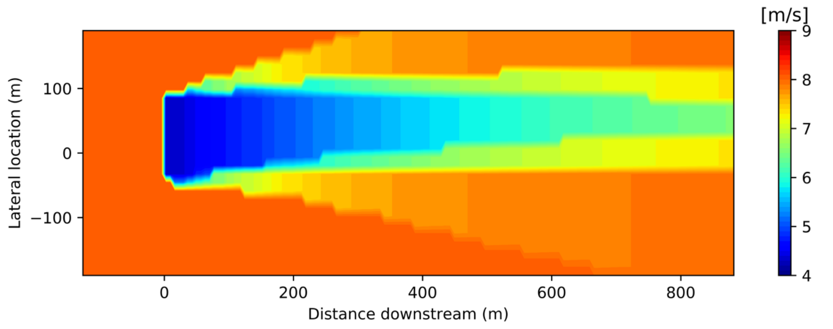
In order to obtain a better prediction of the turbine power generation, the input (inflow wind speed) uncertainty is also considered in the FLORIS and stochastic FLORIS prediction. The inflow wind speed is simply assumed as a Gaussian, i.e. $u_\infty \sim \mathscr{N}(\mu_u, \sigma_u^2)$ ($\mu_u = 7.83$ and $\sigma_u = 0.276$). The mean value and the standard



(a) Mean value predicted by stochastic FLORIS

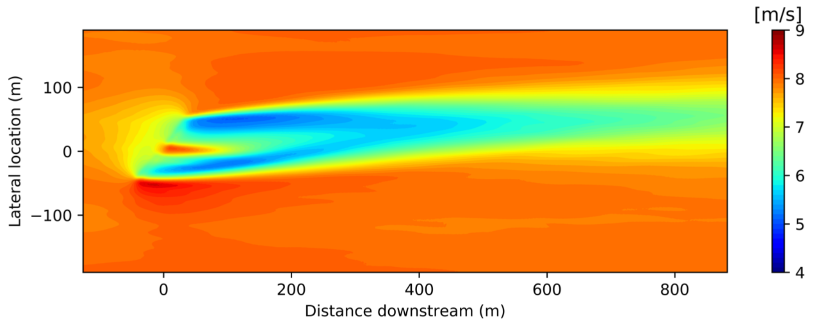


(b) Standard deviation predicted by stochastic FLORIS

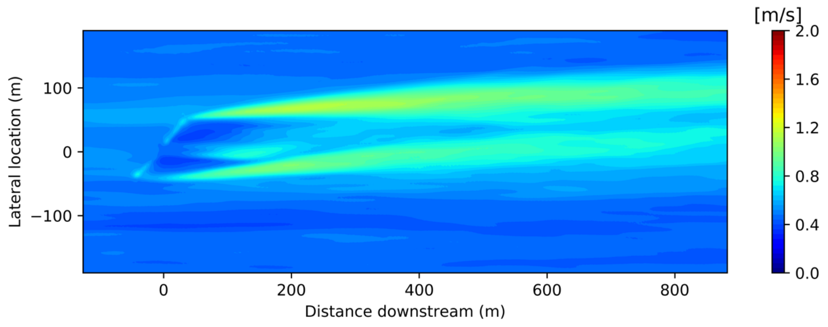


(c) FLORIS with MAP model parameters

Figure 3.6: The prediction of the flow field by the stochastic FLORIS model and FLORIS with MAP model parameters, for the case of turbine yaw equal to $-40°$.



(a) Mean value predicted by SOWFA



(b) Standard deviation predicted by SOWFA

Figure 3.7: The prediction of the flow field by SOWFA for the case of turbine yaw equal to $-40°$. The mean and standard deviation are calculated based on the instantaneous flow field from 400s to 900s.

Case No.	1	2	3	4	5	6	7	8	9	10
γ_1	16.5	0.1	12.7	2.3	-23.9	21.0	-6.7	-17.3	-3.1	-4.5
γ_2	4.7	-1.0	24.8	8.7	-5.1	1.7	7.8	18.6	-21.8	21.2
continued	11	12	13	14	15	16	17	18	19	20
γ_1	-14.1	-24.2	-14.0	8.1	24.4	-19.2	22.3	-9.6	-11.5	-28.4
γ_2	-14.3	-26.4	-8.2	17.2	-2.6	-28.6	-19.4	-23.9	13.4	-17.0
continued	21	22	23	24	25	26	27	28	29	30
γ_1	28.6	15.4	6.7	-1.5	5.2	19.8	-27.3	27.5	-20.7	11.1
γ_2	-10.5	10.1	-24.2	28.0	-7.5	-13.9	14.9	3.1	27.0	22.6

Table 3.2: The samples of turbine yaw angles generated by Latin Hypercube Sampling. γ_1 represents the front turbine yaw and γ_2 represents the rear turbine yaw.

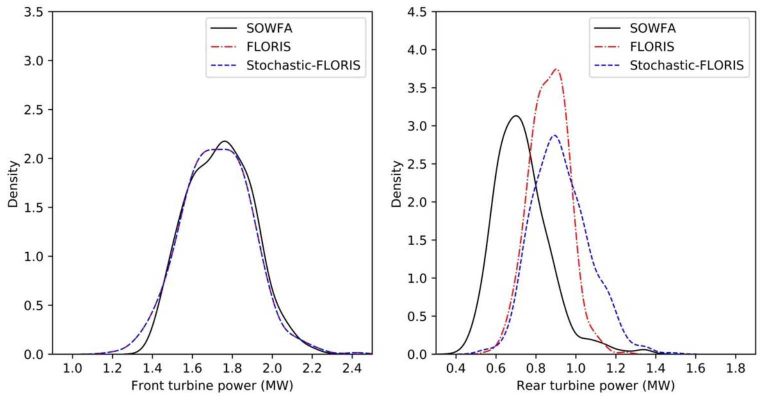
deviation are estimated from the unsteady inflow boundary condition of SOWFA simulations so that all the predictions are carried out with the same inflow wind condition.

First, the distributions of power generation predicted by SOWFA and by FLORIS are compared. Three typical cases are shown in Figure 3.8, where the front turbine is in the condition of no yaw, negative yaw, and positive yaw respectively. The results predicted by FLORIS with MAP model parameters and stochastic FLORIS are both included. For all three cases, FLORIS and stochastic FLORIS get the same results for predictions of power generation of the front turbine, as the inclusion of parameter uncertainty only introduces uncertainty to the downstream wake flow (thus only affecting the rear turbine's power generation). For the rear turbine, stochastic FLORIS predicts similar power fluctuations as the ones given by SOWFA, while FLORIS with only input uncertainty predicts much smaller power fluctuations. It is clear that stochastic FLORIS results match with SOWFA results much better than FLORIS. The FLORIS can be employed as an internal model for wind farm control to maximise the power generation, e.g. in [40]. Because the stochastic FLORIS can predict the distribution of power generation, it can be used for more purposes, such as maximising the average power, minimising the power fluctuation, guaranteeing certain amount of power generation with certain confidence level, etc.

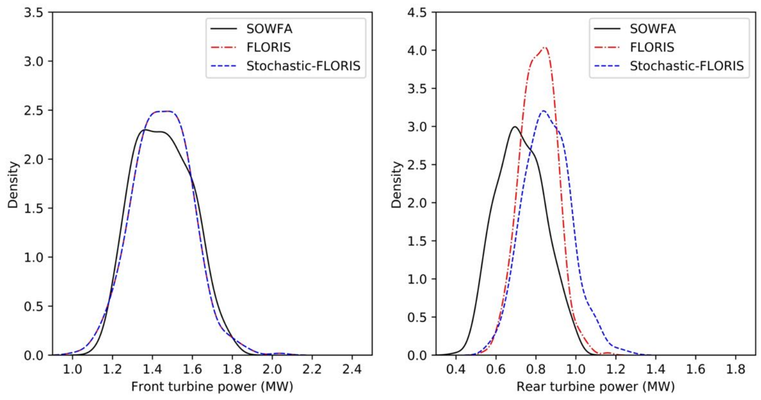
To further evaluate the performance of the stochastic FLORIS model, the predictions of turbine average powers and power fluctuations for all the 30 cases are given in Figures 3.9 and 3.10. The FLORIS and stochastic FLORIS get same results for the power predictions of the front turbine. They differ slightly in the predictions of rear turbine's average power, and their performances are similar compared to SOWFA results. However, compared with FLORIS, the power fluctuation prediction ability of the stochastic FLORIS is improved dramatically in all the 30 cases due to the inclusion of parameter uncertainty.

3.5 Conclusions

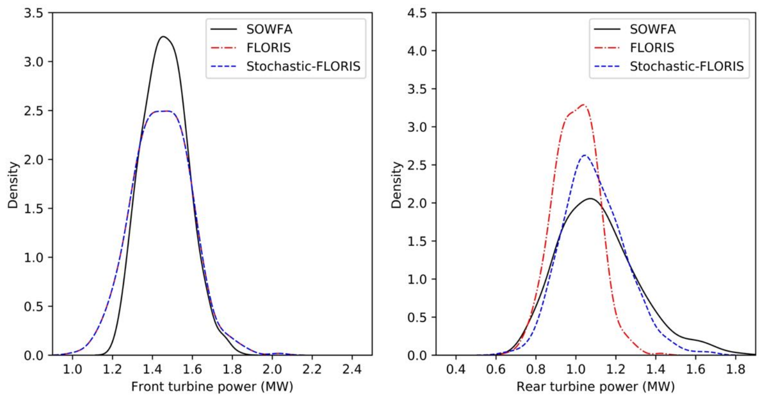
The parameter uncertainty of FLORIS model has been investigated in this work. LES of wind farm wakes was carried out with different turbine yaw angles and the generated high-fidelity flow field data was used to infer the model parameters. After model calibration, the posterior model check showed that the predicted mean velocity profile with the quantified uncertainty improved agreement with SOWFA relative to the standard FLORIS model. The predictions matched with SOWFA



(a) $\gamma_1 = 0.1^\circ$, $\gamma_2 = -1.0^\circ$

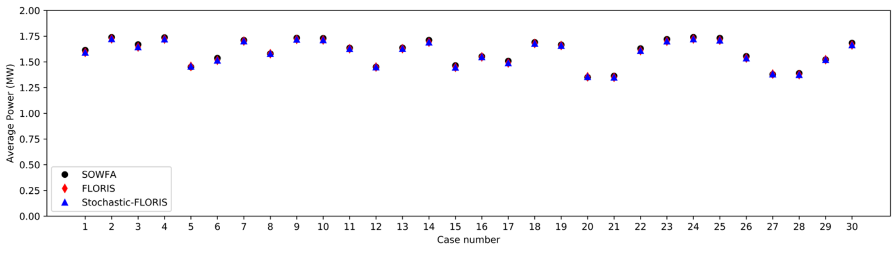


(b) $\gamma_1 = -24.2^\circ$, $\gamma_2 = -26.4^\circ$

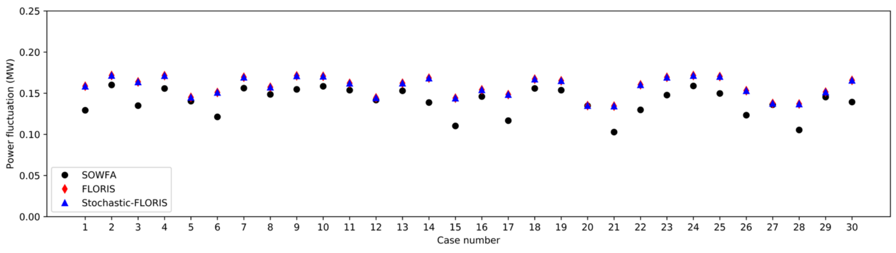


(c) $\gamma_1 = 24.4^\circ$, $\gamma_2 = -2.6^\circ$

Figure 3.8: The distribution of power generation predicted by SOWFA, FLORIS, and Stochastic-FLORIS. Three cases with different yaw angles are included.

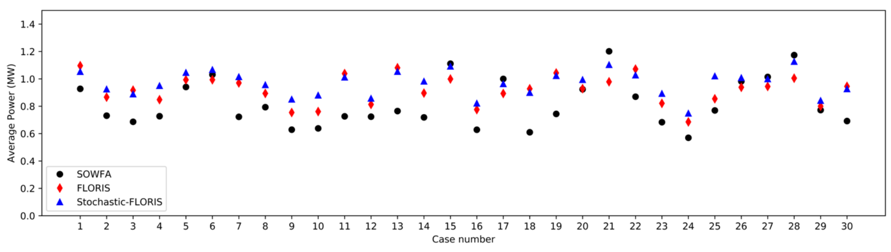


(a) Power generation average value

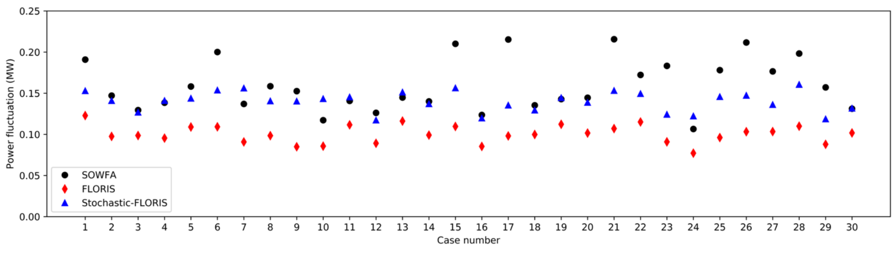


(b) Power generation standard deviation

Figure 3.9: Front turbine power generation predicted by SOWFA, FLORIS, and Stochastic-FLORIS.



(a) Power generation average value



(b) Power generation standard deviation

Figure 3.10: Rear turbine power generation predicted by SOWFA, FLORIS, and Stochastic-FLORIS.

results much better in yawed cases than the non-yaw case (where a moderate discrepancy was observed), indicating that the calibration was biased towards yawed conditions, as more high-fidelity data was used at these conditions. The prediction of other wind farm quantities, such as the hub-height flow field and the turbine power generation, was then carried out. The results showed that the inclusion of parameter uncertainty improved the flow field prediction compared to the predictions using point calibration such as maximum likelihood estimate, and a correct characteristic of uncertainty in the 'mixing zone' was predicted (see Figure 3.6), which agreed with the high-fidelity SOWFA results. As for the power generation, the inflow wind speed uncertainty was also included and the stochastic FLORIS performed similarly as the FLORIS model in terms of predicting average turbine power, but it performed much better in terms of predicting the turbine power fluctuation in all the test cases. As most of the existing analytical wake models are static with deterministic model parameters, the resulting stochastic model shows a great advantage over the original model, as it can give not only the static wind farm properties but also their statistical distributions.

Future research may involve the UQ of other wake models in order to reveal further insight in wake modelling and to improve the reliability of wind farm wake predictions. Another important research direction is the application of the resulting stochastic wake model in wind farm optimal design and control.

Chapter 4

Machine Learning Based Surrogate Modelling of Fluid Flows around Distributed Structures

4.1 Introduction

This chapter investigates the ML-based surrogate modelling of fluid flows around distributed structures and its application on wind farm wake modelling. The general framework of the developed surrogate modelling method is shown in Figure 4.1, where

$$\{\mu_0, \mu_1, ..., \mu_N\}, \{\mathscr{U}_0, \mathscr{U}_1, ..., \mathscr{U}_N\} \text{ and } \{\mathscr{U}_0^r, \mathscr{U}_1^r, ..., \mathscr{U}_N^r\}$$

represent the training samples of the input parameters, the corresponding CFD flow fields and the reduced coefficients of the flow fields, and μ_{test}, $\mathscr{U}_{test}^r$ and $\hat{U}_{test}$ represent the input parameter of interest, the predicted reduced coefficients of the flow and the full flow field prediction. Various dimensionality reduction techniques can be used in this framework as long as both the forward and the inverse transforms are available. The surrogate modelling framework consists of the offline and online stages. In the offline stage, as shown by the solid arrowed lines, multiple CFD simulations are carried out to generate the flow field data, using a set of input parameters obtained by a sampling strategy (e.g. Latin hypercube sampling). The generated flow fields are then reduced to their low-dimension representations by a dimension-

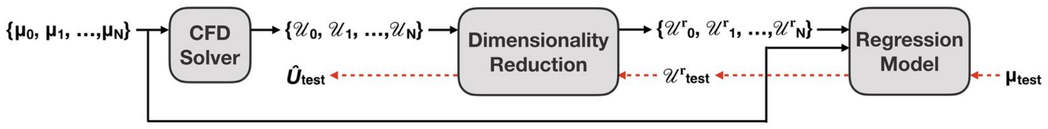


Figure 4.1: The general framework of the ML-based surrogate modelling.

ality reduction technique, and a regression model is constructed to approximate the mapping from the input parameters to the reduced coefficients. The regression model can be a simple curve fitting or a complex supervised ML model. As for the dimensionality reduction, it can be achieved by either traditional reduced basis methods or ML techniques. In fact, dimensionality reduction in ML shares a lot of similarities with model reduction in scientific computing. For example, principal component analysis (PCA) in ML is equivalent to POD in scientific computing in certain senses [124]. Once the offline stage is completed, the online stage, as shown by the dashed arrowed lines, is carried out by propagating the input parameter of interest to the reduced coefficients through the regression model and then predicting the flow field by the inverse transform of the dimensionality reduction process. In this chapter three dimensionality reduction techniques arising from both scientific computing and ML are investigated, including POD, ICA [166], and AE [167].

Another novelty of this work is that it extends the proposed surrogate modelling method to tackle distributed flow problems. Distributed fluid systems are quite common in daily life and industrial applications, such as the natural convection of heater array in heat exchangers [168, 169], the distributed roughness elements in boundary layer control [170], the heat transfer of building array in turbulent boundary layer [171], and the wake interactions of wind turbines within a wind farm [52–54]. The numerical simulation of such systems usually requires a lot of computational resources, and the optimisation/control of such systems is very difficult as the repetitive evaluations of CFD models with distributed flow parameters are needed. This motivates the work in this chapter on the surrogate modelling of distributed systems, which has not been investigated yet in the literature. A suitable surrogate modelling method for distributed systems should have the following features: (i) the surrogate model can simulate the fluid system with different flow parameters and preferably different layout; (ii) the method should be scalable such that the surrogate model can simulate the distributed system of different scales. In this chapter, a scalable surrogate modelling method for distributed fluid systems is proposed, where the whole fluid domain is first decomposed into subdomains, then the surrogate modelling for each subdomain is carried out by treating both the boundary information and the distributed flow parameters as the input parameters, and finally the whole flow field is obtained by coupling the flow field of each subdomain altogether with the consideration of the matching condition at the subdomain interface. The information exchange at the subdomain interface needs to be tackled specifically according to the types of the problems. Here, an iterative updating process is introduced for diffusion-dominant problems since each subdomain has an impact on all the other subdomains, while a sequential prediction process is sufficient for convection-dominant problems as the impact of the downstream structures on the upstream flow can be ignored.

Two test cases are used to demonstrate the efficiency, accuracy, and scalability of the proposed surrogate modelling method. The first one is the 1D Poisson's equation which is used to represent the application of the proposed method to diffusion problems. Then a large-scale industrial application, the surrogate modelling of wind farms, is investigated. SOWFA [160], an LES solver developed for the 3D flow simulation around wind turbine array, is used to generate the high-fidelity data. The application in wind farms aims at capturing the wake interactions between wind turbines, which have a large impact on the plant's overall performance. In order to retain the reliability of the high-fidelity model while achieving much faster predictions, data-driven surrogate modelling provides a promising direction for wind farm modelling. The surrogate modelling of a single turbine was studied in [172, 173], where the wake prediction was presented. In [174], the surrogate modelling of two turbine cases was investigated, but the scalability to the wind farm was not considered. In this chapter, the proposed surrogate modelling approach is applied for the challenging issue of wind farm wake modelling.

The remaining part of this chapter is organised as follows: the ML-based surrogate modelling method for distributed fluid systems is described in Section 4.2. The proposed method is applied to a diffusion-dominant problem (more specifically, 1D Poisson's equation) and a convection-dominant problem (more specifically, wind farm simulations) in Section 4.3 to demonstrate its scalability, efficiency, and accuracy. Finally the conclusions are drawn in Section 4.4.

4.2 Machine Learning Based Surrogate Modelling of Distributed Fluid Systems

In ML, dimensionality reduction is usually used to reduce high-dimension training input into its low-dimension representation before feeding it into a regression/classification model. For surrogate modelling of fluid systems, however, the goal is to predict the high-dimension flow field giving a few input parameters. The direct construction of a regression model to predict the high-dimension flow field is prone to overfitting, especially when there are not enough training data available. Therefore, the idea of surrogate modelling for fluid systems is to first reduce the high-dimension flow field into their low-dimension representation, then predict the reduced coefficients from the input parameters, and finally reconstruct the flow field based on the reduced coefficients. One way to achieve this is the use of reduced basis methods, e.g. POD, as is done in [125–128].

Here the reduced basis method is briefly described and then the equivalence of reduced basis method and other ML-based dimensionality reduction technique in the context of surrogate modelling is illustrated. Given N samples of input parameters $[\mu_0, \mu_1, ..., \mu_N]$ and the corresponding CFD flow fields $\mathbf{Z} = [\mathscr{U}_0, \mathscr{U}_1, ..., \mathscr{U}_N]$, the POD basis $\{\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, ..., \boldsymbol{\nu}_k, ...\}$ can be constructed by the singular value decomposition (SVD)

$$\mathbf{Z} = \mathbf{V}\mathbf{\Sigma}\mathbf{W}^T \tag{4.1}$$

where $\boldsymbol{\nu}_k$ is the k^{th} column vector of $\mathbf{V}$. Then after choosing the number of the POD basis as N_r, the flow field can be approximated by

$$\tilde{U}(\mu_i) = \sum_{k=1}^{N_r} \alpha_k(\mu_i)\boldsymbol{\nu}_k \tag{4.2}$$

where the POD coefficients $\boldsymbol{\alpha}(\mu_i) = [\alpha_1(\mu_i), \alpha_2(\mu_i), ..., \alpha_{N_r}(\mu_i)]$ are the reduced coefficients of the original flow field $\mathscr{U}_i$. Then a regression model $\mathscr{M}$ can be trained based on the training input $[\mu_0, \mu_1, ..., \mu_N]$ and the training target $[\boldsymbol{\alpha}(\mu_0), \boldsymbol{\alpha}(\mu_1), ...\boldsymbol{\alpha}(\mu_N)]$ such that

$$\mathscr{M}(\mu_i) \approx \boldsymbol{\alpha}(\mu_i) \tag{4.3}$$

After training, the prediction of the flow field can then be given by

$$\hat{U}(\mu_{test}) = \sum_{k=1}^{N_r} [\mathscr{M}(\mu_{test})]_k \boldsymbol{\nu}_k \tag{4.4}$$

where μ_{test} is the input parameter of interest. In fact, the reduced basis method here can be viewed as a dimensionality reduction technique in ML, where the forward

Algorithm 2 The surrogate modelling method using machine learning

```
1: % **Offline process**
2: Generate $N$ samples of the input parameters: $[\mu_0, \mu_1, ..., \mu_N]$.
3: Run the high-fidelity CFD solver $N$ times to generate the flow field $\mathbf{Z} = [\mathscr{U}_0, \mathscr{U}_1, ..., \mathscr{U}_N]$.
4: Use a chosen dimensionality reduction technique to obtain the reduced coefficients $[\mathscr{U}_0^r, \mathscr{U}_1^r, ..., \mathscr{U}_N^r]$ and the corresponding forward and inverse transforms $\mathbf{g}$ and $\hat{\mathbf{g}}^{-1}$.
5: Train the regression model $\mathscr{M}$ using the training input $[\mu_0, \mu_1, ..., \mu_N]$ and the training target $[\mathscr{U}_0^r, \mathscr{U}_1^r, ..., \mathscr{U}_N^r]$.
6: % **Online process**
7: Set the input parameter of interest $\mu_{test}$.
8: Predict the flow field as $\hat{\mathbf{g}}^{-1}(\mathscr{M}(\mu_{test}))$.
```

transform $\mathbf{g}$ and the inverse transform $\hat{\mathbf{g}}^{-1}$ are defined as:

$$[\mathbf{g}(\mathscr{U})]_k =< \mathscr{U}, \boldsymbol{\nu}_k >, 1 \leq k \leq N_r \tag{4.5}$$

$$\hat{\mathbf{g}}^{-1}(\boldsymbol{\alpha}) = \sum_{k=1}^{N_r} \alpha_k \boldsymbol{\nu}_k \tag{4.6}$$

where $<>$ denotes the inner product. From the above formulation, it is clear that the use of reduced basis methods in surrogate modelling can be replaced by any other dimensionality reduction techniques in ML as long as there exist both the forward transform $\mathbf{g}$ which maps the flow field into the reduced coefficients, and the inverse transform $\hat{\mathbf{g}}^{-1}$ which maps the reduced coefficients to the approximation of the flow field. After dimensionality reduction and regression model training, the flow field can then be predicted by

$$\hat{U}(\mu_{test}) = \hat{\mathbf{g}}^{-1}(\mathscr{M}(\mu_{test})) \tag{4.7}$$

The proposed surrogate modelling procedure is summarised as Algorithm 2. In this work, the fully-connected NN with one hidden layer is chosen as the regression model. POD, ICA, and AE are employed for dimensionality reduction. Hereby these methods are referred as POD-NN, ICA-NN, and AE-NN. The ML packages Scikit-learn [175] and Keras [176] are used to facilitate the implementation of the proposed algorithm.

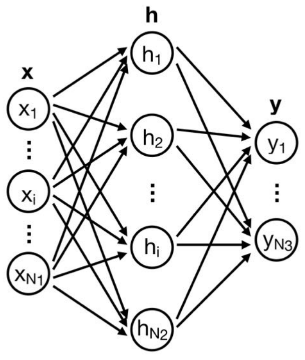


Figure 4.2: The illustration of a fully-connected NN with one hidden layer.

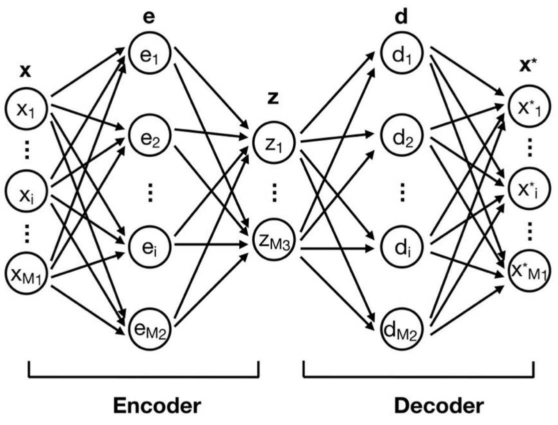


Figure 4.3: The illustration of an auto-encoder with three hidden layers in the whole neural network structure.

4.2.1 Regression Model

The regression model used in this work is a fully-connected NN with one hidden layer, as illustrated in Figure 4.2, and the corresponding input-output relation can be expressed as

$$\begin{aligned} \mathbf{h} &= \sigma(\mathbf{w}_1\mathbf{x} + \mathbf{b}_1), \\ \mathbf{y} &= \mathbf{w}_2\mathbf{h} + \mathbf{b}_2 \end{aligned} \tag{4.8}$$

where $\mathbf{w}_1$, $\mathbf{b}_1$, $\mathbf{w}_2$ and $\mathbf{b}_2$ are the training variables, and σ is the activation function. N_1, N_2, and N_3 in Figure 4.2 represent the input dimension, the hidden-layer neuron number and the output dimension. The NN training process involves the updating of the training variables w_{ij} based on the gradient $\partial J/\partial w_{ij}$, where J is the objective function to be minimised. Automatic differentiation is employed to calculate the gradients. The MSE between the target and the NN output is chosen as the objective function where a L2 regularisation term is further added in order to tackle overfitting. The Adam optimisation algorithm [158] is used for NN training in this work.

4.2.2 Dimensionality Reduction

Three types of dimensionality reduction techniques (i.e., POD, ICA, and AE) are employed in this work, to demonstrate the feasibility of using any dimensionality reduction technique with an inverse transform in the proposed surrogate modelling framework. As most of previous studies were based on POD, this work investigates the use of ICA and AE as the alternative dimensionality reduction technique in the context of surrogate modelling. ICA is a popular technique first developed for signal processing and it can also be used for general purpose dimensionality reduction. The interested readers may refer to [177] for more details. AE is an ML technique that makes use of NN to encode a high-dimension input to a low-dimension latent space and then decode it back. It consists of an encoder NN and a decoder NN, and is trained in a self-supervised manner. The AE used in this work is illustrated in Figure 4.3, where three hidden layers are included in the NN structure. The encoder part can be expressed as

$$\begin{aligned} \mathbf{e} &= \sigma(\mathbf{v}_1\mathbf{x} + \mathbf{a}_1), \\ \mathbf{z} &= \mathbf{v}_2\mathbf{e} + \mathbf{a}_2 \end{aligned} \tag{4.9}$$

where $\mathbf{v}_1$, $\mathbf{a}_1$, $\mathbf{v}_2$ and $\mathbf{a}_2$ are the training variables of the encoder, while the decoder part can be expressed as

$$\begin{aligned} \mathbf{d} &= \sigma(\mathbf{v}_3\mathbf{z} + \mathbf{a}_3), \\ \mathbf{x}^* &= \mathbf{v}_4\mathbf{d} + \mathbf{a}_4 \end{aligned} \tag{4.10}$$

where $\mathbf{v}_3$, $\mathbf{a}_3$, $\mathbf{v}_4$ and $\mathbf{a}_4$ are the training variables of the decoder. M_1, M_2, and M_3 in Figure 4.3 represent the original data dimension, the hidden-layer neuron number and the reduced data dimension. In this work, for simplicity, the hidden-layer neuron number of the encoder and the decoder is assumed to be the same and is set as twice of the reduced dimension. The NN training is carried out using Adam optimisation algorithm to minimise the MSE between the NN output and its target value. The target value is the same as the input of the training dataset, thus it is termed as self-supervised training.

4.2.3 Extension to Distributed Flow Problems

The surrogate modelling method is extended to tackle distributed flow problems here. A typical example of a distributed fluid system is illustrated in Figure 4.4, where M cylinders of diameters $\{d_1, d_2, ..., d_M\}$ are positioned in a rectangular flow domain. The cylinder diameter is the distributed parameter in this example and in fact the distributed parameter can be an array of any other properties of the structures (the surface roughness, thermal conductivity, etc.). The surrogate modelling of this problem is formulated as how to construct a model to predict the whole

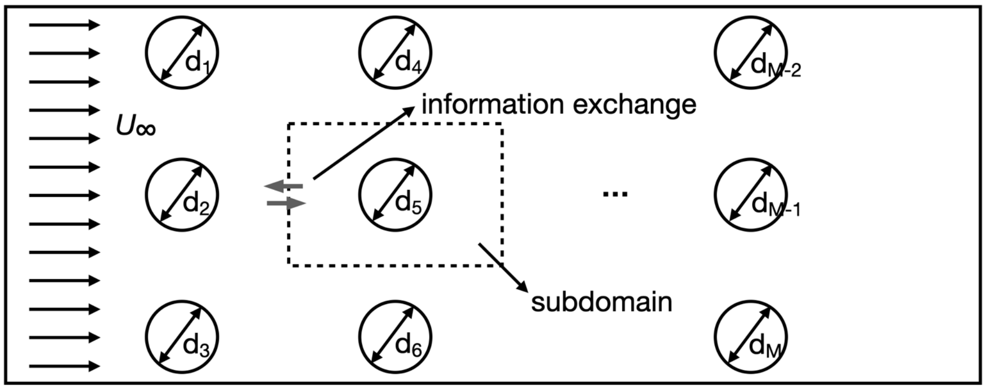


Figure 4.4: The illustration of a general distributed fluid system.

flow field around the cylinder array given the values of $d_1, d_2, ..., d_M$. The method proposed in the previous section can be used directly by treating $[d_1, d_2, ..., d_M]$ as the input parameter μ. However, this approach has two fundamental flaws: (i) the training of the surrogate model requires a significant number of CFD evaluations, as the whole flow domain simulation is regarded as a single training sample and the dimension of μ is very high such that a large sample size is required in order to cover the input parameter space. (ii) the so-constructed surrogate model is not scalable that it can only be used to simulate the distributed system of the same scale as the training samples.

The scalable surrogate modelling method proposed in this chapter can solve these issues. First, the whole flow domain is decomposed into subdomains with each subdomain containing one distributed structure, as illustrated by the dashed rectangular in Figure 4.4. Then the surrogate modelling is carried out for each subdomain by treating both the distributed parameter of the structure inside the subdomain and the boundary information as the input parameter μ. Finally the flow fields of all the subdomains are combined together with the consideration of the matching condition at the subdomain interface. The proposed approach is inspired naturally by the domain decomposition in high-performance computing in CFD, where the whole domain/mesh is divided into subdomains/blocks and each MPI thread handles only its assigned subdomain with the interface information exchange between MPI threads. The approach can be viewed as employing the surrogate model of individual subdomain to replace the task of each MPI thread, thus greatly reducing the online prediction time. The whole surrogate modelling procedure is summarised as Algorithm 3, where an iterative process is introduced to update the flow quantities at the interface in order to enforce the physical constraints at

Algorithm 3 The surrogate modelling method for distributed flow problems

```
1: % Offline process
2: Generate $N$ samples of the input parameters: $[\mu_0, \mu_1, ..., \mu_N]$, where $\mu_i = [d_i, I_i]$.
   $d_i$ represents the distributed parameters of a single structure and $I_i$ represents
   the flow quantities at the subdomain boundary.
3: Run the high-fidelity CFD solver multiple times to obtain the flow fields in a
   single subdomain $[\mathscr{U}_0, \mathscr{U}_1, ..., \mathscr{U}_N]$, where $\mathscr{U}_i$ is generated with the distributed
   parameter $d_i$ and the boundary condition given by $I_i$.
4: Carry out surrogate modelling using Algorithm 2. Then the flow field in a single
   subdomain $\hat{U}_{test}$ can be predicted by the surrogate model given $[d_{test}, I_{test}]$.
5: % Online process
6: Set the distributed parameters of all the $K$ structures: $[\tilde{d}_1, \tilde{d}_2, ..., \tilde{d}_K]$.
7: Initialise the flow quantities at the boundary of each subdomain $[\tilde{I}_1, \tilde{I}_2, ..., \tilde{I}_K]$.
8: while True do
9:     for j = 1 to K do
10:        Predict the flow field $\tilde{U}_j$ in $j^{th}$ subdomain given the input parameter $\tilde{d}_j$
   and boundary information $\tilde{I}_j$.
11:    end for
12:    Update $[\tilde{I}_1, \tilde{I}_2, ..., \tilde{I}_K]$ based on the surrogate model prediction from both
   sides of the interfaces.
13:    if the changes of $[\tilde{I}_1, \tilde{I}_2, ..., \tilde{I}_K]$ are very subtle. then
14:        The updating process converges. Break.
15:    end if
16: end while
17: Combine $[\tilde{U}_1, \tilde{U}_2, ..., \tilde{U}_K]$ to obtain the prediction of the whole flow field.
```

the interface. The detailed updating rule in Line 12 of Algorithm 3 is problem-dependent. Two numerical examples are given in the rest of the chapter with the detailed implementation of the updating rules. As the iterative updating at all subdomain boundaries is very challenging for the wind farm case and the wake interactions take place mainly in the downstream direction, this work only considers the impact of the upstream subdomains on downstream subdomains, ignoring the interactions in the lateral directions and in the upstream direction.

4.3 Numerical Results

The application of the proposed method on 1D Poisson's equation and on wind farm wake modelling is described in this section. The first test case demonstrates the accuracy, efficiency, and scalability of the proposed surrogate modelling method in diffusion-dominant problems. The second test case further demonstrates the ability of the proposed method in modelling large-scale fluid systems.

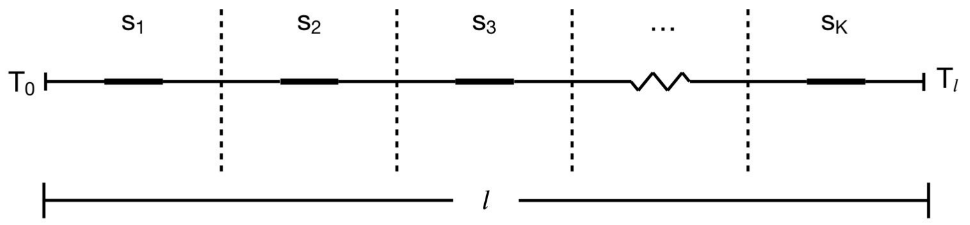


Figure 4.5: The 1D steady-state heat transfer problem under consideration.

4.3.1 Application on 1D Poisson's Equation

Problem Setup

The 1D steady-state heat transfer with distributed heat sources under consideration is illustrated in Figure 4.5, where a 1D domain of length l is shown with K uniformly-distributed heat sources of magnitudes $[S_1, S_2, ..., S_K]$. It can be described by the following equation

$$-\beta \frac{\partial^2 T}{\partial x^2} = q(x) \tag{4.11}$$

where the heat source term is defined as

$$q = \sum_{i=1}^{K} S_i \mathbb{I}_{(i-\frac{2}{3})\frac{l}{K} < x < (i-\frac{1}{3})\frac{l}{K}} \tag{4.12}$$

and the boundary condition is given as $T(0) = T_0$ and $T(l) = T_l$. Here $\mathbb{I}$ represents the indicator function. The surrogate modelling of this problem aims at predicting the temperature field efficiently given the boundary conditions (T_0 and T_l) and the distributed parameters $[S_1, S_2, ..., S_K]$.

In the following, the length of each subdomain l/K is set as 1 and the thermal diffusivity β is set as 1. The FTCS (forward-time central-space) scheme is implemented for numerically solving the equation. The mesh-dependence study shows that a uniform mesh of size 1/80 is sufficient. All calculations are deemed convergent when the RMSE of the temperature profiles between two consecutive time steps is less than 10^{-6}.

Results

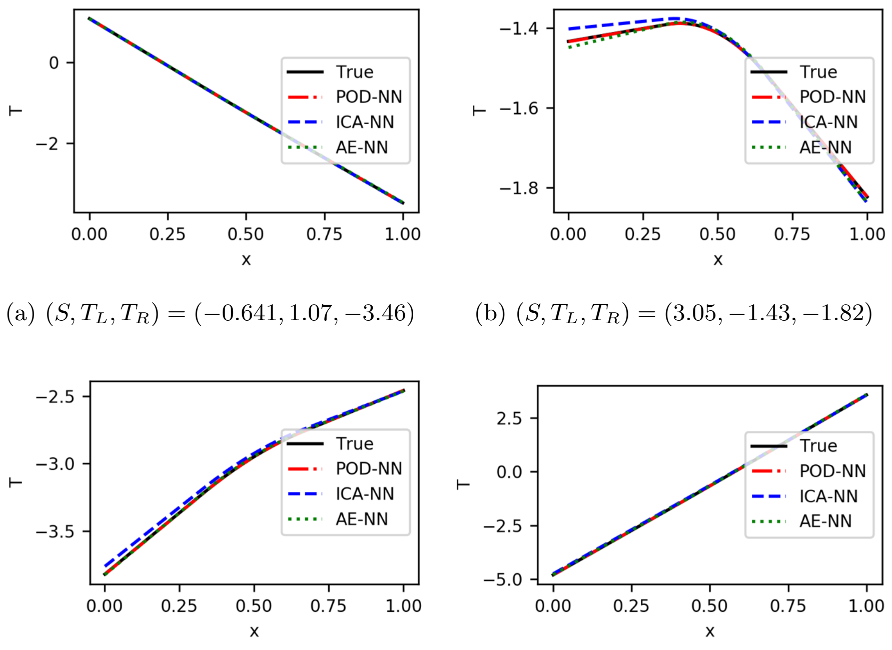
First the surrogate modelling of a subdomain with a single heat source is carried out. Four hundred samples of the input parameters $[\mu_0, \mu_1, ..., \mu_{400}]$ in the parameter space $\Omega = [-5.0, 5.0] \times [-5.0, 5.0] \times [-5.0, 5.0]$ are generated using Latin hypercube

Method	N_r	N_h	Activation	α_{NN}	ϵ_{mr}	ϵ_{all}
POD-NN	3	10	relu	10^{-6}	2.47×10^{-5}	2.68×10^{-3}
ICA-NN	3	12	relu	10^{-1}	2.47×10^{-5}	5.86×10^{-2}
AE-NN	3	8	relu	10^{-2}	2.41×10^{-3}	9.30×10^{-3}

Table 4.1: The optimal hyper-parameters of the three surrogate modelling methods for the 1D Poisson case.

sampling, where $\mu_i = [S_i, T_{Li}, T_{Ri}]$ with S_i, T_{Li} and T_{Ri} representing the magnitude of the heat source, the temperature at the left boundary and the temperature at the right boundary of the i^{th} sample. Then the Poisson's equation in a domain of length l/K is solved numerically for each sample of the input parameters to generate the temperature profiles. The generated data is then split into two parts, 320 training and validation samples and 80 test samples.

Three surrogate modelling methods, i.e. POD-NN, ICA-NN, AE-NN, are employed to build surrogate models to predict the temperature profile with the magnitude of the heat source, the temperature at the left boundary and the temperature at the right boundary as the model input. The temperatures at both boundaries and heat sources are scaled to zero mean and unit variance separately before feeding into the NN for training. The learning rate is set as 10^{-3}. A grid-search procedure is carried out to determine the optimal hyper-parameters in the dimensionality reduction and regression models, based on 4-fold cross-validation errors. The optimal hyper-parameters, the dimensionality reduction errors, and the prediction errors are given in Table 4.1 for all the three methods, where N_r represents the dimension of the reduced coefficient, N_h represents the hidden-layer neuron number of the NN regressor, and α_{NN} represents the L2 regularisation coefficient of the NN regressor. The optimal hyper-parameter is chosen from the parameter space $\Omega_{N_r \times N_h \times ActFun \times \alpha_{NN}} = \{2,3,4,5,6,7\} \times \{4,6,8,10,12\} \times \{tanh, relu\} \times \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. The dimensionality reduction error ϵ_{mr} is defined as the RMSE between $\hat{\mathbf{g}}^{-1}(\mathbf{g}(\mathscr{U}_{test}))$ and $\mathscr{U}_{test}$ and the prediction error ϵ_{all} is defined as the RMSE between $\hat{\mathbf{g}}^{-1}(\mathscr{M}(\mu_{test}))$ and $\mathscr{U}_{test}$, where μ_{test} is the set of test parameters and $\mathscr{U}_{test}$ is the corresponding temperature profiles obtained by the numerical solver. The comparisons of the temperature profiles between the surrogate model predictions and numerical solutions for 4 randomly-selected test cases are shown in Figure 4.6, including the results given by POD-NN, ICA-NN, and AE-NN. As can be seen, the temperature profiles given by POD-NN match perfectly with the ones given by the numerical solver. Therefore, POD-NN is used in the following for the surrogate modelling of the distributed systems.



(a) $(S, T_L, T_R) = (-0.641, 1.07, -3.46)$

(b) $(S, T_L, T_R) = (3.05, -1.43, -1.82)$

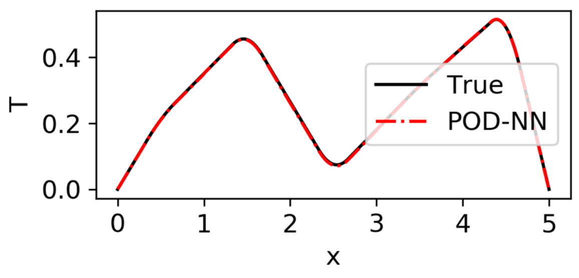
(c) $(S, T_L, T_R) = (2.66, -3.82, -2.46)$

(d) $(S, T_L, T_R) = (-0.697, -4.81, 3.58)$

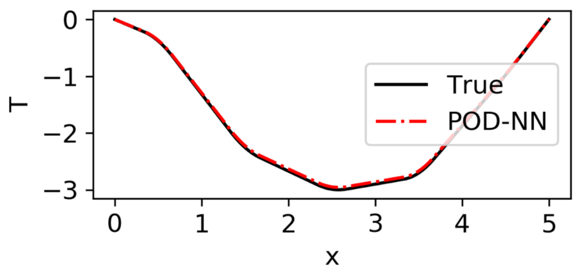
Figure 4.6: The prediction results of a subdomain with a single heat source.

For the prediction of the temperature profile of a whole domain with distributed heat sources, the initialisation and updating of the temperature at the subdomain boundary need to be specified. Here the temperatures are initialised randomly between T_0 and T_l for the internal subdomain interfaces and the boundary conditions T_0 and T_l are imposed for the most left and the most right subdomains. Then the updating process aims at matching T and $\partial T/\partial x$ at the subdomain interface, as it can be derived that the temperature and the first-order derivative of the temperature need to be continuous at the interface.

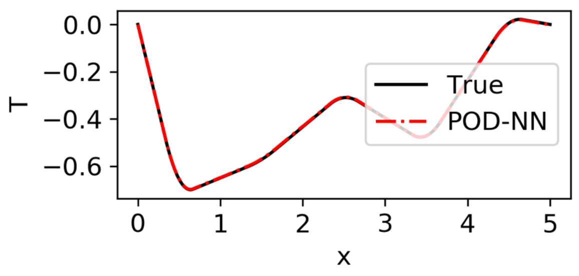
The results for the case of the domain length $l = 5$ with $K = 5$ heat sources are given here to illustrate the surrogate model's scalability. 100 samples of the distributed parameters and the corresponding numerical solutions are generated in order to assess the accuracy of the surrogate model. The comparisons of the temperature profiles between the surrogate model predictions and numerical solutions for 4 randomly-selected test cases are shown in Figure 4.7. The RMSE of the temperature profiles predicted by the surrogate model compared to the high-fidelity numerical solutions averaged over all the test samples is 3.77×10^{-2}. The results clearly demonstrate that the proposed surrogate model (i.e. POD-NN in this case) predicts the distributed heat transfer problem efficiently and accurately.



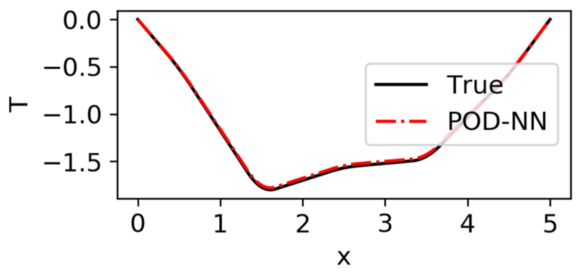
(a) $S_1, S_2, S_3, S_4, S_5 = 0.508, 2.08, -2.09, 0.108, 3.93$



(b) $S_1, S_2, S_3, S_4, S_5 = 3.96, -3.74, -2.93, -4.49, -0.592$



(c) $S_1, S_2, S_3, S_4, S_5 = -4.70, -0.432, 1.49, -2.22, 1.76$



(d) $S_1, S_2, S_3, S_4, S_5 = 0.909, -4.76, 0.589, -2.41, -0.849$

Figure 4.7: The prediction results of a domain with 5 uniformly-distributed heat sources.

4.3.2 Surrogate Modelling of Wind Farms

Wind Farm Model

The high-fidelity flow field data are needed for the surrogate modelling of wind farms. In this work, the LES flow solver SOWFA [160] is employed for CFD data generation. For wind farm simulations, a precursor simulation of neutral atmospheric boundary layer is first carried out to obtain the initial flow field and inflow boundary conditions, then wind farms are simulated using the ALM. The simulation domain considered in this work is illustrated in Figure 4.8 where a typical instantaneous flow field visualisation is also shown. The corresponding hub-height horizontal plane is extracted and shown in Figure 4.9. The size of the simulation domain is $3000 \times 3000 \times 1000$m, with the inflow wind coming from southwest direction. For the mesh generation, a two-level local mesh refinement is used, as is suggested in [35]. The outer mesh dimension is $12 \times 12 \times 12$m, the inner mesh dimension is $3 \times 3 \times 3$m, and the dimension of the mesh in-between is $6 \times 6 \times 6$m. The total number of cells is 1.8×10^7. Three NREL 5-MW baseline turbines are positioned in the simulation domain with a 5 rotor diameter spacing in the downstream direction. The rotor diameter of this baseline turbine (denoted as D hereafter) is 126.4m. For each simulation case, 1500-second simulations are carried out with a time step of 0.02s. Each case requires around 44 hours using 256 processors in the local HPC clusters. After LES simulations, the mean velocity field is obtained by averaging the instantaneous flow field from 400s to 1400s. The surrogate modelling in the following part aims at predicting the 2D mean velocity field at turbine hub height efficiently given the inflow conditions and the yaw angles of all the turbines $[\gamma_1, \gamma_2, ..., \gamma_K]$ for a wind farm consisting of K turbines.

Results

First, the surrogate modelling of a subdomain containing a single turbine is carried out. For the high-fidelity data generation, the wind farm simulation is carried out for the case with 3 turbines in a row, as is illustrated in Figure 4.9. Three inflow wind velocities (i.e., 8m/s, 9m/s, and 10m/s) are considered with a FSTI of 6%. For each wind velocity, 30 samples of turbine yaw angles $[\gamma_0, \gamma_1, ..., \gamma_{30}]$ in the parameter space $\Omega = [-30.0°, 30.0°] \times [-30.0°, 30.0°] \times [-30.0°, 30.0°]$ are generated using Latin hypercube sampling, where $\gamma_i = [\gamma_i^1, \gamma_i^2, \gamma_i^3]$ with γ_i^1, γ_i^2 and γ_i^3 representing the yaw angles of the 1^{st}, 2^{nd}, and 3^{rd} turbines of the i^{th} sample. Then SOWFA is used for generating the flow field for each yaw setting. In total, 90 simulations are carried out. The flow field of each subdomain containing one turbine is then extracted as

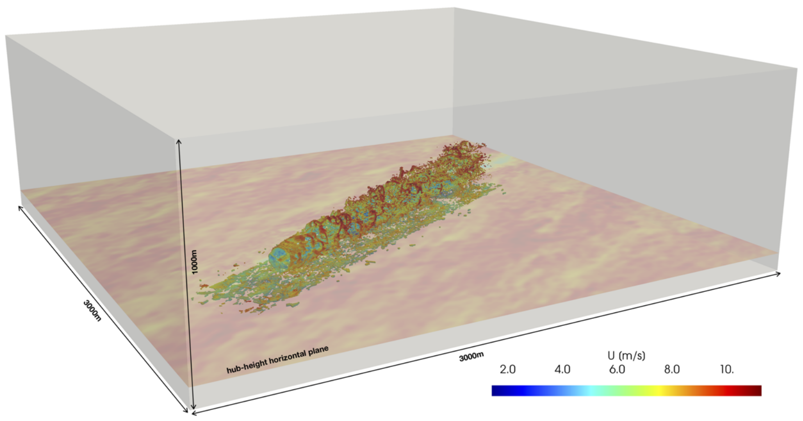


Figure 4.8: The illustration of the 3D simulation domain. A typical instantaneous vorticity contour coloured by velocity magnitude is shown.

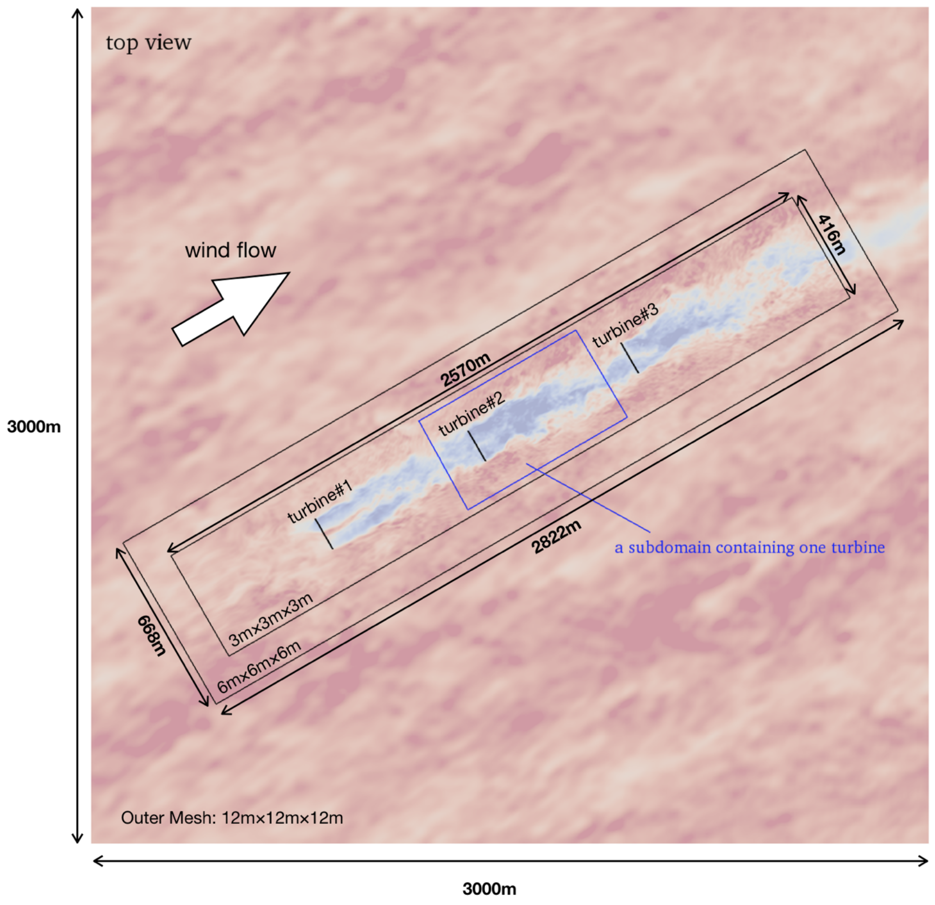


Figure 4.9: A top view of the simulation domain at turbine hub height. The contour shows the instantaneous flow velocity magnitude.

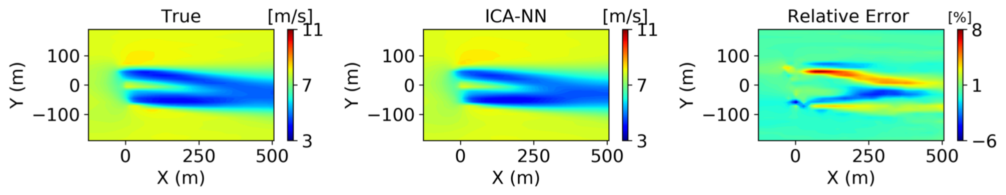
Method	N_r	N_h	Activation	α_{NN}	ϵ_{mr}	ϵ_{all}
POD-NN	15	50	relu	10^{-5}	5.35×10^{-2}	1.67×10^{-1}
ICA-NN	20	50	tanh	10^{-1}	4.45×10^{-2}	1.29×10^{-1}
AE-NN	15	50	relu	10^{-4}	7.74×10^{-2}	1.64×10^{-1}

Table 4.2: The optimal hyper-parameters of the three surrogate modelling methods for the wind farm case.

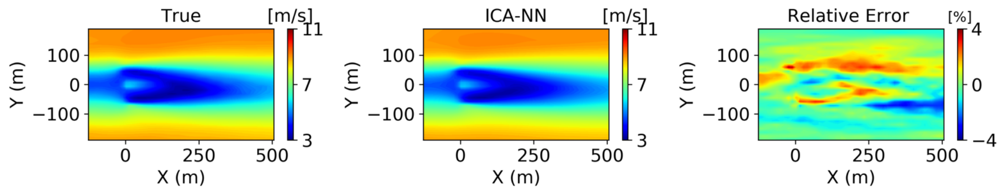
a single data sample. One such subdomain is shown in Figure 4.9. Therefore, 270 data samples are available for the surrogate modelling of the flow field around one turbine. The data samples are then split into two parts, 216 training and validation samples and 54 test samples.

After completing offline data generation, three surrogate modelling methods, i.e. POD-NN, ICA-NN, AE-NN, are employed to build surrogate models to predict the flow field in a single subdomain with the turbine yaw angle and the wind speeds at 30 uniformly-distributed discrete points along the inflow boundary as the model input. The yaw angles and the inflow velocity profiles are scaled to zero mean and unit variance separately before feeding into the NN for training. The learning rate is set as 10^{-3}. A grid-search procedure is carried out to determine the optimal hyper-parameters in the dimensionality reduction and regression models, based on 4-fold cross-validation errors. The optimal hyper-parameters, the dimensionality reduction errors, and the prediction errors are given in Table 4.2 for all three methods, where the optimal hyper-parameter is chosen from the parameter space $\Omega_{N_r \times N_h \times ActFun \times \alpha_{NN}} = \{5, 10, 15, 20, 25, 30\} \times \{10, 20, 30, 40, 50\} \times \{tanh, relu\} \times \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. The dimensionality reduction error ϵ_{mr} is defined as the RMSE between $\hat{\mathbf{g}}^{-1}(\mathbf{g}(\mathscr{U}_{test}))$ and $\mathscr{U}_{test}$ and the prediction error ϵ_{all} is defined as the RMSE between $\hat{\mathbf{g}}^{-1}(\mathscr{M}(\mu_{test}))$ and $\mathscr{U}_{test}$, where μ_{test} is the set of test inflow conditions and yaw angles and $\mathscr{U}_{test}$ is the corresponding 2D velocity fields obtained by SOWFA.

After NN training, the predictions of single turbine wakes are carried out and the results are compared with the SOWFA results for all the 54 test cases. Two typical test cases, the first one with the turbine operating in freestream inflow and the second one with the turbine operating in upstream wake, are shown in Figure 4.10, 4.11 and 4.12, where the flow fields are predicted by ICA-NN, POD-NN and AE-NN respectively. The relative error is defined as the absolute error of the surrogate model prediction divided by the inflow mean velocity. As can be seen, the overall flow fields predicted by all three methods match with the SOWFA results quite well. As the turbine wake evolves from upstream to downstream, the

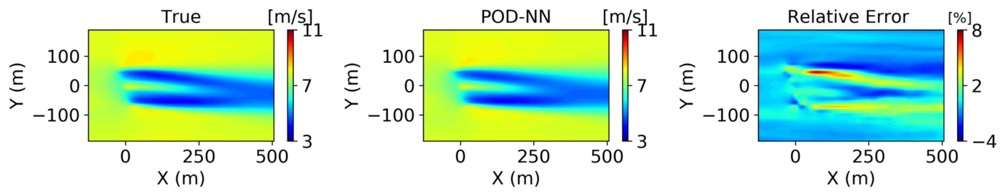


(a) $\gamma = 16.5°$, freestream inflow

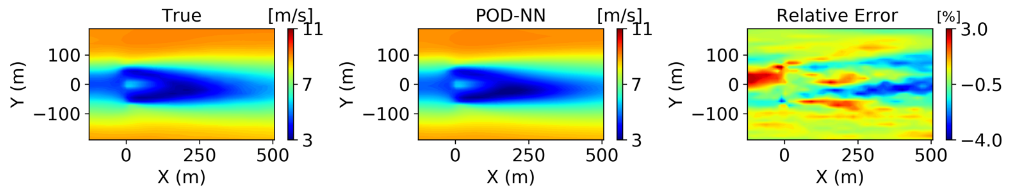


(b) $\gamma = 11.6°$, upstream wake

Figure 4.10: The comparisons between ICA-NN predictions and high-fidelity simulations.

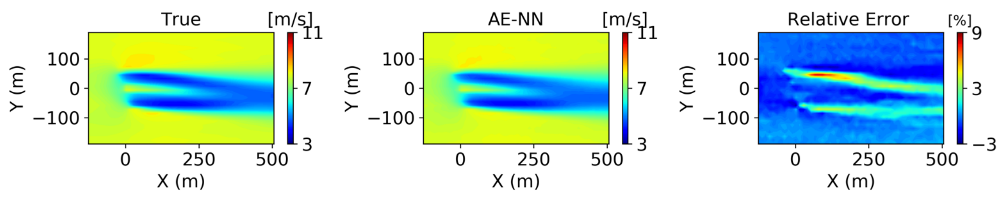


(a) $\gamma = 16.5°$, freestream inflow

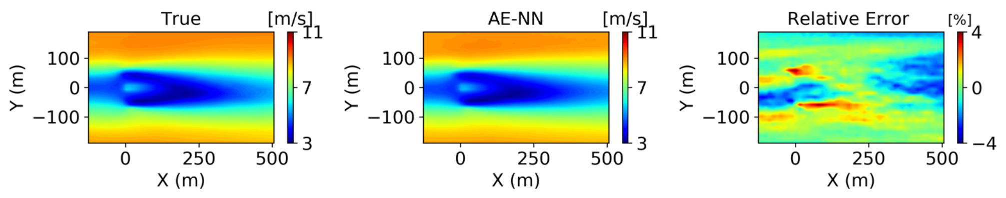


(b) $\gamma = 11.6°$, upstream wake

Figure 4.11: The comparisons between POD-NN predictions and high-fidelity simulations.



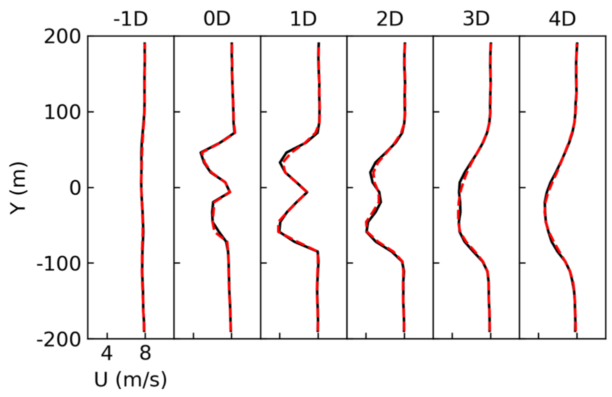
(a) $\gamma = 16.5°$, freestream inflow



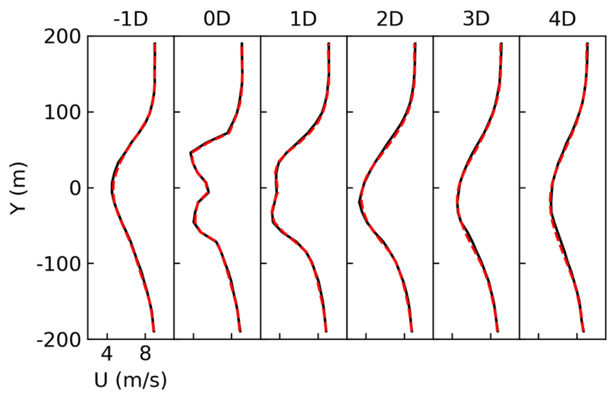
(b) $\gamma = 11.6°$, upstream wake

Figure 4.12: The comparisons between AE-NN predictions and high-fidelity simulations.

wake profiles at different streamwise locations, from one rotor diameter ($X = -1D$) in front of the turbine to four rotor diameter ($X = 4D$) behind the turbine, are investigated in order to further examine the prediction performance. As shown in Figure 4.13(a), a freestream inflow at $X = -1D$ travels downstream and hits the turbine rotor at $X = 0D$. The generated wake then travels from $X = 1D$ to $X = 4D$ in the deflected direction caused by the yaw effects, while slowly recovering toward the freestream conditions and reaching a Gaussian-shape profile at $X = 4D$. At the same time, the wake expands in the spanwise direction as the wake deficit decreases. As for the case shown in Figure 4.13(b), the inflow at $X = -1D$ is of Gaussian-shape which is generated by the upstream turbines, and the wake development from $X = 0D$ to $X = 4D$ shows similar features as in the freestream case. In summary, the velocity profile predictions match perfectly with the high-fidelity simulation results at all streamwise locations, and all the main flow features such as the wake recovery, the wake deflection and the wake expansion are captured very well by the surrogate model under various turbine yaw and inflow conditions. As ICA-NN is the best among all three methods in terms of prediction error as shown in Table 4.2, it is used in the following for the surrogate modelling of wind turbine arrays. It is worth mentioning that for the 1D Poisson case, POD is the best among all three dimensionality reduction techniques in terms of prediction errors, while in this case ICA is the best choice. In this work, the choice of the dimensionality reduction technique is treated similarly as the tuning of the hyper-parameters, i.e. the technique with the smallest RMSE is chosen for the predictions of the flow field



(a) $\gamma = 16.5°$, freestream inflow



(b) $\gamma = 11.6°$, upstream wake

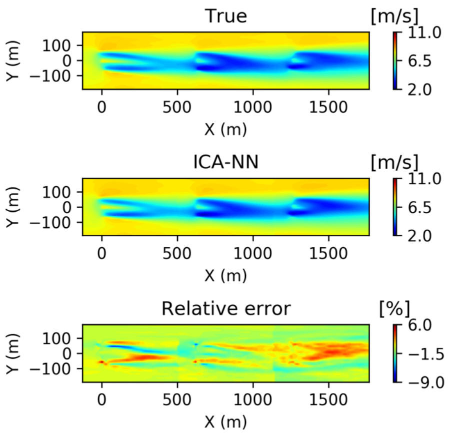
Figure 4.13: The velocity profiles for the 1-turbine cases, by ICA-NN (dashed) and high-fidelity simulations (solid).

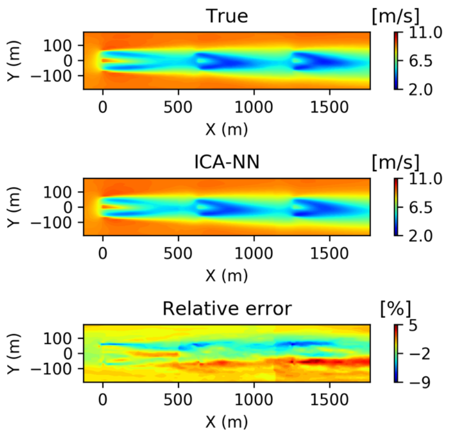
around distributed structures.

The prediction of the flow field around distributed wind turbines follows from Algorithm 3. As the problem here is convection-dominant, the iterative updating process in Algorithm 3 is ignored and the flow quantities at the subdomain boundary are directly imposed according to the upstream surrogate model predictions. Specifically, the turbines' relative positions are first determined according to the wind direction and turbine coordinates. Then the wind fields around the front turbines are predicted by surrogate model with their yaw angles and the freestream inflow profile as the model input. Next, the flow fields around subsequent turbines are predicted similarly, but with the inflow profile extracted from the surrogate model prediction of the corresponding upstream subdomains. In this way, the flow field in the whole wind farm can be obtained.

The results of a wind farm with 3 turbines in a row are given here to test the surrogate model's accuracy, efficiency, and scalability. 20 samples of turbine yaw angles are generated and SOWFA is employed to generate the high-fidelity test data in order to assess the accuracy of the surrogate model. The surrogate model predictions and the SOWFA results are only based on the same distributed yaw angles and the same freestream inflow profile. The results of 3 typical test cases are given in Figure 4.14 where the mean freestream wind speeds are 8m/s, 9m/s and 10m/s respectively. As shown, the prediction errors are quite small in the whole domain, where the maximum value of the relative errors is within 10% for all the test cases. To further examine the surrogate model's performance, the corresponding velocity profiles at various locations from $X = -1D$ to $X = 14D$ are given in Figure 4.15. As can be seen, the wake deficits generated by the turbines at $X = 0D$, $X = 5D$ and $X = 10D$, the wake development including deflection, recovery and expansion behind each turbine, and the wake interactions between turbines are all successfully predicted. The predicted profiles at all the locations match very well with the corresponding high-fidelity simulation results. Also, the average RMSE of the surrogate model predictions compared to SOWFA results for all 20 test cases is 1.60×10^{-1} m/s, which is just 2% of the freestream wind speed, and the online run time of the surrogate model is negligible (around 9×10^{-4}s using one core) compared to SOWFA which requires around 44 hours using 256 cores for each case. Therefore, it is concluded that the surrogate model predicts the wind farm flow field efficiently and accurately.

To further illustrate the scalability of the surrogate model, the prediction of a wind farm with 5×5 wind turbines is carried out and the results are given in Figure 4.16. The inflow velocity is set as 9m/s and the turbine yaw angles from left to right are set as [28.4°, 13.3°, -7.63°, -8.50°, -28.1°] for the 1st row, [27.9°, 8.50°, 21.4°, -29.4°, 25.1°] for the 2nd row, [19.5°,16.5°,-26.5°,-9.15°, -10.8°] for the 3rd row, [-15.7°, 6.54°, -12.1°, -14.6°, -16.9°] for the 4th row, and [16.7°, -2.79°, -27.9°, 10.8°, 26.9°] for the 5th row. The lateral distance between wind turbines is three rotor diameters and the downstream distance is five rotor diameters. As can be seen, the wake interactions and the yaw effects are both captured satisfactorily by the surrogate model. Because the so-constructed surrogate model can achieve fast approximation of the original high-fidelity LES model, it can be used directly for the control design of large-scale wind farms.


(a) 8m/s. $\gamma = (16.5°, 4.69°, -22.9°)$


(b) 9m/s. $\gamma = (2.56°, 15.7°, 4.11°)$

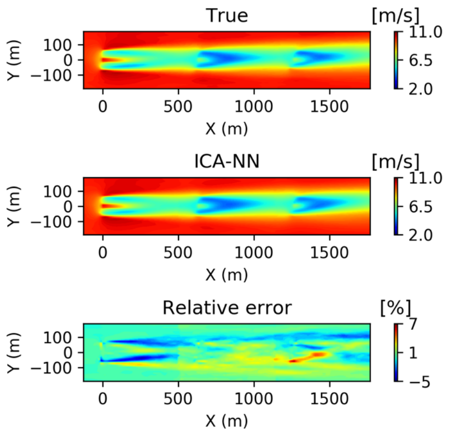
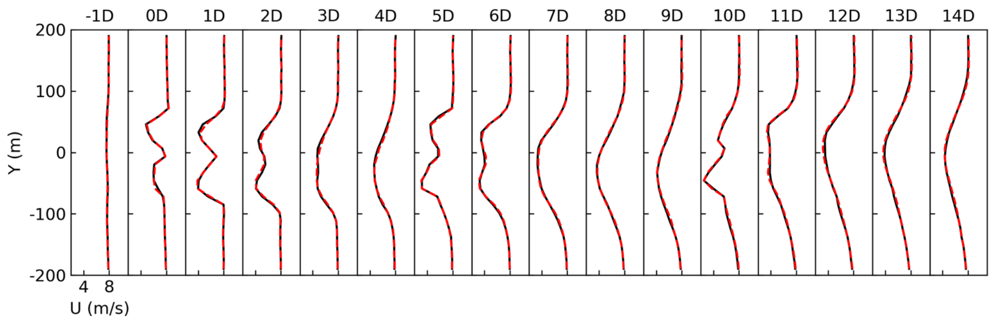
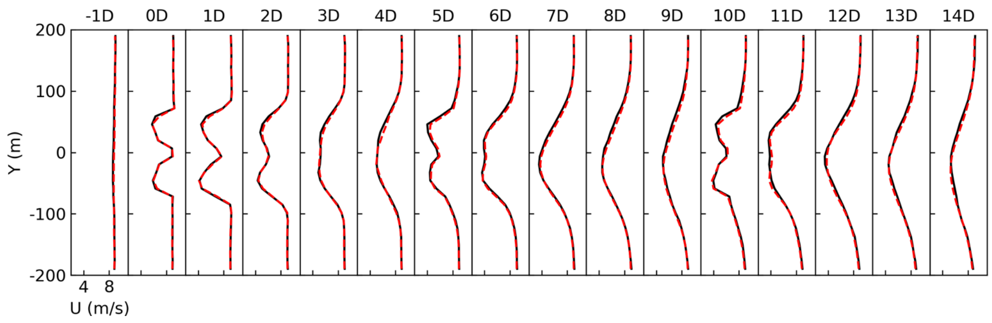

(c) 10m/s. $\gamma = (-6.98°, -15.4°, -23.4°)$

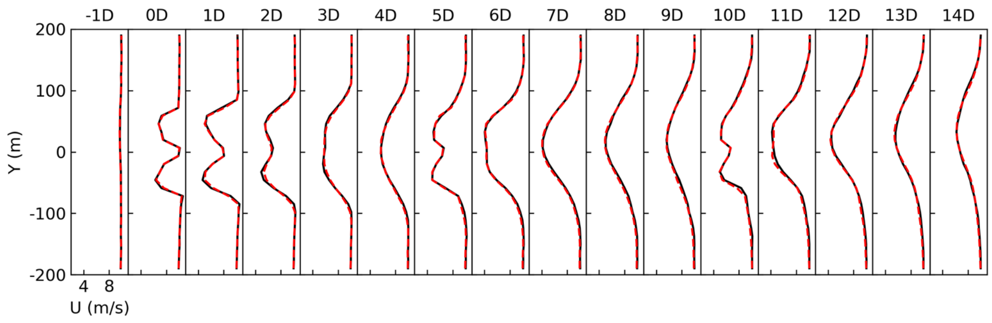
Figure 4.14: The comparisons between ICA-NN predictions and high-fidelity simulations for 3-turbine cases.



(a) 8m/s. $\gamma = (16.5^{\circ}, 4.69^{\circ}, -22.9^{\circ})$



(b) 9m/s. $\gamma = (2.56^{\circ}, 15.7^{\circ}, 4.11^{\circ})$



(c) 10m/s. $\gamma = (-6.98^{\circ}, -15.4^{\circ}, -23.4^{\circ})$

Figure 4.15: The velocity profiles for the 3-turbine cases, by ICA-NN (dashed) and high-fidelity simulations (solid).

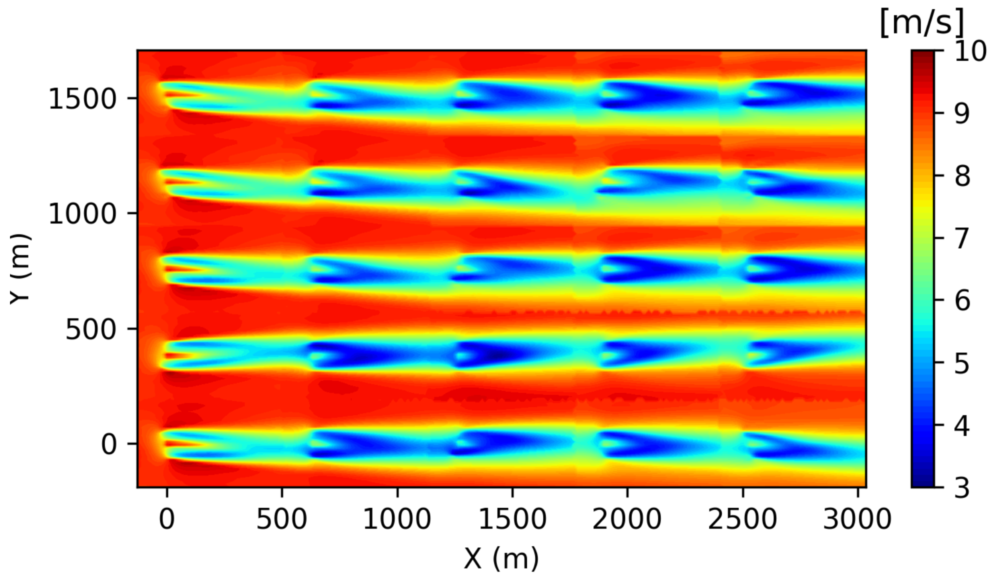


Figure 4.16: The wake prediction of a wind farm with 5×5 wind turbines.

4.4 Conclusions

In this chapter, an ML-based surrogate modelling method was proposed, where three dimensionality reduction techniques (i.e. POD, ICA, AE) were investigated for reducing the flow field dimension and the fully-connected NN was used to predict the reduced coefficients from the input parameters. The surrogate modelling method was specifically designed to tackle distributed fluid systems, by carrying out surrogate modelling for each subdomain and combining the flow field of each subdomain with the consideration of the matching condition at the interface. The applications to a diffusion-dominant problem (more specifically, 1D Poisson's equation) and a convection-dominant problem (more specifically, wind farm simulations) demonstrated the efficiency, accuracy, and scalability of the proposed surrogate modelling method. In particular, the surrogate model of wind farm wakes predicted the wind farm velocity field very accurately with an average RMSE (compared to high-fidelity results) being 2% of the freestream wind speed, while the online prediction time is negligible (around 9×10^{-4}s using one core) compared to high-fidelity simulation which requires around 44 hours using 256 cores for each case. The wake features including wake deflection, recovery, and expansion behind each wind turbine, and the wake interactions through the upstream interface, have been captured well. This demonstrated the ability of the proposed method in modelling large-scale distributed fluid systems. However, the developed data-based wind farm wake model has only considered the upstream flow's impact on the downstream turbines, ignoring the interactions in the lateral and upstream directions. As shown in Figure 4.16, there exist lateral discontinuities in the predicted wind farm flow field. This issue may be solved by considering all the subdomain boundary conditions as the input of the machine learning model (similarly as in the Poisson case). However, this task is not trivial and needs further investigations regarding the method design and the need for more training data.

Future research may involve the application of the proposed surrogate modelling to other flow problems, such as the flow field around heater arrays in heat exchangers and the distributed roughness elements in boundary layer flow. The optimal design using the constructed surrogate model is also of great interest. For example, it can be used as an internal model for wind farm yaw optimisation. Another research direction is the surrogate modelling of unsteady distributed systems and the optimal control design based on such surrogate models.

Chapter 5

Wind Farm Wake Modelling Based on Deep Convolutional Conditional Generative Adversarial Network

5.1 Introdution

In this chapter, to avoid the explicit dimensionality reduction errors and meanwhile mitigate the overfitting issues in predicting high-dimensional target in supervised manner, a novel surrogate modelling method is proposed based on one of the most recent advancements in the field of deep learning i.e. the GAN framework [154]. It is then applied to build an efficient, accurate, and robust data-based wind farm wake model.

Specifically, the surrogate modelling method proposed in this chapter follows the DC-CGAN, which can take advantage of both the deep convolutional network's ability in image processing and the CGAN's ability in generating high-dimensional content according to specific labels/images. In order to build a robust and flexible surrogate model for fluid systems, (i) the conditional GAN framework instead of the original GAN is employed so that the generator will generate the 'realistic' flow field according to the corresponding flow parameters; (ii) the flow parameters are embedded through a fully-connected embedding layer before concatenated with the flow field and fed to the discriminator. In this way, the flow parameters from various sources can be accommodated such as the parameters defining the governing equations and boundary conditions; (iii) the noise prior and the batch normalisation

[178] that are usually present in GAN models are excluded, as the considered fluid systems are deterministic and the surrogate model should not rely on the batch information such as the batch mean and standard deviation.

The proposed surrogate modelling method is then applied to wind farm wake modelling. Specifically, a CFD database of wind turbine wakes is first generated [179]. Then the CGAN-based wake model is trained to take the inflow wind profiles and the yaw conditions as the model input and to predict the multi-channel flow field (i.e. both streamwise and spanwise velocity fields) as the model output. After training, the prediction results are first validated against high-fidelity simulations. A comprehensive parametric study is then carried out to evaluate the generalisation performance of the developed model to the flow scenarios that are not present in the training dataset.

To further demonstrate the use of the developed CGAN-based wake model in wind farm applications, a case study for a small wind farm is carried out. For comparison purpose, the corresponding high-fidelity simulations are also carried out using the LES flow solver SOWFA [160].

This chapter is organised as follows: the proposed deep learning based surrogate modelling method is described in Section 5.2, where the NN structure and generative adversarial training process are explained in detail. It is then applied to the development of a novel wind farm wake model in Section 5.3, where the validation and generalisation performance of the developed model are evaluated first, then a case study for a small wind farm is presented including the results from both the developed model and the high-fidelity LES model. Finally the conclusions are drawn in Section 5.4.

5.2 DC-CGAN Based Surrogate Modelling Method

A general steady-state parametrised fluid system can be described by

$$\begin{aligned} &\mathscr{P}_{\mu_p}[u] = 0, x \in \Omega_{\mu_\omega}; \\ &\mathscr{B}_{\mu_b}[u] = 0, x \in \partial\Omega_{\mu_\omega} \end{aligned} \tag{5.1}$$

where u is the state of the system while the differential operator $\mathscr{P}$ (parametrised by μ_p), the differential operator $\mathscr{B}$ (parametrised by μ_b) and Ω (parametrised by μ_ω) represent the PDEs describing the fluid systems, the boundary conditions and the flow domain respectively. Hereby the flow parameters arising from the governing equations, the domain geometry and the boundary conditions are denoted as $\mu =$

$[\mu_p, \mu_\omega, \mu_b]$. Given a specific value of μ, the flow field in the domain Ω, hereby denoted as U, can be obtained by solving Equation 5.1 numerically. However, this usually requires a lot of computational resources and is time-consuming, as the DoF of the discretised PDEs is usually very high. This section is devoted to developing a surrogate modelling method to approximate the mapping between μ and U so that fast and accurate predictions of U can be achieved. The proposed method is based on the state-of-the-art deep learning technique DC-CGAN. It is illustrated in Figure 5.1, including the generation of training data in Figure 5.1(A), the GAN structure and training in Figure 5.1(B), and the online prediction in Figure 5.1(C). They are detailed as below.

5.2.1 Training Dataset

The surrogate model is trained based on a set of samples where each sample consists of the flow parameter μ and the corresponding flow field U, as shown in Figure 5.1(A). In order to generate the training dataset, a sampling method is usually employed to generate a set of flow parameters $[\mu^1, \mu^2, ..., \mu^N]$ where N represents the sample size and μ^i represents the flow parameters arising from the governing equation, the domain geometry and the boundary conditions (i.e. $\mu^i = [\mu_p^i, \mu_\omega^i, \mu_b^i]$). Then a set of CFD simulations are carried out for each flow parameter μ^i so that the corresponding flow field U^i can be obtained, as shown in Figure 5.1 (A). After data generation, all the flow parameters are collected as the training input matrix $\mathscr{X}$ of shape $[N, N_\mu]$ where N_μ is the dimension of the flow parameter and each row contains a sample of the flow parameter. All the corresponding high-fidelity flow field data are collected as the training target matrix $\mathscr{Y}$ of shape $[N, N_1, ..., N_d, C]$ where $[N_1, ..., N_d]$ is the spatial resolution of the d-dimension flow domain, C is the number of the channels of the flow field data, and each row contains a sample of the flow field in the d-dimension domain with C channels. Each channel usually represents a colour such as red, green or blue of an RBG image in image processing while it is used to represent a flow quantity such as streamwise velocity or spanwise velocity in this work. It is worth noting that the dimensions of both the training input μ and the training target U are typically very high as the former can include the boundary conditions (such as the inflow velocity) at discrete points while the latter can include multi-channel flow field on a grid of high dimension. It is this high-dimensionality that makes the surrogate modelling of such systems very challenging.

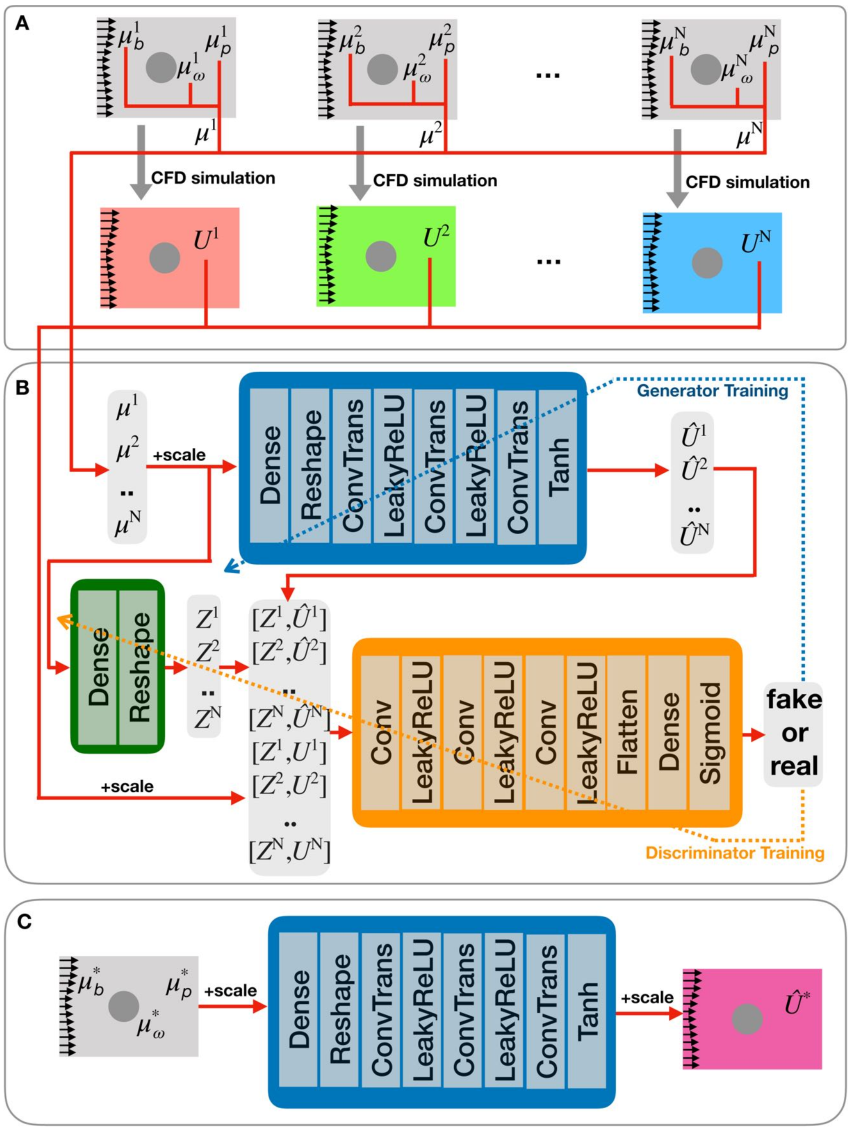

A
μ_b^1 μ_p^1 μ_ω^1 μ^1
μ_b^2 μ_p^2 μ_ω^2 μ^2
μ_b^N μ_p^N μ_ω^N μ^N
CFD simulation
U^1 U^2 U^N
B
μ^1 μ^2 μ^N
+scale
Dense Reshape ConvTrans LeakyReLU ConvTrans LeakyReLU ConvTrans Tanh
Generator Training
$\hat{U}^1$ $\hat{U}^2$ $\hat{U}^N$
Dense Reshape
Z^1 Z^2 Z^N
$[Z^1,\hat{U}^1]$ $[Z^2,\hat{U}^2]$ $[Z^N,\hat{U}^N]$ $[Z^1,U^1]$ $[Z^2,U^2]$ $[Z^N,U^N]$
Conv LeakyReLU Conv LeakyReLU Conv LeakyReLU Flatten Dense Sigmoid
fake or real
Discriminator Training
C
μ_b^* μ_p^* μ_ω^*
+scale
Dense Reshape ConvTrans LeakyReLU ConvTrans LeakyReLU ConvTrans Tanh
+scale
$\hat{U}^*$

Figure 5.1: The flowchart illustrating the proposed DC-CGAN based surrogate modelling method. The dashed lines illustrate the directions of the backpropagation in the generator and discriminator trainings.

5.2.2 GAN Structure

After obtaining the training dataset, the DC-CGAN based surrogate model is constructed, which is illustrated in Figure 5.1 (B). It consists of a generator and a discriminator. The generator, as shown in shaded blue in Figure 5.1 (B), takes the flow parameters as the input, processes it through a dense layer and a reshape layer which are then followed by a series of transposed convolution layers, and finally returns the flow field prediction $\hat{U}$ as the output. The input-output mapping of the generator, denoted as $\mathscr{G}$, can be expressed as

$$\mathscr{G}(\mu) = (\sigma \circ \mathscr{T}_g)^{L_g} \circ \mathscr{R} \circ \mathscr{N}_g(\mu) \tag{5.2}$$

where $\circ$ represents the function composition, L_g represents the number of transposed convolution layers, and $\mathscr{N}$, $\mathscr{R}$, $\mathscr{T}$ and σ represent the mappings of the dense layer, the reshape layer, the transposed convolution layer, and the activation respectively. In this work, as shown in Figure 5.1 (B), LeakyReLU is used for the activations in the intermediate layers and the hyperbolic tangent function is used for the last layer. The subscript g in Equation 5.2 indicates the corresponding mappings rely on the trainable weights which will be updated during the generator training.

The discriminator, as shown in shaded orange in Figure 5.1 (B), takes the data pair of the embedded flow parameter Z and the corresponding flow field U or $\hat{U}$ (real or generated) as the input, processes it through a series of convolution layers, and finally returns a single classification indicator (i.e. fake or real) as the output. The main difference between CGAN and GAN is that the labels (here the flow parameters) are combined with the corresponding flow field for the examination by the discriminator, while GAN only distinguishes the generated flow field from the real flow field without the labelling information. Therefore, the CGAN structure is more suitable for the surrogate modelling of parametrised fluid flows, as the unique correspondence between the flow parameter and the flow field can be established. Here in the proposed discriminator structure, an embedding layer which consists of a dense and a reshape layer (as shown in shaded green in Figure 5.1 (B)) is employed to map the flow parameters to the same shape as the flow field so that it can be concatenated with the flow field and fed to the discriminator. The input-output mapping of the discriminator including the embedding part can be expressed as

$$\mathscr{D}([\tilde{U}, \mu]) = \sigma \circ \mathscr{N}_d \circ \mathscr{F} \circ (\sigma \circ \mathscr{C}_d)^{L_d}([\tilde{U}, \mathscr{R} \circ \mathscr{N}_e(\mu)]) \tag{5.3}$$

where $\tilde{U}$ can be the real flow field U or the generated flow field $\hat{U}$, L_d represents the number of convolution layers, and $\mathscr{F}$ and $\mathscr{C}$ represent the mappings of the flatten layer and the convolution layer respectively. As shown in Figure 5.1 (B), LeakyReLU is used for the activation in the intermediate layers while the sigmoid function is used for the last layer so that a value between 0 and 1 can be returned as the output for the binary classification. The subscripts d and e in Equation 5.3 indicate the corresponding mappings in the discriminator and embedding part rely on the trainable weights which will be updated during the discriminator training.

5.2.3 GAN Training

The discriminator is trained to distinguish the pair of real flow field and flow parameter from the pair of generated flow field and flow parameter, while the generator is trained to generate realistic flow field such that the generated data pair is not distinguishable from the real data pair. Specifically, the discriminator is trained to minimise the classification error defined as

$$\epsilon_{\mathscr{D}} = \frac{1}{N_b}\sum_{i=1}^{N_b} -\log \mathscr{D}([U^i, \mu^i]) + \frac{1}{N_b}\sum_{i=1}^{N_b} -\log(1 - \mathscr{D}([\hat{U}^i, \mu^i])) \tag{5.4}$$

where $\{[U^i, \mu^i], 1 \leq i \leq N_b\}$ is a batch of training samples consisting of the real flow field and the corresponding flow parameters, and $\{[\hat{U}^i, \mu^i], 1 \leq i \leq N_b\}$ is a batch of training samples consisting of the fake flow field generated by the generator and the corresponding flow parameters. The data batches are fed into the discriminator network to minimise $\epsilon_{\mathscr{D}}$, so that the generated and real data pair can be classified as fake (i.e. 0) and real (i.e. 1) by the discriminator after training. The discriminator training is illustrated by the dashed line coloured in orange in Figure 5.1 (B).

The generator is trained to minimise the classification error defined as

$$\epsilon_{\mathscr{G}} = \frac{1}{N_b}\sum_{i=1}^{N_b} -\log \mathscr{D}([\mathscr{G}(\mu^i), \mu^i]) \tag{5.5}$$

Here $\{\mu^i, 1 \leq i \leq N_b\}$ is a batch of flow parameters which are fed into the generator to generate flow field that is then examined by the discriminator. The minimisation of the classification error $\epsilon_{\mathscr{G}}$ thus guides the generator to produce data pairs which are likely to be classified as real by the discriminator. As can be seen from Equation 5.5, the generator training involves the whole CGAN network including the generator and the discriminator. It is worth noting that the trainable weights within the discriminator network are kept frozen during the training process while the trainable weights inside the generator are trained to minimise $\epsilon_{\mathscr{G}}$. The generator training is illustrated by the dashed line coloured in blue in Figure 5.1 (B).

The discriminator training and the generator training are carried out alternatively until the generator can produce realistic flow field that is not distinguishable from the real flow field obtained by the high-fidelity simulations. All the training data including the training input and the training target are standardised before fed into the NN for training. The Adam optimisation algorithm [158] is used for the NN training and the model is implemented based on the ML package Keras [176] with Tensorflow [155] backend. The training is carried out using NVIDIA Tesla

Algorithm 4 Surrogate modelling of parametrised fluid flows based on DC-CGAN

```
 1: % The offline training
 2: Generate $N$ samples of the flow parameters: $[\mu^1, \mu^2, ..., \mu^N]$.
 3: Run CFD simulations to generate the corresponding flow field data
    $[U^1, U^2, ..., U^N]$.
 4: Preprocess the flow parameters and the flow field data by MinMaxScalers.
 5: Set the batch size $N_b$; Set the total number of training iterations $N_{tr}$.
 6: for j = 1 to $N_{tr}$ do
 7:     Sample a random batch of training data: $\{[U^i, \mu^i], 1 \le i \le N_b\}$.
 8:     Generate the flow field predictions $\{[\hat{U}^i], 1 \le i \le N_b\}$ by propagating
    $\{[\mu^i], 1 \le i \le N_b\}$ through the generator.
 9:     Train the discriminator to minimise $\epsilon_{\mathscr{D}}$ by feeding $\{[U^i, \mu^i], 1 \le i \le N_b\}$ and
    $\{[\hat{U}^i, \mu^i], 1 \le i \le N_b\}$ to the discriminator.
10:     Train the generator to minimise $\epsilon_{\mathscr{G}}$ by feeding $\{\mu^i], 1 \le i \le N_b\}$ to the whole
    network. Only the trainable weights inside the generator are trained while the
    trainable weights in the discriminator (including the embedding part) are kept
    frozen.
11: end for
12: % The online prediction
13: Set the test flow parameter $\mu^*$ and process it through the MinMaxScaler.
14: Predict the scaled flow field by propagating the scaled flow parameter through
    the generator network.
15: Obtain the final flow field prediction by scaling back the generator output.
```

K80 GPU. After training, the generator can be used as the surrogate model of the parametrised fluid systems for the online prediction of the flow field with flow parameters as the model input, see Figure 5.1 (C). The overall process including both the offline training and the online prediction is summarised as Algorithm 4.

5.3 Application to Wind Farm Wake Modelling

This section is devoted to the development of an accurate and efficient data-based model of wind farm wakes, by employing the surrogate modelling method proposed in Section 5.2. In particular, this work focuses on the prediction of the wind velocity field (including the streamwise and spanwise velocity fields) around the wind turbine with the inflow wind profile U_{in} and the turbine yaw angle γ as the model input. The flow domain and the input flow parameter are illustrated in Figure 5.2. In the rest part of this section, the training data, which is generated by high-fidelity CFD simulations, is described in Section 5.3.1. The model training and validation are then presented in detail in Section 5.3.2. Next, a parametric study is carried out in Section 5.3.3 to systematically evaluate the generalisation performance of the

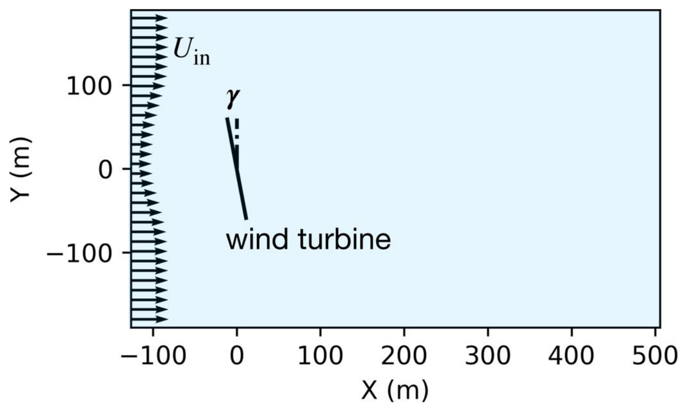


Figure 5.2: The illustration of the flow domain and the input flow parameter including the inflow wind profile U_{in} and the turbine yaw angle γ.

developed CGAN wake model. Finally a case study is investigated in Section 5.3.4 to fully demonstrate the use of the developed model in wind farm applications.

5.3.1 High-Fidelity Data

To generate the training dataset, the high-fidelity LES flow solver SOWFA [160] is employed to solve the filtered NS equations where the turbine rotors are modelled as actuator lines. In this work, in order to capture the wake interactions in the training dataset, the cases of three turbines operating in a row are simulated and then the 2D mean velocity field around each turbine at the turbine hub height is extracted from the simulation data. Thus each simulation can generate three training samples i.e. the flow field around the 1st, the 2nd and the 3rd turbines. In order to cover a wide range of inflow conditions, three groups of LESs are carried out where different freestream mean wind speeds at 8m/s, 9m/s and 10m/s are used for different groups. An illustration of the inflow wind profiles is given in Figure 5.3, where Group#1, Group#2 and Group#3 correspond to the simulation groups with the freestream mean wind speed of 8m/s, 9m/s and 10m/s respectively. As shown in Figure 5.3, for each group, the inflow wind profiles include the 'flat' profiles representing the freestream wind conditions and the 'bell-shape' profiles representing the incoming wind conditions induced by the upstream turbine wakes. Furthermore, in order to capture the yaw effects in the training dataset, 30 simulations have been carried out for each simulation group, where the turbine yaw angles are randomly sampled in the interval $[-30°, 30°]$ for each simulation case. Therefore, in total, 90 LESs have been carried out and 270 training samples are finally generated, which used around one million CPU hours on HPC clusters. This high-fidelity database is used in this

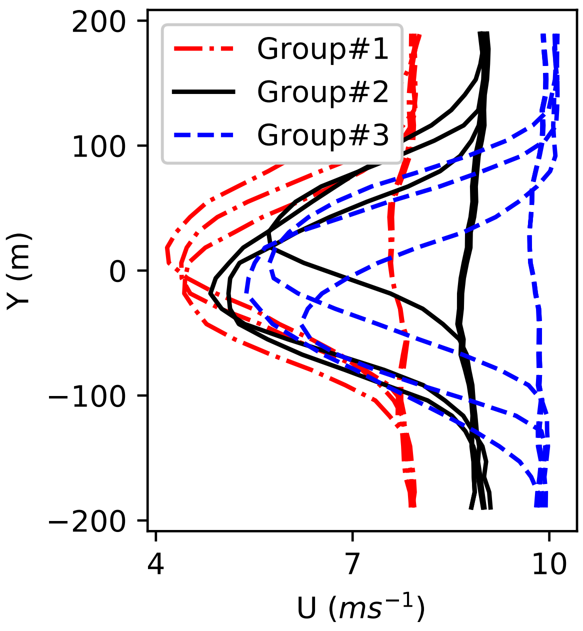


Figure 5.3: The illustration of the inflow wind profiles for selected samples that are representatives of the training dataset.

section to build an accurate and efficient surrogate model to predict the wind flow around wind turbine with the inflow wind profile U_{in} and the turbine yaw angle γ as the model input. The interested reader can refer to [179] for the further details such as the mesh generation, the atmospheric boundary layer simulation, and the parametrisations of the NREL 5MW wind turbine (with a rotor diameter of 126m) considered in this work.

Specifically, the model input μ is specified as the combination of the wind speed at 32 uniformly-distributed points along the inflow boundary and the value of turbine yaw angle. Thus the dimension of μ is 33. The flow field U is specified as the combination of the streamwise velocity field U_x and spanwise velocity field U_y at the 32×32 uniform grid points in the flow domain shown in Figure 5.2. Thus the dimension of U is $32 \times 32 \times 2$. It is worth noting that the considered fluid system is governed by the NS equations where the turbine rotors are modelled as actuator lines in the momentum equations. Thus the turbine parameters such as γ appear in the governing equations. Therefore, this case demonstrates that the proposed surrogate modelling method can accommodate flow parameters arising from different sources (here boundary conditions and governing equations) and can be used for the prediction of multi-channel (here both streamwise and spanwise velocity components) flow fields.

5.3.2 Model Training and Validation

All the high-fidelity CFD data are first divided into 216 training samples and 54 test samples, then the surrogate model is trained based on the flow field data in the training samples. After training, its performance is evaluated by comparing the flow field predictions with the corresponding CFD data in the test samples which are assumed unavailable during the training process. The results for four randomly-selected test cases are shown in Figure 5.4 and 5.5. The flow conditions for these four cases are: $\gamma = -13.6°$ and the upstream wake inflow (Case#1); $\gamma = 16.5°$ and the freestream inflow (Case#2); $\gamma = -26.1°$ and the freestream inflow (Case#3); $\gamma = -1.1°$ and the upstream wake inflow (Case#4). As can be seen, the flow field predictions, including both the streamwise velocity field U_x and the spanwise velocity field U_y, match with the corresponding LES results very well for all the cases. The main features of the flow field are captured accurately, such as the wake deflection with the turbine yaw angle, the wake recovery in the streamwise direction, the wake expansion in the spanwise direction, and the upstream wake's impact on the downstream flow field. The flow details are also predicted such as the fluctuations in the incoming wind, the flow acceleration on both sides of the turbines, the turbines' blockage effects on the upstream flow, and the yaw-induced spanwise velocity in the wake regions.

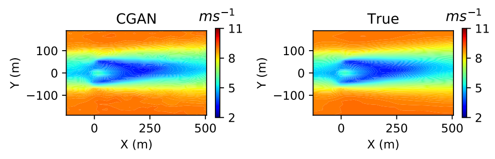
To further quantify the prediction accuracy, the RMSEs of the CGAN model predictions for all the 54 test cases are calculated, which are defined as

$$\epsilon_{U_x} = \sqrt{\frac{1}{N_{test}N_{grd}} \sum_{j=1}^{N_{test}} \sum_{i=1}^{N_{grd}} (U_{xi}^j - \hat{U}_{xi}^j)^2} \tag{5.6}$$

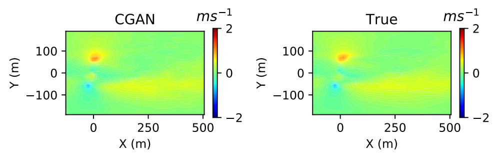
and

$$\epsilon_{U_y} = \sqrt{\frac{1}{N_{test}N_{grd}} \sum_{j=1}^{N_{test}} \sum_{i=1}^{N_{grd}} (U_{yi}^j - \hat{U}_{yi}^j)^2} \tag{5.7}$$

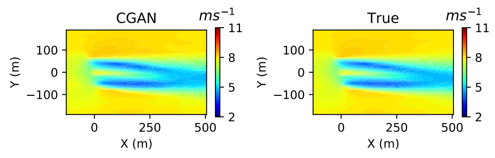
for the streamwise and spanwise velocity field respectively. Here N_{test} is the total number of test cases, N_{grd} is the dimension of the grid, U_{xi}^j and $\hat{U}_{xi}^j$ represent the true and predicted values of the streamwise velocity at the i^{th} grid point for the j^{th} test case, and U_{yi}^j and $\hat{U}_{yi}^j$ represent the true and predicted values of the spanwise velocity at the i^{th} grid point for the j^{th} test case. The RMSEs of the streamwise



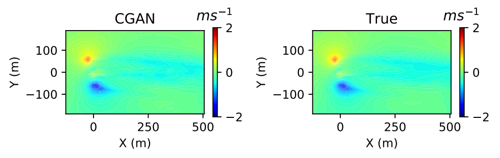
(a) Case#1, U_x



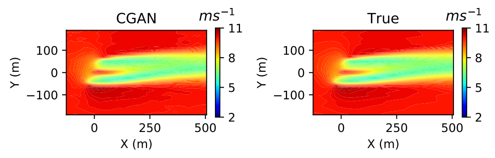
(b) Case#1, U_y



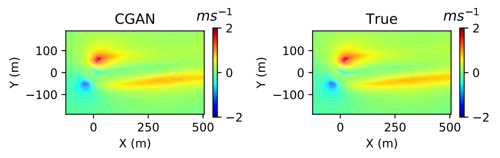
(c) Case#2, U_x



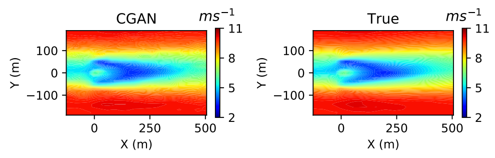
(d) Case#2, U_y



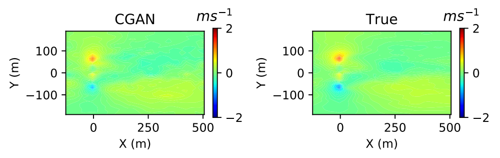
(e) Case#3, U_x



(f) Case#3, U_y

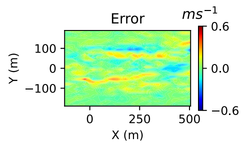


(g) Case#4, U_x

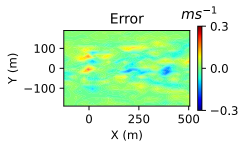


(h) Case#4, U_y

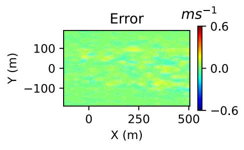
Figure 5.4: The comparisons between the predictions by the developed CGAN model and the corresponding true values for four randomly-selected test cases.



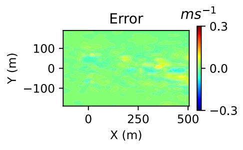
(a) Case#1, U_x



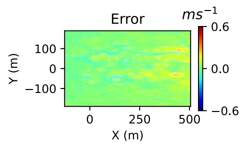
(b) Case#1, U_y



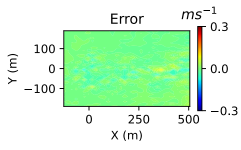
(c) Case#2, U_x



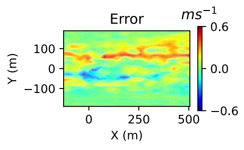
(d) Case#2, U_y



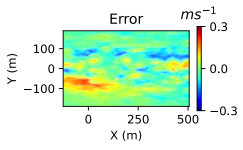
(e) Case#3, U_x



(f) Case#3, U_y



(g) Case#4, U_x



(h) Case#4, U_y

Figure 5.5: The difference between the flow field predicted by the developed CGAN model and the corresponding true values for four randomly-selected test cases.

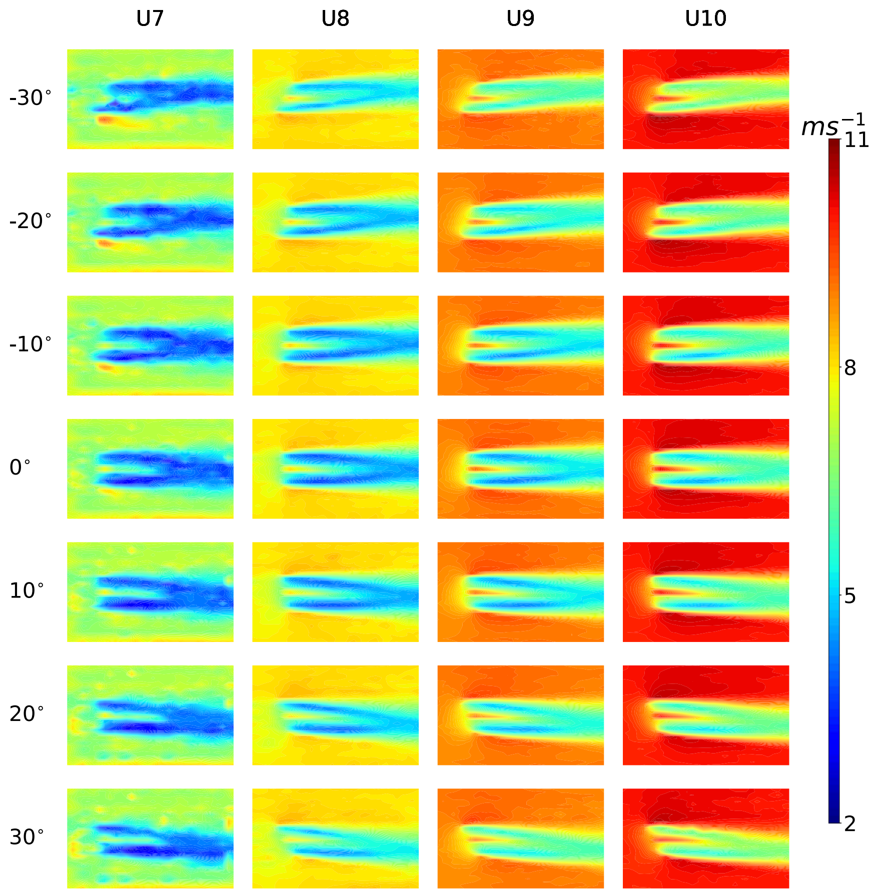


Figure 5.6: The flow fields predicted by the developed CGAN model for a series of turbine yaw angles and freestream wind speeds.

and spanwise velocity field predictions are 0.10m/s and 0.18m/s respectively, which are just 1.1% and 4.1% of the corresponding value ranges. It is worth noting that the wake prediction with such accuracy is achieved with an online prediction time of just 0.002s, which is negligible compared to large-scale numerical simulations which will require several thousand CPU hours. It is concluded that the developed CGAN model is able to predict the wind turbine wake flows efficiently and accurately.

5.3.3 Generalisation Performance

To systematically evaluate the generalisation performance of the developed CGAN model, in this subsection, a series of flow field predictions (including the flow scenarios that are distinct from the training dataset) are carried out. In particular, the parametric study considers a series of turbine yaw angles, freestream wind speeds,

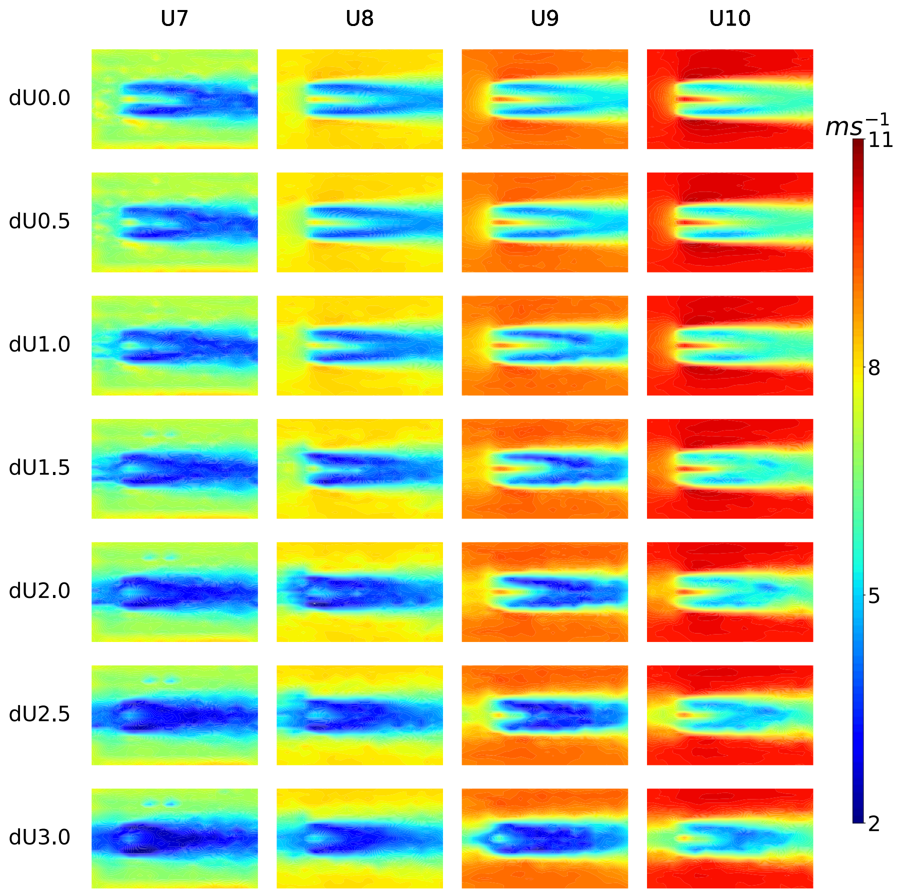


Figure 5.7: The flow fields predicted by the developed CGAN model for a series of upstream wake magnitudes and freestream wind speeds.

and upstream wake magnitudes.

First, a series of turbine yaw angles and freestream wind speeds are considered. For this set of predictions, the inflow wind profile U_{in} is specified as constant values (i.e. the freestream wind speed) along the inflow boundary. The prediction results are given in Figure 5.6. Each column in Figure 5.6 shows the flow field predicted with the same inflow profiles and different yaw angles i.e. [$-30°$, $-20°$, $-10°$, $0°$, $10°$, $20°$, $30°$], while each row shows the flow field predicted with the same yaw angle and different freestream wind speeds i.e. [7, 8, 9, 10] m/s. As can be seen, the trend of wake deflection is captured with the change of the turbine yaw angle, and the qualitative feature of the velocity magnitude in the whole domain is captured with the change of the freestream wind speed. It is worth mentioning that the training dataset, as described in Section 5.3.1, includes only the cases with the freestream wind speed of 8m/s, 9m/s and 10m/s. Thus the prediction of the

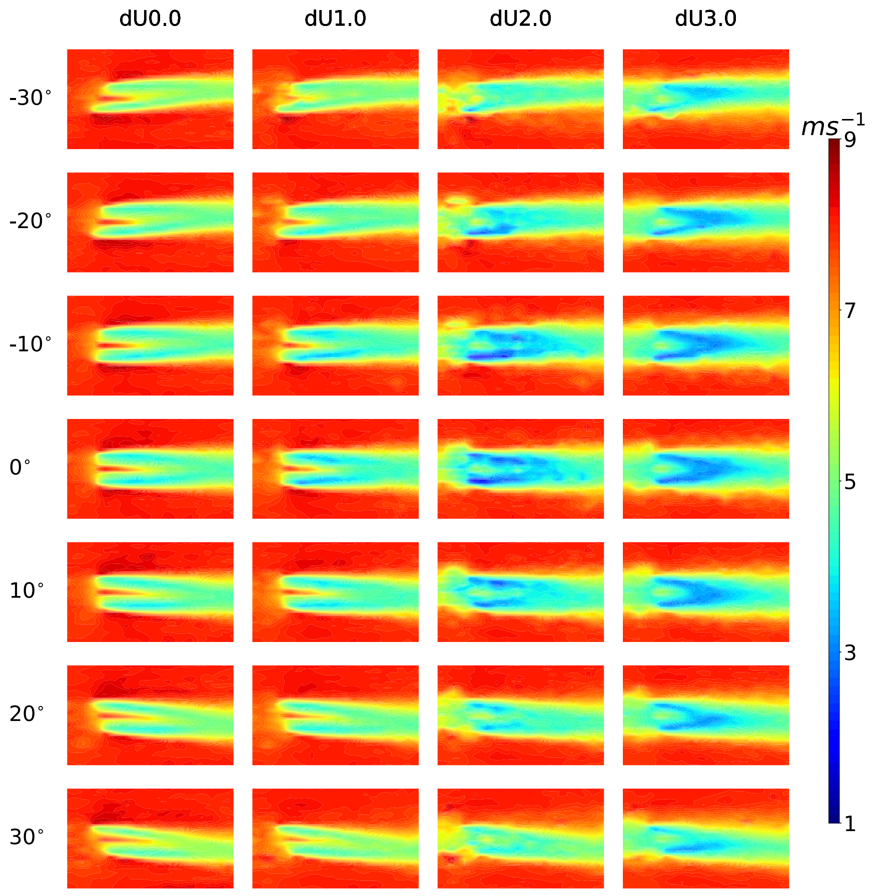


Figure 5.8: The flow fields predicted by the developed CGAN model for a series of turbine yaw angles and upstream wake magnitudes.

7m/s cases, i.e. the first column of Figure 5.6, demonstrates that the CGAN wake model captures the qualitative flow features (such as yaw effects) successfully even for unknown flow scenarios.

Next, a series of upstream wake inflows are considered. Specifically, the Gaussian profile, one of the most popular wake profiles in analytical wake modelling, is used to specify the inflow wind profile U_{in} as

$$U_{in} = U_0 - dU \exp{(-\frac{y^2}{2\sigma_g^2})}, \tag{5.8}$$

where U_0 represents the freestream wind speed, dU represents the magnitude of the upstream wake, y is the spanwise coordinate, and σ_g characterises the wake width. The value of σ_g is set as 50m in the following predictions, which leads to a reasonable wake width. The prediction results are given in Figure 5.7, where each column shows the flow field predicted with the same freestream wind speed U_0 and different upstream wake magnitudes dU i.e. [0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0] m/s, while each row shows the flow field predicted with the same upstream wake magnitude dU and different freestream wind speed U_0 i.e. [7, 8, 9, 10] m/s. As can be seen, the main features of the flow field are captured successfully, where the far field is mainly influenced by the freestream wind speed and the wake regions are clearly influenced by the upstream wake magnitudes. It is worth mentioning that the training dataset only includes the cases where the upstream wake magnitude is around 3-4m/s, while the CGAN wake model is shown to be able to predict the flow field with various upstream wake magnitudes. In addition, the 'bell-shape' inflow profiles in the training dataset are not Gaussian but the results shown in Figure 5.7 are predicted with Gaussian profiles as the model input. This set of predictions thus demonstrate that the developed CGAN model generalises well to the inflow wind profiles that are not present in the training dataset.

Last but not least, to complete this parametric study, the predictions for a series of turbine yaw angles and upstream wake magnitudes are carried out. The results are shown in Figure 5.8, where the freestream wind speed is set as 8m/s. As can be seen, the trend of the wake deflection and the wake deficit is captured with respect to the turbine yaw angles and the upstream wake magnitudes.

In summary, the results in Figure 5.6, 5.7, and 5.8 fully demonstrate that the developed CGAN wake model generalises well to unknown flow scenarios and learns the qualitative features of wind turbine wake flows successfully. It is concluded that the developed model can achieve efficient and robust predictions of wind turbine wakes.

5.3.4 A Case Study

To further demonstrate the use of the developed CGAN wake model for wind farm applications, a case study for a small wind farm is carried out in this subsection. The considered wind farm consists of six NREL 5-MW wind turbines and the turbine layout is illustrated in Figure 5.9. For comparison purpose, the high-fidelity simulations using SOWFA are also carried out for this wind farm. The corresponding mesh configuration is shown in Figure 5.9 where two-level mesh refinement is applied so that the mesh size in the turbine wake region is 3m. The total number of mesh is around 2.6×10^7. The SOWFA simulations are carried out with a precursor atmospheric boundary layer simulations where the mean wind speed at hub height is set as 10m/s and the free-stream turbulence intensity (FSTI) is set as 6%. After

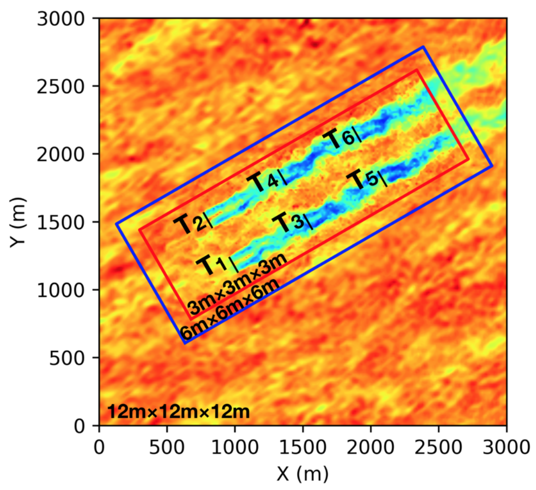
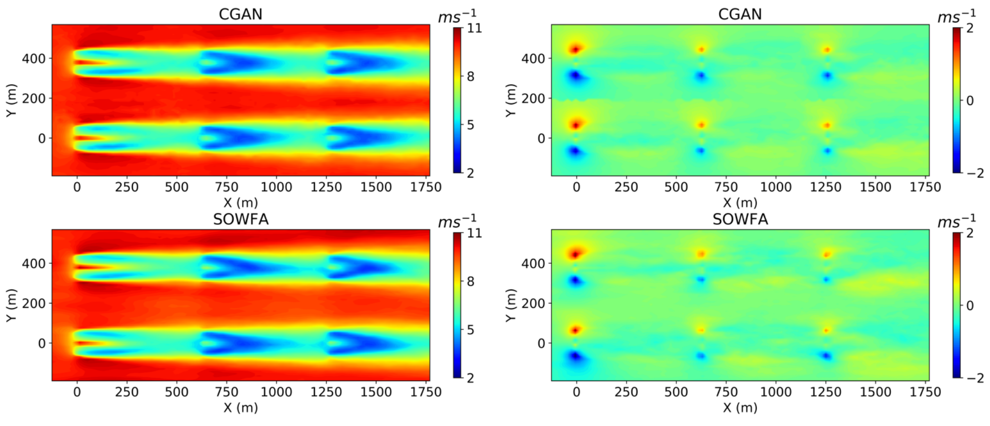


Figure 5.9: The illustration of the considered wind farm layout and the mesh configuration for high-fidelity simulations.

wind farm simulations, the mean flow field in the 2D horizontal plane at turbine hub height is extracted. In order to compare the CGAN model predictions with SOWFA results at the same wind conditions, the freestream wind profile is extracted from SOWFA and then used as the inflow wind conditions for the CGAN model.

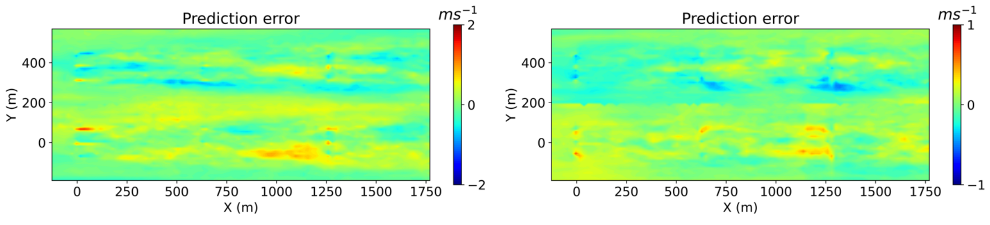
The prediction of single turbine wakes has been demonstrated in previous subsections. The extension to wind farm wake prediction in this subsection follows the method proposed in Chapter 4. Specifically, it is carried out by predicting single turbine wakes from upstream to downstream locations, where the inflow wind profiles for the downstream turbines are specified by the prediction results of the upstream turbine wakes. The flow field of the whole wind farm is finally obtained by combining the prediction results of all the single turbine wakes with the consideration of the matching conditions at the upstream boundary (all the other boundaries are not considered in this work). Further details can be referred to Chapter 4.

Two yaw control strategies are considered in this case study, i.e. the greedy case and the wake-steering case. The wind velocity fields, including both the streamwise and spanwise velocity, are predicted using both SOWFA and the developed CGAN model. For the greedy case, the yaw angles of all the wind turbines are set as 0°. The results are given in Figure 5.10 and 5.11. As shown, the streamwise and spanwise velocity fields predicted by the CGAN model match well with SOWFA results. It is observed that the rear turbines operate in the full wakes generated by the upstream turbines. Then the wake-steering case is investigated, where the yaw angles of all the wind turbines are specified according to the optimised yaw angles



(a) U_x (b) U_y

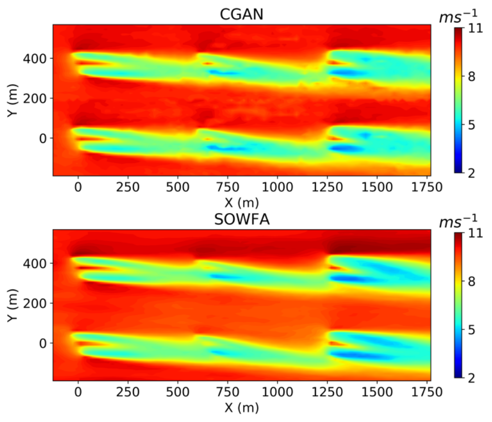
Figure 5.10: The prediction results by the developed CGAN model and SOWFA for the greedy case.



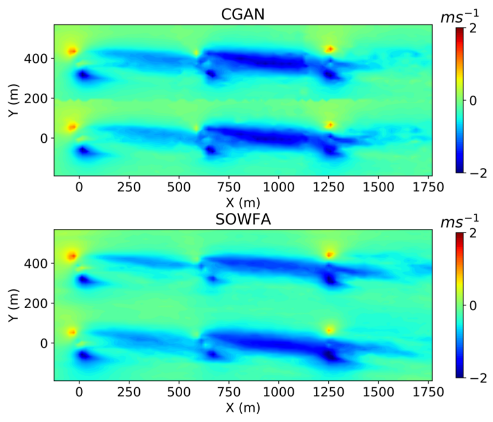
(a) U_x (b) U_y

Figure 5.11: The difference between the flow field predicted by the developed CGAN model and SOWFA for the greedy case.

reported in [40]. The yaw angles for T1, T2, T3, T4, T5 and T6 are [25.85°, 25.15°, 39.80°, 39.75°, 0.35°, 0.45°] respectively. The results are given in Figure 5.12 and 5.13. A great match is observed between the SOWFA and CGAN predictions, such as the yaw-induced wake deflections (see Figure 5.12(a)) and the yaw-induced span-wise wind speed magnitudes (see Figure 5.12(b)). As expected, the wakes generated by the front turbines are steered away from the rear turbines (i.e. T5 and T6) in this wake-steering case. The predictions in the wake regions show a small level of discrepancy with the SOWFA results. It is worth mentioning that the data used for training the CGAN model only includes the yaw angles in the range of $[-30°, 30°]$, while the yaw angles of T3 and T4 in this case largely exceed this range. Therefore, the wake-steering case further demonstrates the robustness of the developed CGAN wake model in the scenarios of extrapolated turbine yaw angles.

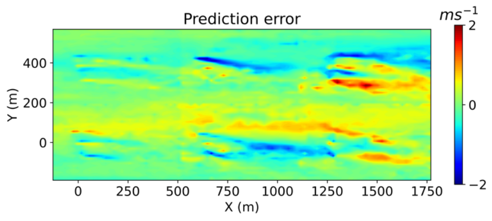


(a) U_x

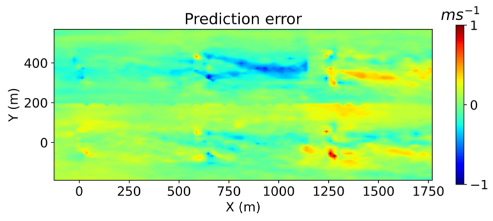


(b) U_y

Figure 5.12: The prediction results by the developed CGAN model and SOWFA for the wake-steering case.



(a) U_x



(b) U_y

Figure 5.13: The difference between the flow field predicted by the developed CGAN model and SOWFA for the wake-steering case.

5.4 Conclusions

In this work, a surrogate modelling method for general fluid flows was proposed based on the DC-CGAN. The proposed method was then applied to develop a novel wind farm wake model for the wake predictions under various inflow conditions and turbine yaw settings. The results showed that the developed model was able to predict the multi-channel (i.e. both streamwise and spanwise velocity fields) wind turbine wake flows accurately and efficiently. In particular, the prediction RMSEs for the streamwise and spanwise velocity fields were just 0.10m/s and 0.18m/s respectively which were just 1.1% and 4.1% of the corresponding value ranges, while the online computation time was only 0.002s. Also, the main features of turbine wakes were predicted very well, including both the overall features (such as the wake deflection with the turbine yaw angle, the wake recovery in the streamwise direction, the wake expansion in the spanwise direction) and the detailed features (such as the fluctuations in the freestream wind, the flow acceleration on both sides of the turbines, the turbines' blockage effects on the upstream flow, and the yaw-induced spanwise velocity in the wake regions). In addition, a parametric study was carried out to systematically evaluate the generalisation performance of the developed wake model. The results showed that the model generalised well to the flow scenarios that were not present in the training dataset, and that the developed model successfully learned the qualitative features of wakes of yawed wind turbines over a range of velocities, including the change of the wake deficit with upstream inflow profiles, wake deflection with turbine yaw angles, yaw-induced spanwise velocity, and wake interactions through the upstream interface. It is concluded that the developed model achieved accurate and robust real-time predictions of static wind farm wakes.

Furthermore, a case study for a small wind farm was carried out using both the developed model and the high-fidelity LES model. The comparison study showed a good agreement between the developed model and the LES model, including the predictions for both the streamwise and the spanwise velocity fields. This case study fully demonstrated the performance of the developed model in predicting wind farm wake flows. However, as the developed wake model only considers the impact of the upstream flow on the downstream turbine through the upstream interface, it cannot be used (yet) for multiple adjacent wakes since lateral boundaries are not coupled. In addition, this chapter still focused on static wake modelling. Thus the dynamic features of wind farm wakes are still missing. The dynamic wind farm wake modelling will be investigated in the next chapter.

Chapter 6

A Novel Dynamic Wind Farm Wake Model Based on Deep Learning

6.1 Introduction

In this chapter, a surrogate modelling method for unsteady distributed fluid systems is proposed to build a novel dynamic wind farm wake model. In the proposed method, the high-dimensional flow field data is first reduced to low-dimensional coefficients by POD [123]. Then a deep recurrent neural network (RNN), called LSTM [14,180], is employed to predict the reduced coefficients at current time step based on the reduced coefficients in the previous time steps. POD is chosen as it can capture the main flow features according to the flow field's energy content while LSTM is chosen as it is very powerful in handling time-series predictions. The proposed method is hereby referred as POD-LSTM. It is then employed for wind farm wake modelling. First, a series of LESs are carried out with wind turbine rotors operating in different yaw conditions under different turbulent inflows. Then the generated LES database is used to train the ML model.

The main contributions of this chapter are summarised as follows:

(1) A novel data-based dynamic wind farm wake model is developed and validated through a series of simulation tests including single turbine wakes, multiple turbine wakes, yawed wakes, and wake interactions within a large wind farm. As the existing wake models in the literature are either unable to predict unsteady wake details (low-fidelity models) or too time-consuming to run (high-fidelity models), this work bridges the research gap by developing a wake model that can predict

Type	Analytical	Numerical	Machine learning
Models	Jensen, Frandsen, FLORIS, 3D wake, ...	PALM, UTDWF, Nalu-Wind, SOWFA, ...	The model developed in this chapter
Based on	flow observations	NS equations	LES database
Method	flow analysis	CFD	deep learning
Speed	fast	slow	fast
Accuracy	low	high	moderate/high
Flow details	no	yes	yes

Table 6.1: The comparison of the developed ML-based wake model with existing wake models.

unsteady wind turbine wakes similarly as high-fidelity wake models while running as fast as the low-fidelity static wake models. The comparison of the developed model with existing wake models is summarised in Table 6.1.

(2) A deep learning based surrogate modelling method, called POD-LSTM, is proposed to build this novel dynamic wake model. The proposed method is generic and can also be used to model other unsteady fluid flows around distributed structures, such as tidal farms and building arrays in the atmospheric boundary layer.

(3) A high-fidelity CFD database of wind flow around turbine rotors is generated, through a series of CFD simulations which takes around 7×10^5 CPU hours' run time on local HPC clusters. The above deep learning based dynamic wind farm wake model is then trained to learn the complex wind farm wake dynamics from this valuable database.

The remaining part of this chapter is organised as follows: the proposed deep learning based surrogate modelling method is described in Section 6.2. Its application on wind farm wake modelling is described in Section 6.3. After developing the wake model, the model validation and prediction are carried out in Section 6.4, where the prediction results on wakes behind a turbine with changing yaw and wake interactions in a 9-turbine wind farm are demonstrated. Finally, the conclusions are drawn in Section 6.5.

6.2 Methodology

An example distributed fluid system, a wind farm, is illustrated in Figure 6.1, where M distributed structures (wind turbines in this example) with distributed control parameters $[d_1, d_2, ..., d_M]$ (such as the yaw angle and blade pitch angle for a wind

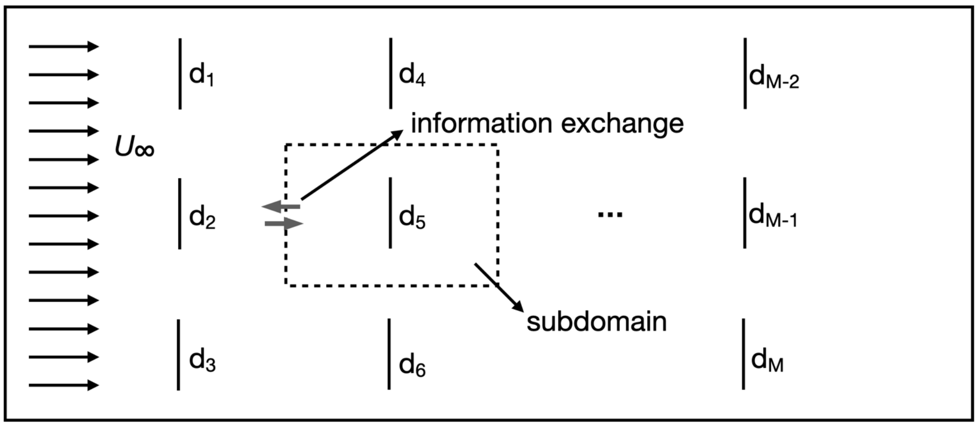


Figure 6.1: The illustration of a typical distributed fluid system, where a subdomain containing one distributed structure is illustrated by the dashed rectangular. The information exchange in this work only includes the upstream subdomain's impact on the adjacent downstream subdomain.

turbine) are shown in the rectangular flow domain. With a CFD approach, a mesh is first generated for the whole flow domain and then the discretised governing equation (such as the NS equations) is solved on the mesh. This approach is usually costly as the DoF (the number of cells) is very high. In this section, an ML-based surrogate modelling approach is developed to build a surrogate model that can predict the flow field efficiently given the distributed control parameters (wind turbines' operating parameters $[d_1, d_2, ..., d_M]$ in this example). The proposed surrogate modelling procedure includes four steps as shown below.

6.2.1 Design of Experiments

This step serves to generate/collect high-fidelity training data. For a system with M distibuted structures as in Figure 6.1, a set of design input variables, denoted as $\mathscr{D}$, are generated according to a sampling strategy (such as Latin hypercube sampling), where $\mathscr{D}$ is a matrix of shape $[M, N_t, P]$ with $\mathscr{D}_{m,i_t,p}$ representing the design value of the p^{th} control parameter of the m^{th} distributed structure at time step i_t. N_t is the total number of time steps and P is the dimension of each distributed control parameter (for example, $P = 2$ for the wind turbine case if two control parameters, e.g., turbine yaw angle and turbine blade pitch angle, are considered). The corresponding output variables, the flow field data $\mathscr{U}$, are obtained by running CFD or wind tunnel experiment with the designed input $\mathscr{D}$. Here $\mathscr{U}$ is a matrix of shape $[M, N_t, N_x]$ with $\mathscr{U}_{m,i_t,i_x}$ representing the value of the flow velocity magnitude in the m^{th} subdomain at spatial coordinate indexed by i_x at time step i_t. N_x is the total number of grid points in each subdomain. A matrix $\mathscr{X}$ of shape [N_x, d] is used to record the location of all the grid points within a subdomain relative to the corresponding distributed structure location, where $\mathscr{X}_{i_x,1:d}$ represents the d-dimension spatial coordinate (for example $d = 2$ for 2-D plane) indexed by i_x.

In order to expand the training dataset to include more scenarios, a set of CFD/experiments can be carried out with different inflow conditions and design input variables. All the resulting data can then be collected and reshaped together as the final training dataset. If in total S simulations/experiments are carried out, the final design input matrix $\mathscr{D}$ is of shape [$M \times S$,N_t,P] and the design output matrix $\mathscr{U}$ containing all the flow field data is of shape [$M \times S$,N_t,N_x].

6.2.2 Dimensionality Reduction

Here, the design output matrix $\mathscr{U}$ obtained in previous step is first reshaped into a matrix U of shape [$M \times S \times N_t$,N_x], with each row in U representing a snapshot of the flow field. The POD as in [123] is done by SVD as

$$U^T = V\Sigma W^T, \tag{6.1}$$

where the k^{th} column vector of V, denoted as v_k, is the k^{th} POD basis. Setting the total number of POD basis as R, each snapshot of the flow field, $\tilde{u}$, can then be approximated by

$$\hat{u} = \sum_{k=1}^{R} \alpha_k v_k, \tag{6.2}$$

where the reduced coefficients α_k are calculated by

$$\alpha_k =< \tilde{u}, v_k >, 1 \leq k \leq R. \tag{6.3}$$

In this way, the output matrix $\mathscr{U}$ can be reduced into its reduced representation $\mathscr{U}^r$ of shape [$M \times S$,N_t,R], where $\mathscr{U}^r_{m,i_t,r}$ represents the r^{th} reduced coefficient of the flow field $\mathscr{U}_{m,i_t,1:N_x}$. This dimensionality reduction process thus reduces the original flow field dimension from N_x to R. It is worth noting that other dimensionality reduction techniques can also be used. The ICA [166] and the AE [167] have been implemented and tested for the work in this chapter, and the results are omitted here as their performances are not as good as POD.

6.2.3 Neural Network Training

After dimensionality reduction, a supervised ML problem is formulated to predict the reduced coefficients at the current time step based on historical data of the flow. The LSTM network, which is particularly powerful in time-series predictions, is employed here.

The overall data pipeline is illustrated in Figure 6.2, where the flow field, the inflow velocity, and the distributed control parameters from time steps 1 to T are required as the input in order to predict the flow field at time step $T + 1$. The flow fields from 1 to T are first reduced to their POD coefficients, which are then fed into the LSTM network along with the inflow and distributed control parameter history to predict the POD coefficients at time step $T+1$. Finally the flow field at time step $T+1$ is reconstructed based on the predicted POD coefficients according to Equation 6.2. The LSTM network in Figure 6.2 is detailed in Figure 6.3 which shows that the POD coefficients, the inflow velocity, and the distributed control parameters' value are standardised by the standard scalers before being fed into the LSTM cells. The scalers (denoted as Scaler1 and Scaler2 respectively in Figure 6.3) standardise the input features (the POD coefficients and the rest features respectively) by removing their mean and scaling to unit variance. A dense layer is stacked on top of the LSTM cells to predict the standardised POD coefficients at the next time step, which is then scaled back to obtain the POD coefficient predictions through the inverse transform of the Scaler1.

For the model training, a data generator is implemented which extracts the training input and corresponding training target by mini-batches from the database $\mathscr{D}$ and $\mathscr{U}$, and then feeds them to the LSTM network. The LSTM network is trained to minimise the MSE between the network output and the training target. The Adam optimisation algorithm [158] is used with a learning rate of 0.001 for the NN training in this chapter. The model is implemented based on the ML package Keras [176] with Tensorflow backend [155].

After training, the POD-LSTM model can predict the flow field at time step $T + 1$ (i.e. $\hat{u}_{T+1}$), given the history of the flow field data (i.e. $[\tilde{u}_1, \tilde{u}_2, ...\tilde{u}_T]$), inflow velocity (i.e. $[u^0_1, u^0_2, ...u^0_T]$), and the distributed control parameters' value (i.e. $[d_1, d_2, ...d_T]$). For the prediction of the flow field at time step $T + 2$ (i.e. $\hat{u}_{T+2}$), the predicted flow field at time step $T+1$ (i.e. $\hat{u}_{T+1}$), the user-defined inflow velocity at time step $T + 1$ (i.e. u^0_{T+1}), and the user-defined distributed control parameter's value at time step $T + 1$ (i.e. d_{T+1}) along with the history data (i.e., $[\tilde{u}_2, ...\tilde{u}_T]$, $[u^0_2, ...u^0_T]$, $[d_2, ...d_T]$) are fed into the data pipeline. In this way, all future flow fields can be predicted iteratively. It is worth noting that the prediction error

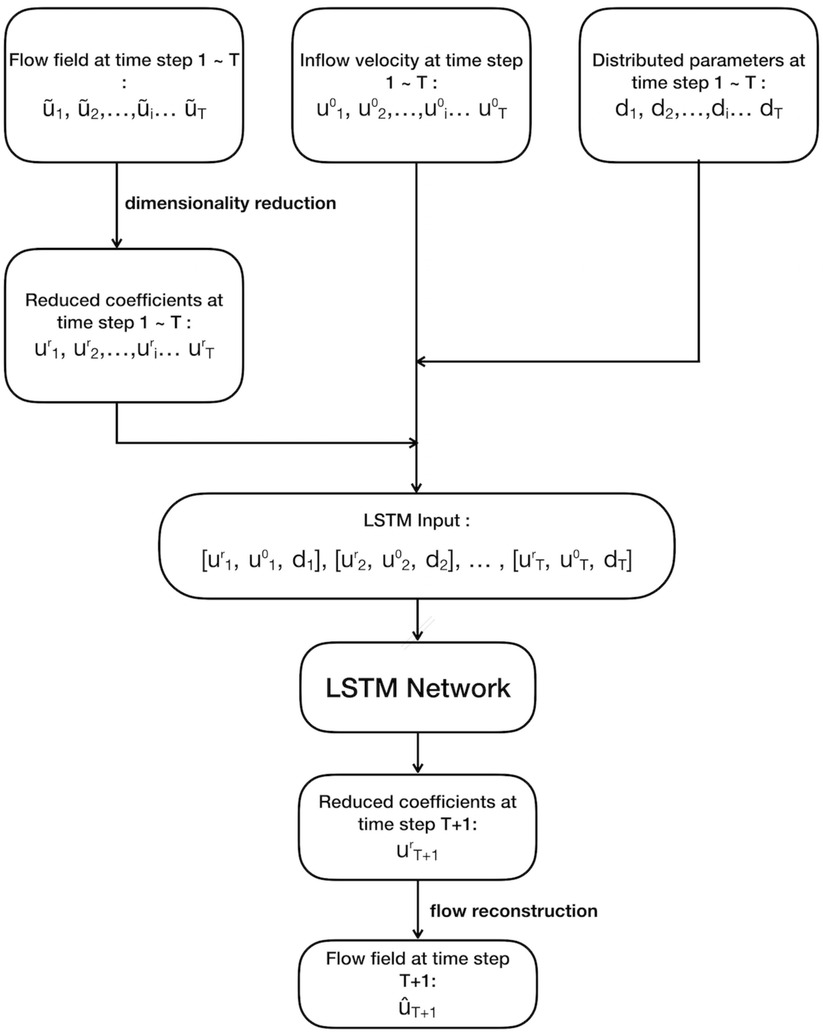

Flow field at time step 1 ~ T :
ũ1, ũ2,…,ũi… ũT
Inflow velocity at time step 1 ~ T :
u01, u02,…,u0i… u0T
Distributed parameters at time step 1 ~ T :
d1, d2,…,di… dT
dimensionality reduction
Reduced coefficients at time step 1 ~ T :
ur1, ur2,…,uri… urT
LSTM Input :
[ur1, u01, d1], [ur2, u02, d2], … , [urT, u0T, dT]
LSTM Network
Reduced coefficients at time step T+1:
urT+1
flow reconstruction
Flow field at time step T+1:
ûT+1

Figure 6.2: An illustration of the data pipeline of the proposed POD-LSTM method.

will accumulate in the iterative prediction procedure. Therefore, for the long-term flow field prediction, it is more interesting to analyse the results qualitatively instead of quantitatively.

6.2.4 Prediction of the Whole Flow Field

The POD-LSTM model developed above can predict the flow field in a single sub-domain at all future time steps given the history of the flow field and the future inflow velocity and distributed control parameters' value. For the prediction of the whole flow field, the trained POD-LSTM model is used to predict the flow field in each subdomain sequentially from upstream to downstream, and then the whole

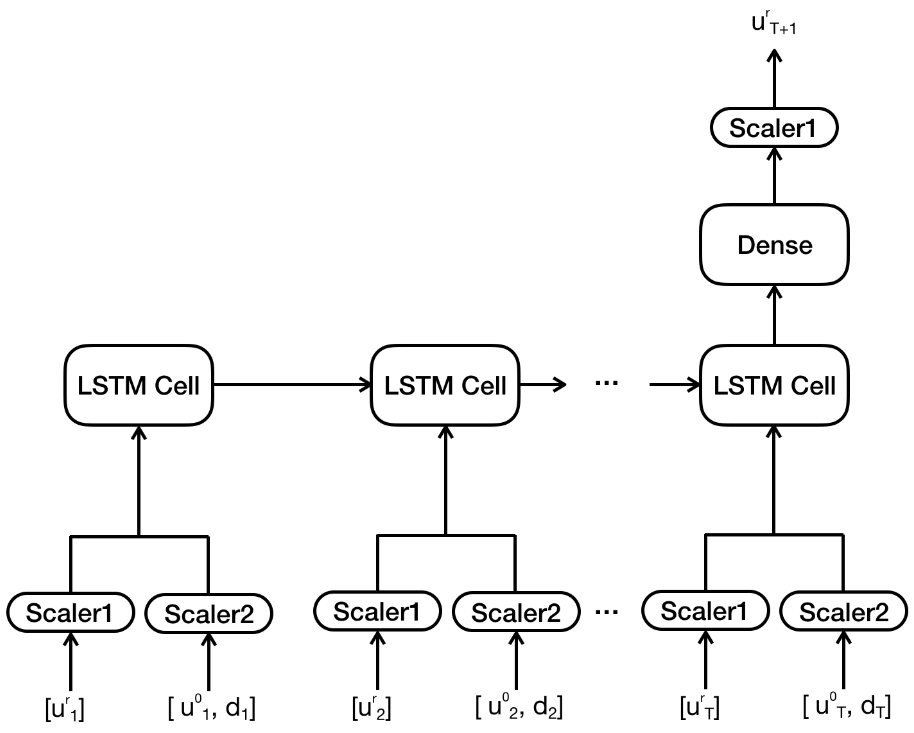


Figure 6.3: The detailed illustration of the LSTM network in Figure 6.2. The scalers at the input of the LSTM cells represent the forward transforms of the scalers while the scaler at the output of the LSTM cell represents the corresponding inverse transform.

flow field is obtained by combining all the subdomains' predictions with the consideration of the upstream flow's impact on the downstream subdomain through the domain interface in the streamwise direction. It is worth mentioning that the information exchange in the lateral boundaries is not considered in this work, which could improve the developed wake model's performance by the inclusion of lateral interactions. However, this will require much more training data characterizing the lateral interactions and also new method to tackle the two-way coupling at these interfaces. Therefore, it is not investigated in this thesis and is treated as a future work. The detailed procedure for predicting the flow field around multiple wind turbines is summarised below as Algorithm 5.

Algorithm 5 The prediction of the whole flow field

```
1: Divide the whole flow domain into $M$ subdomains (as illustrated in Figure 6.1) and number them from upstream to downstream as $[1, 2, ..., M]$.
2: Initialize the flow field history in all subdomains: $\{[\tilde{u}_{1,i}, \tilde{u}_{2,i}, ...\tilde{u}_{T,i}], 1 \leq i \leq M\}$.
3: Initialize the inflow velocity history in all subdomains: $\{[u^0_{1,i}, u^0_{2,i}, ...u^0_{T,i}], 1 \leq i \leq M\}$.
4: Initialize the distributed control parameters' history in all subdomains: $\{[d_{1,i}, d_{2,i}, ...d_{T,i}], 1 \leq i \leq M\}$.
5: Set total prediction step $T_{tot}$.
6: $k \leftarrow 1$.
7: while $k \leq T_{tot}$ do
8:     for i in $[1, 2, ..., M]$ do
9:         Propagate the input $[\tilde{u}_{k,i}, \tilde{u}_{k+1,i}, ...\tilde{u}_{T+k-1,i}]$, $[u^0_{k,i}, u^0_{k+1,i}, ...u^0_{T+k-1,i}]$, $[d_{k,i}, d_{k+1,i}, ...d_{T+k-1,i}]$ through the data pipeline shown in Figure 6.2 to obtain the flow field in the $i^{th}$ subdomain at time step $T + k$: $\tilde{u}_{T+k,i}$.
10:    end for
11:    Obtain the inflow velocity at time step $T + k$: $u^0_{T+k,i}, 1 \leq i \leq M$, by setting it directly from the user-defined boundary condition for the most upstream subdomains, while for the downstream subdomains, extracting the inflow velocity from the neighbouring upstream subdomains' flow field predictions.
12:    Set the distributed control parameters' value at time step $T + k$ from user-defined values: $d_{T+k,i}, 1 \leq i \leq M$.
13:    Output the whole flow field at time step $T + k$ by combining $\tilde{u}_{T+k,1}, \tilde{u}_{T+k,2}, ..., \tilde{u}_{T+k,M}$ together.
14:    $k \leftarrow k + 1$
15: end while
16: The unsteady flow fields for the whole domain from time $T + 1$ to $T + T_{tot}$ are obtained.
```

6.3 Development of a Novel Dynamic Wind Farm Wake Model

The POD-LSTM method developed above is employed to build a novel dynamic wake model in this section.

6.3.1 High-Fidelity Data Generation

The high-fidelity flow field data is generated using the LES flow solver SOWFA [160]. The turbine rotors are modelled as actuator lines in this work. The mesh generation and the simulation domain in this chapter are the same as in Chapter 4, which are illustrated in Figure 4.8 and 4.9. For turbine wake simulations, a precursor simulation of the atmospheric boundary layer is first carried out to obtain

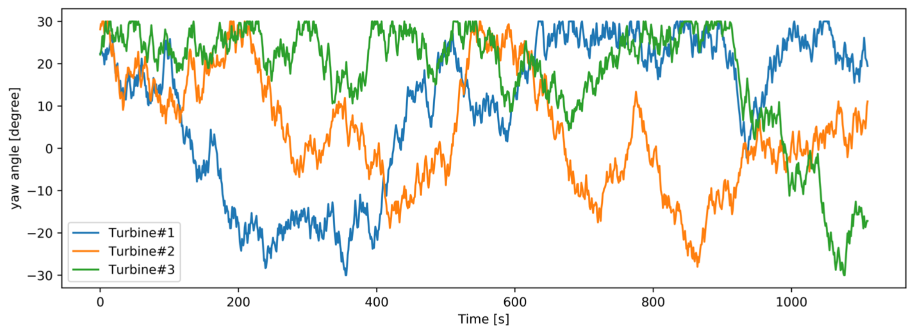


Figure 6.4: An example of the designed yaw angles in a simulation case. The yaw angles of all the three turbines are included.

the initial flow field and the inflow boundary condition, and then three NREL 5-MW wind turbines with the baseline turbine pitch and torque control [165] are added in the simulation domain with a downstream spacing of 5 rotor diameters. For each simulated case, 1110s simulations are carried out with a time step of 0.02s.

Three inflow conditions with average wind speeds of 8m/s, 9m/s, and 10m/s and an FSTI of 6%, are considered. Twenty simulations are carried out for each inflow condition, with different yaw angles for each simulation case. The yaw angles are designed by random initial yaw and random yaw changes of less than $3°$ every second during the whole simulation period. The yaw angles are limited to the range $[-30°, 30°]$. By these settings, the generated LES data covers a wider flow speed range and turbine yaw range. An example of the designed yaw angles in a simulation case is shown in Figure 6.4, where random yaw changes are applied at each time step. After CFD simulations, the flow fields at turbine hub-height are sampled every second to extract the training data. The first 400s simulation results are discarded as the turbine wakes have not been well established during this period. Therefore, 710 snapshots of the flow field are recorded for each case. Then the flow field in each subdomain containing one turbine (as shown in Figure 4.9) is extracted and interpolated into a uniform grid of 50×30. All the generated flow field data is collected together to form the training dataset $\mathscr{U}$. The shape of the final training matrix $\mathscr{U}$ is [180,710,1500], with $\mathscr{U}_{s,i_t,1:1500}$ representing a snapshot of the flow field in one subdomain at time step i_t for the scenario indexed by s. All the designed yaw angles are collected as the design input matrix $\mathscr{D}$ of shape $[180, 1110, 1]$, where $\mathscr{D}_{s,i_t,1}$ represents the yaw angle at time step i_t for the scenario indexed by s. Here a scenario designates the unsteady flow fields in one subdomain of one simulation case. The whole data generation process takes around 7×10^5 CPU hours where each simulation requires around 46 hours' computation on a local cluster with 256 CPUs.

6.3.2 Model Training

The generated LES data contains 180 flow scenarios with each scenario consisting of the unsteady flow fields at 710 discrete time instants. For model training purpose, the whole dataset is divided into a training dataset (the first 64% time instants), a validation dataset (the 64% $\sim$ 85% time instants), and a test dataset (the last 15% time instants). The training dataset is fed into the POD-LSTM network by mini-batches with a batch size of 1024 while the validation dataset is used to evaluate the model after each training epoch. The test dataset is not used in the training process but only for model testing after training.

Dropout, including the input dropout and the recurrent dropout, is an efficient technique to tackle overfitting. The LSTM networks with and without the input and recurrent dropout are both tested. It turns out that the one with dropout performs much better, thus it is used in this chapter. The stack of multiple LSTM layers does not further increase the model performance thus only one LSTM layer, as illustrated in Figure 6.3, is included in the POD-LSTM model. There are still a few hyper-parameters not yet determined in the POD-LSTM network, i.e., the total lookback time step of the flow history, the number of POD basis, and the output features' dimension of the LSTM cell. The validation errors are used to determine these hyper-parameters' empirical values, by a grid-search procedure. The final hyper-parameters' values are given in Table 6.2, along with the evaluations of the POD-LSTM model's performance by using the test dataset. Here T represents the total lookback time step of the flow history in order to predict the current flow field, R represents the number of the POD basis, N_h represents the output features' dimension of the LSTM cell, and α represents the dropout rate of both input dropout and recurrent dropout. The model reduction error ϵ_{POD} (m/s) and the total prediction error ϵ_{total} (m/s) are also included in Table 6.2 for model evaluations. The POD model reduction error is defined as the mean value of the RMSEs between the reconstructed flow fields from the exact POD coefficients and the exact flow fields:

$$\epsilon_{POD} = \frac{1}{180 \times 102} \sum_{s=1}^{180} \sum_{i_t=609}^{710} RMSE(\mathscr{U}_{s,i_t,1:1500}, \mathscr{U}^{POD}_{s,i_t,1:1500}), \tag{6.4}$$

where

$$\mathscr{U}^{POD}_{s,i_t,1:1500} = \sum_{k=1}^{R} < \mathscr{U}_{s,i_t,1:1500}, v_k > v_k. \tag{6.5}$$

T	R	N_h	α	ϵ_{POD}	ϵ_{total}
5	350	350	0.2	0.328	0.428

Table 6.2: The hyper-parameters in the POD-LSTM model and the model evaluations.

And the POD-LSTM model prediction error is defined as the mean value of the RMSEs between the flow fields predicted by the POD-LSTM model and the exact flow fields:

$$\epsilon_{total} = \frac{1}{180 \times 102} \sum_{s=1}^{180} \sum_{i_t=609}^{710} RMSE(\mathscr{U}_{s,i_t,1:1500}, \hat{\mathscr{U}}_{s,i_t,1:1500}). \tag{6.6}$$

where $\hat{\mathscr{U}}_{s,i_t,1:1500}$ represents the POD-LSTM predictions. As shown in Table 6.2, the POD-LSTM prediction error arises from both the representation of the flow field by the reduced coefficients, which is characterised by ϵ_{POD}, and the difference of the exact POD coefficients and the ones predicted by the LSTM network, which is characterised by $\epsilon_{total} - \epsilon_{POD}$. The overall prediction error ϵ_{total} is 0.428m/s, which is just 4.8% with respect to the freestream wind speed.

After training, the POD-LSTM model can be used for the prediction of the flow field of the next second given the flow history in the past five seconds. This prediction process can be carried out iteratively so that all the future flow fields can be predicted with a time step of 1s.

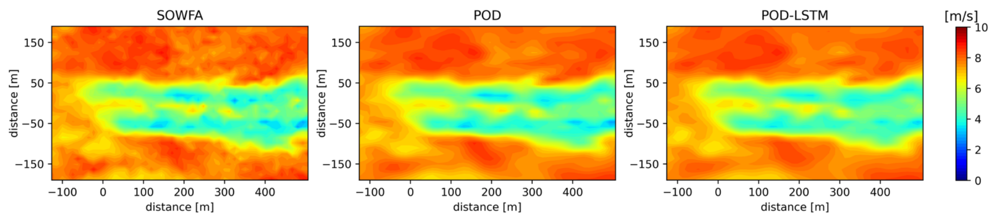
6.4 Results and Discussions

The flow field predictions, including both the single-turbine wake and multiple-turbine wake predictions, are carried out using the above developed dynamic wake model. The results are compared with the high-fidelity SOWFA simulation results for model validations. After that, two simulation case studies are carried out to demonstrate the model's ability in capturing the yaw effect on turbine wakes and in simulating large-scale wind farms.

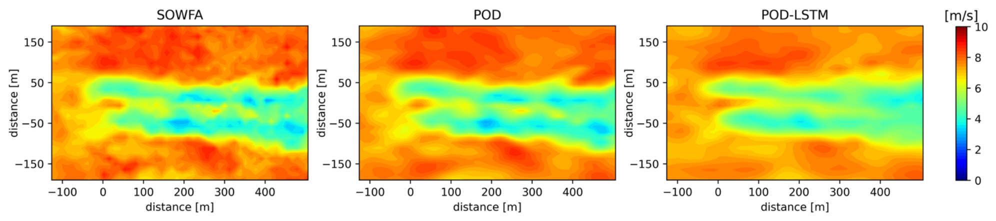
6.4.1 Model Validations

Single-Turbine Wake Predictions

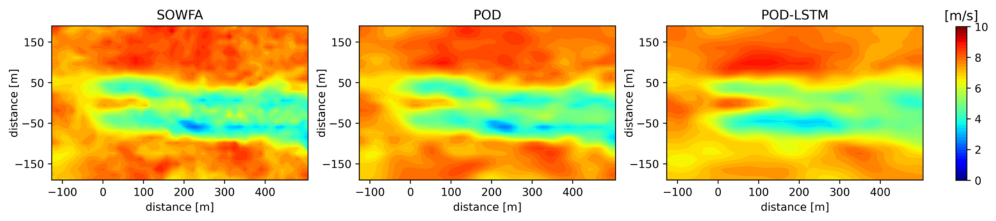
The single-turbine wake predictions are carried out and compared with the test dataset. The POD-LSTM model is used to predict both the flow field in one time step directly and the flow fields in all future time steps iteratively. To predict the



(a) Time step: T



(b) Time step: $T + 10$



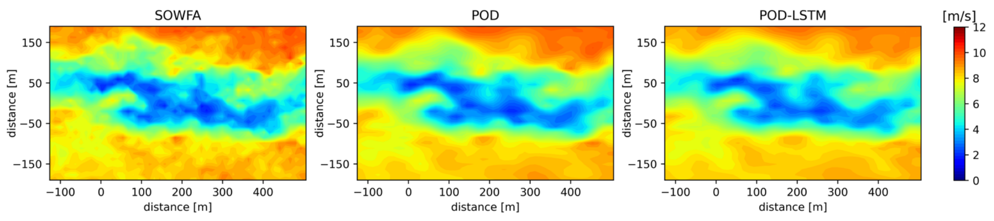
(c) Time step: $T + 20$

Figure 6.5: An example case of the single-turbine wake prediction with the turbine operating in freestream condition. The results include the SOWFA predictions, the flow field reconstructions from exact POD coefficients, and the POD-LSTM model predictions at time step (a) T, (b) $T + 10$, and (c) $T + 20$. The turbine rotor is located at $(0, 0)$m of the 2D plane.

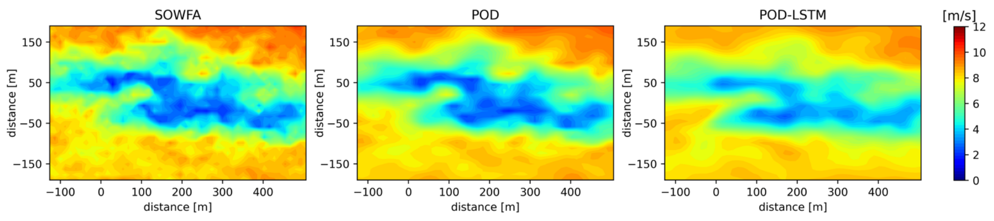
flow fields from time step T to $T + T_{tot}$, the calculation by the POD-LSTM model uses the same initial flow fields as SOWFA only from time step $T - 5$ to $T - 1$, the same inflow conditions as SOWFA from time step T to $T + T_{tot}$, and the same yaw angles as SOWFA from time step T to $T + T_{tot}$.

The predictions are carried out for all the cases in the test dataset. Two typical cases are chosen to demonstrate the model's performance, including one with the turbine operating in freestream condition and the other with the turbine operating in the front turbine's wake. The results are shown in Figure 6.5 and 6.6, including the SOWFA predictions, the flow field reconstructions from exact POD coefficients and the POD-LSTM model predictions, at time step T, $T + 10$ and $T + 20$. The corresponding error distributions are shown in Figure 6.7 and 6.8.

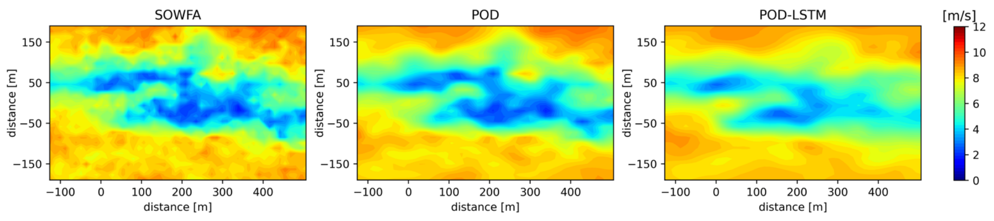
As can be seen, the reconstructions from exact POD coefficients match with



(a) Time step: T



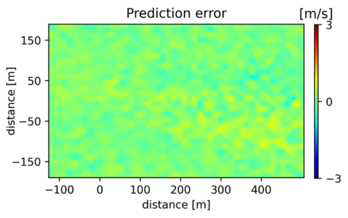
(b) Time step: $T + 10$



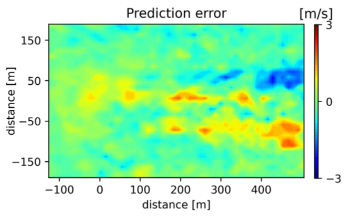
(c) Time step: $T + 20$

Figure 6.6: An example case of the single-turbine wake prediction with the turbine operating in the front turbine's wake. The results include the SOWFA predictions, the flow field reconstructions from exact POD coefficients, and the POD-LSTM model predictions at time step (a) T, (b) $T + 10$, and (c) $T + 20$. The turbine rotor is located at $(0, 0)$m of the 2D plane.

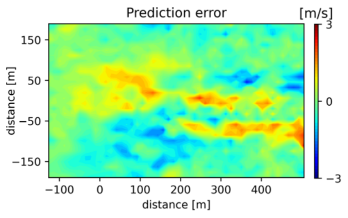
SOWFA results quite well for all time steps in both cases, which illustrates that the chosen POD basis captures the main flow dynamics very well thus this dimensionality reduction process can be combined with the subsequent ML model for the accurate flow field predictions, as in [118, 125]. The direct and iterative flow field predictions at time step T, $T + 10$ and $T + 20$ by the POD-LSTM model match with the POD reconstruction results very well in both cases, which demonstrates that the LSTM network can predict the POD coefficients accurately. The overall prediction error is small (as shown in Figure 6.7 and 6.8), considering the chaotic nature of the turbulent wakes and limited information used for these predictions.



(a) Time step: T



(b) Time step: $T + 10$



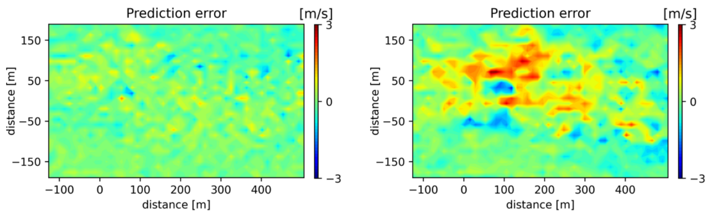
(c) Time step: $T + 20$

Figure 6.7: An example case of the single-turbine wake prediction with the turbine operating in freestream condition. The figures show the difference between the POD-LSTM model predictions and the SOWFA predictions at time step (a) T, (b) $T + 10$, and (c) $T + 20$. The turbine rotor is located at $(0, 0)$m of the 2D plane.

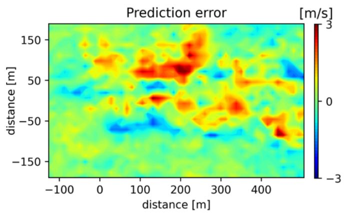
Multiple-Turbine Wake Predictions

The multiple-turbine wake predictions are carried out in this subsection to demonstrate the POD-LSTM model's ability in capturing wake interactions. The case of two turbines in a row with a downstream spacing of 5 rotor diameters is considered. The POD-LSTM model is used to predict the flow field in one time step directly and the flow fields in all future time steps iteratively. To predict the flow fields from time step T to $T + T_{tot}$, the calculation by the POD-LSTM model uses the same initial flow fields as SOWFA only from time step $T-5$ to $T-1$, the same freestream conditions (that is, the inflow conditions for the front turbine) as SOWFA from time step T to $T + T_{tot}$, and the same yaw angles as SOWFA for all the turbines from time step T to $T + T_{tot}$.

The predictions are carried out for all the flow conditions in the test dataset. An example case is chosen to demonstrate the model's performance in capturing the wake interactions. The results are shown in Figure 6.9 and 6.10, including the SOWFA predictions, the POD-LSTM model predictions and the prediction errors at time step T, $T + 10$, and $T + 20$. As can be seen, the direct and iterative flow field predictions at time step T, $T+10$ and $T+20$ by the POD-LSTM model match with SOWFA simulation results quite well for both the front turbine's and the rear



(a) Time step: T (b) Time step: $T+10$



(c) Time step: $T+20$

Figure 6.8: An example case of the single-turbine wake prediction with the turbine operating in the front turbine's wake. The figures show the difference between the POD-LSTM model predictions and the SOWFA predictions at time step (a) T, (b) $T+10$, and (c) $T+20$. The turbine rotor is located at $(0,0)$m of the 2D plane.

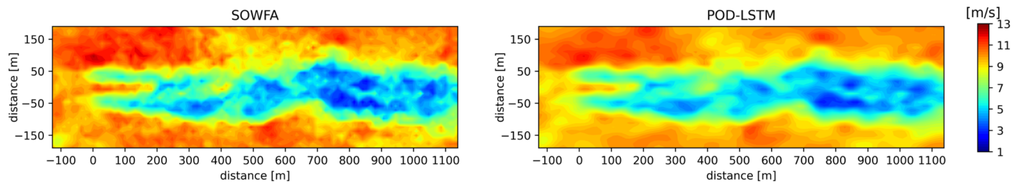
turbine's wake, which demonstrates that the proposed model can predict the wake flow around multiple wind turbines with the consideration of the impact of the upstream flow on the downstream turbine. It is worth noting that the impact of the upstream turbine on the downstream turbine is well captured in all the prediction time steps, which is essential in guaranteeing the performance of the developed model in large-scale wind farm predictions.

6.4.2 Model Predictions - Two Case Studies

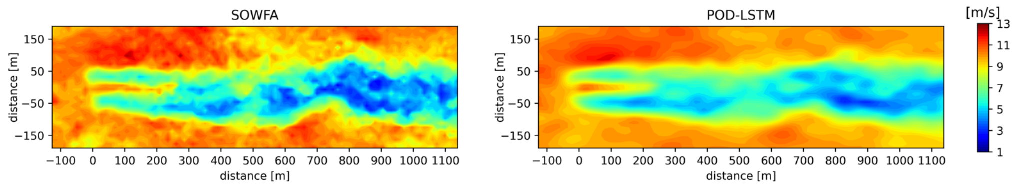
The Yaw Effect on Turbine Wakes

A single-turbine case with designed yaw change is investigated using the developed model to demonstrate its ability in capturing the yaw effect on turbine wakes. The single-turbine wake is predicted for a simulation time of 300s, with the yaw angle being $-20°$ for the first 100s, then increasing linearly from $-20°$ to $20°$ in the next 100s, and staying at $20°$ for the last 100s. The snapshots at 100s and 300s are shown in Figure 6.11. As can be seen, the impact of turbine yaw on unsteady turbine wakes is captured, where the wake deflection is predicted successfully.

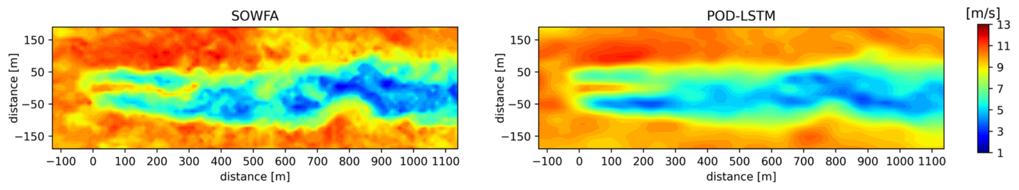
The video showing the unsteady flow field visualisation is available online in the supporting materials of the published paper. As can be seen from the video,



(a) Time step: T



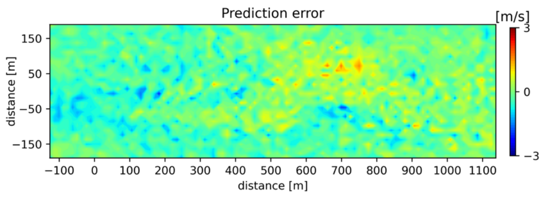
(b) Time step: $T + 10$



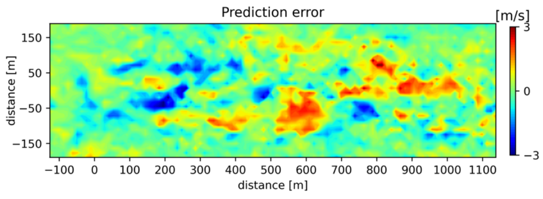
(c) Time step: $T + 20$

Figure 6.9: An example case of the multiple-turbine wake predictions with two turbines in a row. The results include the SOWFA predictions and the POD-LSTM model predictions at time step (a) T, (b) $T + 10$, and (c) $T + 20$. The front and the rear turbine rotors are located at $(0, 0)$m and $(632, 0)$m of the 2D plane respectively.

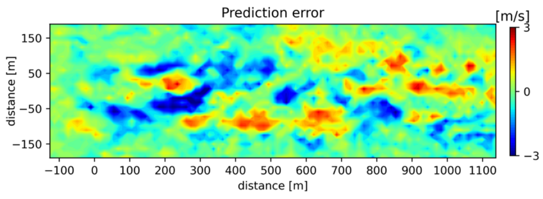
the main feature of the unsteady turbine wake, such as the streamwise convection of flow structures, the wake's crosswind meandering, and the wake's deflection with changing yaw are captured clearly by the developed model during the whole simulation duration. This further validates the developed model's ability in capturing main flow features for long time simulations. To the best of the author's knowledge, there are no existing wake models that can achieve fast predictions of these unsteady flow features. It is worth mentioning that the successful prediction of the streamwise convection and crosswind meandering of flow structures is not trivial, as the LSTM network is not trained to predict the velocity at specific locations but the POD coefficients which do not directly reflect the spatial convection of the flow. The POD only serves as the dimensionality reduction technique and the POD basis does not characterise the coherent structures as in [181], because the flow field snapshots in the training dataset are collected from different simulations under random flow parameters. In addition, this case also demonstrates the generalisation performance



(a) Time step: T



(b) Time step: $T + 10$



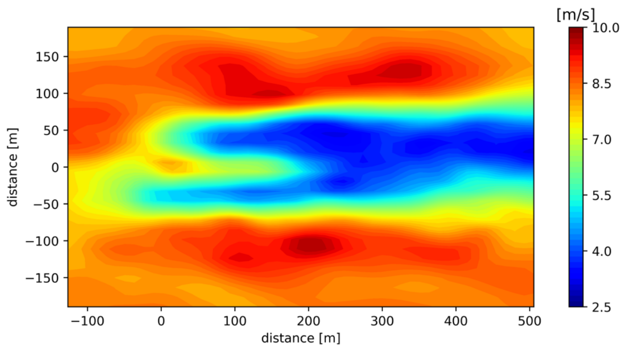
(c) Time step: $T + 20$

Figure 6.10: An example case of the multiple-turbine wake predictions with two turbines in a row. The figures show the difference between the POD-LSTM model predictions and the SOWFA predictions at time step (a) T, (b) $T + 10$, and (c) $T + 20$. The front and the rear turbine rotors are located at $(0, 0)$m and $(632, 0)$m of the 2D plane respectively.

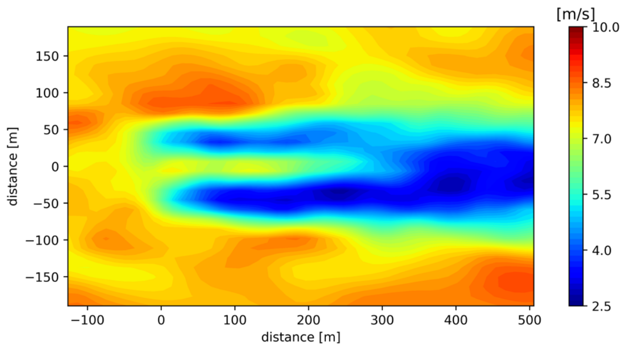
of the developed model, as the model has not encountered the designed yaw patterns (constant yaw and linear yaw change) during training.

A 9-Turbine Test Case

The simulation of a 3×3 wind turbine array is carried out to illustrate the use of the developed model for the fast simulations of large-scale wind farm wakes. The lateral distance between adjacent wind turbines is three rotor diameters and the downstream distance is five rotor diameters. The freestream condition with the average wind speed of 9m/s and FSTI of 6% is used. The turbine yaw angles are kept constant for the simulation time of 300s, with the front turbine yaw angle being 20°, 0°, −20° respectively and the yaw angles of the rest turbines being 0°. The snapshots at 180s, 190s, and 200s are shown in Figure 6.12. As can be seen, both the front turbines' wake deflections and the wake interactions between turbines are captured



(a) $t = 100$s

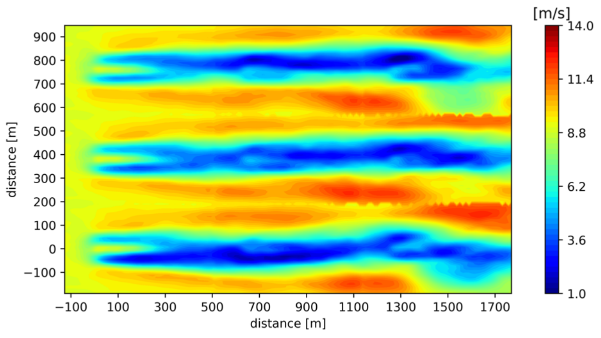


(b) $t = 300$s

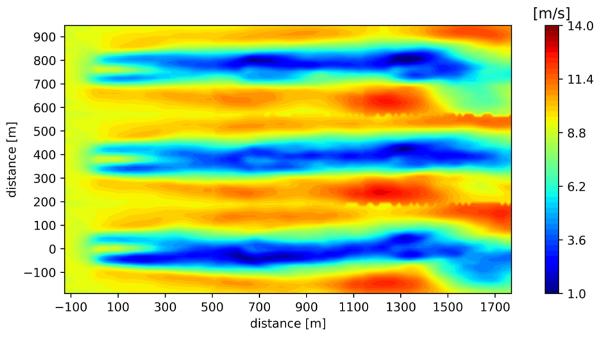
Figure 6.11: The snapshots of the flow field around a single turbine predicted by the POD-LSTM model with designed yaw change, at time steps (a) 100s and (b) 300s. The turbine rotor is located at $(0, 0)$m of the 2D plane.

successfully. However, as can be seen, there are discontinuities in the predicted flow fields at the interface between different rows of wind turbines. This is because the current model only considers the interactions between subdomains through the upstream boundary, which is enough in capturing the main wake interactions. This discontinuity issue may be solved by including all the boundary conditions of each subdomain as the input in the POD-LSTM model. However, the consideration of the interactions at all the subdomain boundaries is not trivial and needs further investigation regarding the generation of the training dataset and the numerical stability. In this thesis, the wake modelling only considers the upstream flow's impact on the flow field in the downstream subdomains.

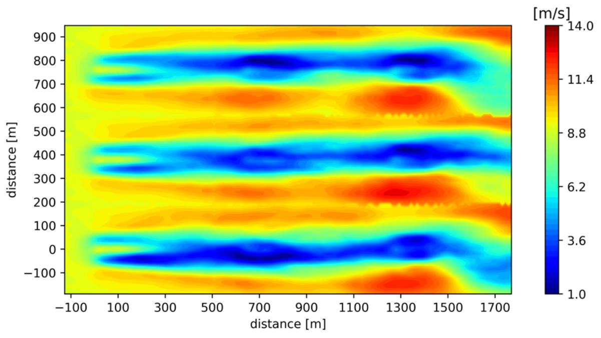
The unsteady flow field visualisation is available online in the supporting materials of the published paper. As can be seen from the video, the POD-LSTM model predictions show similar qualitative flow features seen in the LES of wind farms, such as the wake meandering and the streamwise convection of flow struc-



(a) $t = 180$s



(b) $t = 190$s



(c) $t = 200$s

Figure 6.12: The snapshots of the flow field around a 3×3 wind turbine array predicted by the POD-LSTM model, at time steps (a) 180s, (b) 190s, and (c) 200s. The 9 turbines are located at the grid points of $[0, 632, 1264] \times [0, 379.2, 758.4]$m of the 2D plane.

tures. The simulations by the POD-LSTM model require negligible computational time (several seconds) on a standard desktop, while LES of such system requires tens of thousands of CPU hours on an HPC cluster. This 9-turbine test case demonstrates the full potential of the developed model in the fast simulation, prediction and control design of utility-scale wind farms.

6.5 Conclusions

In this work, a deep learning based surrogate modelling method for distributed unsteady fluid systems was proposed, which was then applied to build a novel data-based dynamic wind farm wake model. A valuable high-fidelity LES database was first generated, which took around 7×10^5 CPU hours using HPC clusters. Based on the generated LES database, the deep learning based dynamic wake model was trained to capture the complex wind farm wake dynamics. The results showed that the developed wake model was able to capture the main unsteady flow features (such as the streamwise convection of flow structures, the wake meandering, the wake's deflection with changing yaw, and the wake interactions between wind turbines) similarly as high-fidelity wake models while running as fast as the low-fidelity static wake models. The model's performance was validated against high-fidelity LES results and the overall prediction error was just 4.8% with respect to the freestream wind speed. After validating the developed wake model, two test cases were carried out, and the results demonstrated that the model was able to capture the yaw effect on turbine wakes and was able to achieve fast simulations of large-scale wind farms. In particular, the results of the 9-turbine test case showed that the developed model was able to predict the unsteady turbine wakes in several seconds on a standard desktop while it requires tens of thousands of CPU hours on an HPC cluster if a high-fidelity model is used. As the existing wake models in the literature are either too time-consuming or unable to capture detailed wake dynamics, the developed model brings a step change in fast and accurate simulations, predictions, and control designs of wind farms. This work also paves the way for developing novel wake models using advanced ML techniques. The proposed surrogate modelling method can also be applied to other distributed fluid systems to build surrogate models based on which optimal designs can be achieved with much less computation cost than based on high-fidelity models. As the proposed approach only considers the interactions between subdomains through the upstream boundary, future works may include further method development to take account of the interactions at all the subdomain interfaces so that flow interactions in the lateral direction (such as wake merging) can be captured as well. In addition, more works are needed to further improve the long-term prediction accuracy of the proposed method.

Future work may also include applying this novel wind farm wake model in wake control in order to reduce wind turbine load, maximise the wind power harvesting, and support the electricity grid. This can be done by either using the developed model as an internal model in the control design or using it as a fast simulation model to design and test control strategies. As the developed model is fast to evaluate and can capture the yaw effect and wake interactions, it can be used to facilitate the exploration of new wake (or wake interaction) patterns through fast simulations. Another possible research direction is to incorporate the 3D wake dynamics in the ML models.

Chapter 7

Spatiotemporal Wind Field Prediction Based on Physics-Informed Deep Learning and LIDAR Measurements

7.1 Introduction

In this chapter, a deep learning based method is proposed for the predictions of spatiotemporal wind field using only LIDAR measurements at sparse locations. Specifically, the NS equations are encoded in the deep NN following the PINNs framework [145] and an observation process is embedded into the NN which maps the full flow state to LIDAR observations. The NN training is then carried out to minimise both the functional loss (which encodes the NS equations) and the measurement loss (which is based on LIDAR observations). After training, the prediction of the spatiotemporal wind field in the whole domain in front of the wind turbine can be achieved.

The method developed in this chapter is different from traditional numerical methods and existing ML-based wind prediction methods as follows: (1) Various numerical models are widely used for wind simulations and the detailed wind field can be obtained by solving the NS equations numerically with properly-defined boundary conditions or the input conditions estimated from measurement data [182]. However, these models are mainly designed for forward simulations of wind flows.

It is extremely challenging to incorporate real-time scattered measurement data in these models because it involves solving the inverse problem, which would require a formidable number of time-consuming simulations to calibrate the model parameters and the input conditions against the measurement data. In contrast, the PINNs framework is specifically designed to incorporate data and PDEs in a unified manner which makes it very powerful in solving inverse problems governed by PDEs. (2) Previous ML-based wind prediction studies e.g. [87, 183, 184] treat ML models as 'black-box' and require the corresponding input and target values for training. Then they can predict the wind patterns which are present in the training dataset. It is worth mentioning that the paper [184] did explore to involve physics in the form of simple analytical relations in the design of the ML on the wind farm modelling, which showed very promising results. However, all these studies followed the traditional supervised ML, thus can not discover the wind patterns that are not present in the measurement data. In summary, the work presented in this chapter, which fuses physics in terms of PDEs and data in the deep learning training process for wind applications for the first time, can achieve the predictions of spatiotemporal wind field in the whole domain based on only LoS LIDAR measurements at sparse spatial locations, which is not achievable by either traditional numerical models or existing ML-based models in the literature.

The method proposed in this chapter is tested and validated using large-scale numerical simulations based on SOWFA [160]. SOWFA is used in this chapter as the high-fidelity numerical experiment platform to simulate the real-world wind flows in the atmospheric boundary layer. The LIDAR measurement and the turbulent wind field are extracted from SOWFA simulations as the model training data and the ground truth (for model validation) respectively.

The main contributions of this chapter are summarised as follows:

(1) The prediction of spatiotemporal wind velocity field in the whole flow domain based on LoS wind speed at only a few sparse locations measured by LIDAR is achieved, which is of great importance for developing advanced approaches for the wind resource assessment and for the monitoring and control of wind turbine/farm. The developed method can achieve: (i) the prediction of flow dynamics over the whole domain of interest, including the spatial locations where no measurements are available; (ii) great performance in wind field estimation, because the spatiotemporal correlations between measurements are taken into account implicitly through NS equations without model reduction; (iii) robust wind estimation in the scenarios of both 'small' and 'big' data, as the issue of overfitting commonly encountered in deep learning is tackled by enforcing the physical constraints.

(2) To the best of the author's knowledge, this is for the first time that physical laws (in terms of PDEs) and data are fused in the training of deep learning models for wind applications. Specifically a deep NN with a large DoF is constructed and then the NS equations (which provide a good description of atmospheric flows) are incorporated directly in the deep NN. After that, the deep NN is trained to minimise the errors from both fitting the LIDAR measurements and enforcing the NS equations. Because the existing wind prediction studies are either purely data-driven [86, 87] or based on low-fidelity/reduced order models [89, 90], they can not take full advantage of both physical laws in terms of PDEs and data.

(3) The developed method is validated through high-fidelity LES wind farm simulations and its robustness is verified under a wide range of scenarios. A short-term wind forecasting is also achieved which does not rely on the Taylor's frozen turbulence hypothesis [185], as the NN learns the dynamics of the evolving wind field from NS equations.

The remaining part of this chapter is organised as follows: the deep learning based method for the spatiotemporal wind field predictions is described in Section 7.2. Then the performance of the proposed method is tested by using an LES wind farm simulator as the experimental platform in Section 7.3, where a wide range of scenarios are considered to verify the robustness of the proposed method. Finally the conclusions are drawn in Section 7.4.

7.2 Methodology

This chapter addresses the problem of predicting the spatiotemporal wind velocity field in the whole flow domain by physics-informed deep learning and LIDAR measurements at sparse spatial locations. The considered LIDAR configuration is illustrated in Figure 7.1(A), where two laser beams in the horizontal hub-height 2D plane (shown as the shaded blue area) are used to measure the LoS wind speed in the laser beam directions at a frequency of 1s at discrete spatial locations along the beams (marked as cross signs in Figure 7.1(A)). The half-angle of the beams is 15°. An example of the wind speed measurements by the left and right beams at these discrete locations during a time period from 0 to T is shown in Figure 7.1(B). The wind field prediction problem thus states as how to predict the spatiotemporal velocity (including both the downwind and crosswind components) field in the domain of interest (i.e. the whole hub-height 2D flow domain coloured in blue in Figure 7.1(A)) from time 0 to T based on only the LoS wind speed measurements at a few sparse locations marked as cross signs in Figure 7.1(A).

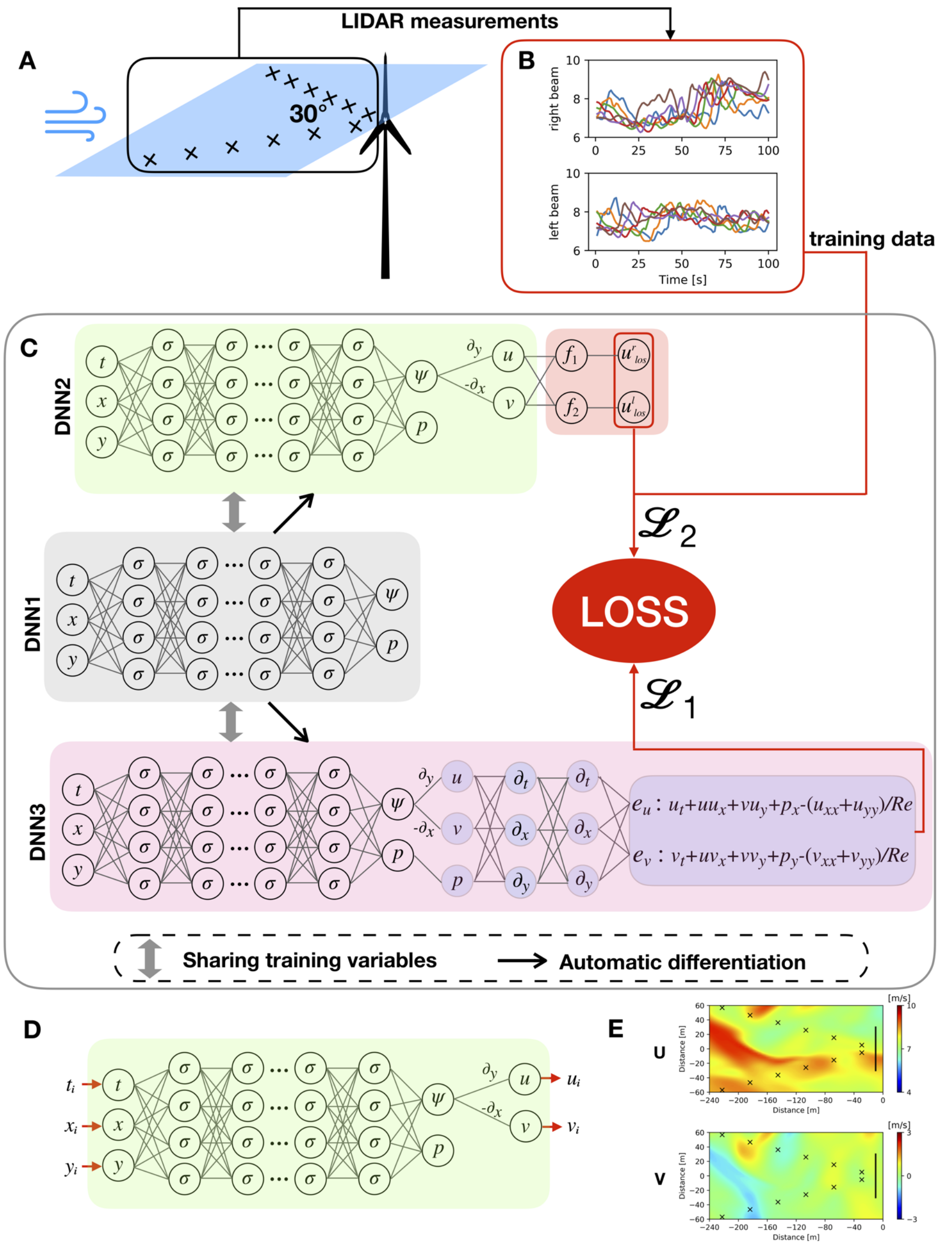

A
LIDAR measurements
30°
B
right beam
left beam
Time [s]
training data
C
DNN2
DNN1
DNN3
$\mathscr{L}_2$
LOSS
$\mathscr{L}_1$
$e_u: u_t+uu_x+vu_y+p_x-(u_{xx}+u_{yy})/Re$
$e_v: v_t+uv_x+vv_y+p_y-(v_{xx}+v_{yy})/Re$
Sharing training variables
Automatic differentiation
D
E
U
V
Distance [m]
[m/s]

Figure 7.1: The flowchart illustrating the proposed spatiotemporal wind field prediction method. (A) LIDAR configuration. (B) The LoS wind speed measured by the left and right laser beams during a certain period. (C) The deep learning model which incorporates the NS equations and LIDAR measurements. (D) The prediction of wind velocity at a given time instant and a given location after NN training. (E) The wind field prediction in the whole domain at a given time instant.

This task is not achievable by using the traditional supervised ML framework with whether simple NN such as multi-layer perceptions or complex NN such as convolution NN and recurrent NN, because the traditional framework requires the information on the whole spatiotemporal wind field as training data, but in reality only the LoS wind speed data at sparse locations is available. In order to reconstruct the spatiotemporal wind field based on only sparse measurement data, this work employs the novel PINNs framework, where the incompressible NS equations, which provides a very good description for many fluid flows such as atmospheric boundary layer flows, are fused with LIDAR data in the training of the deep learning model.

The overall flowchart illustrating the proposed method is shown in Figure 7.1, where the training dataset collection, the deep NN structure and training, and the model prediction are illustrated in Figure 7.1(A-B), Figure 7.1(C), and Figure 7.1(D-E) respectively. The detailed training and prediction process is described in the rest part of this section.

7.2.1 Training Dataset

The training dataset in this work is the LoS wind speed values at sparse spatial locations measured by LIDAR beams. The data measured by the left and right beams are collected separately as the observation process (i.e. the function that maps the flow states to the measurement values) depends on the beam direction. Denote the spatial coordinate of the i^{th} measurement point in the right beam as $[\tilde{x}_i^r, \tilde{y}_i^r]$, the spatial coordinate of the i^{th} measurement point in the left beam as $[\tilde{x}_i^l, \tilde{y}_i^l]$, and the LIDAR measurement values at these coordinates at $\tilde{t}^{th}$ second as $\tilde{u}_{\tilde{x}_i^r,\tilde{y}_i^r,\tilde{t}}$ and $\tilde{u}_{\tilde{x}_i^l,\tilde{y}_i^l,\tilde{t}}$ respectively. All the LIDAR measurements by the right beam during a time period of T seconds are then collected as the data matrix $\tilde{\mathscr{U}}^r$ of shape $[N^r \times T, 4]$, where N^r represents the total number of discrete points in the right beam and each row of $\tilde{\mathscr{U}}^r$ consists of the spatiotemporal coordinate $[\tilde{x}_i^r, \tilde{y}_i^r, \tilde{t}]$ and the corresponding measurement value $\tilde{u}_{\tilde{x}_i^r,\tilde{y}_i^r,\tilde{t}}$. The measurements by the left beam are collected in the same way as $\tilde{\mathscr{U}}^l$ of shape $[N^l \times T, 4]$ with N^l representing the total number of discrete points in the left beam. These data matrices are then nondimensionalised by the characteristic length D, the characteristic time D/U_∞, and the characteristic velocity U_∞, where D represents the turbine rotor diameter and U_∞ represents the average freestream wind speed. The nondimensionalised data matrices, which are the only wind data required for the NN training, are hereby denoted as $\mathscr{U}^r$ and $\mathscr{U}^l$. In order to evaluate the trained ML model, the test dataset is specified as the spatiotemporal flow field in the whole domain in front of the wind turbine during the same time period T. To avoid confusion, it is worth noting that the training and test dataset in this work are totally different, with the former consisting of the LoS wind speed data at sparse locations and the latter consisting of the wind velocity vectors at every locations in the 2D plane in front of the wind turbine, while in the supervised ML they are generally of the same data structure and are usually obtained by dividing the same dataset.

7.2.2 Neural Network Structure

After collecting the training dataset, a fully-connected deep NN is constructed, which is illustrated in shaded grey and denoted as DNN1 in Figure 7.1(C). This deep NN takes the nondimensional spatiotemporal coordinate (i.e. $[t, x, y]$) as the input and returns the nondimensional stream function [186] and the pressure as the output (i.e. $[\psi, p]$). It is used to approximate the mapping between the continuous spatiotemporal coordinate and the corresponding quantities, such that given any time instant t_i and any location $[x_i, y_i]$ the deep NN is trained to return $\psi(t_i, x_i, y_i)$ and $p(t_i, x_i, y_i)$ as the output. However, no data about p and ψ is needed for training this NN because these quantities are just auxiliary quantities used for deriving the velocity and encoding NS equations. This fully-connected NN can be expressed in recursive form as

$$\begin{aligned} &H_0 = [t, x, y], \\ &H_i = \sigma(H_{i-1} \cdot W_i + B_i), 1 \leq i \leq L, \\ &H_L = [\psi, p], \end{aligned} \tag{7.1}$$

where $L + 1$ represents the total number of layers in this deep NN, $\{W_i, 1 \leq i \leq L\}$ and $\{B_i, 1 \leq i \leq L\}$ represent all the training variables in this NN, and σ represents the activation function. The shapes of the weight matrix W_1, $\{W_i, 1 < i < L\}$ and W_L are $[3, N_h]$, $[N_h, N_h]$ and $[N_h, 2]$ respectively, where N_h represents the neuron number of the hidden layers. The shapes of the bias term $\{B_i, 1 \leq i < L\}$ and B_L are $[1, N_h]$ and $[1, 2]$ respectively. The total DoF of this deep NN (i.e. the total number of training variables) can then be calculated as

$$N_{dof} = 3N_h + (L-2)N_hN_h + 2N_h + (L-1)N_h + 2. \tag{7.2}$$

The hyperbolic tangent function is used for all the hidden layers in this work and the activation is not applied for the output layer. It is worth mentioning that L is typically very large. Thus the NN is termed ‘deep’ and it is this deep structure that enhances the ability of the NN in capturing very complex nonlinear dynamics. Fully-connected NN with such large DoF is generally not used in traditional supervised ML as the issue of overfitting is hard to tackle, while it can be used in this work as overfitting is constrained by the encoded PDEs in the physics-informed deep learning framework.

After constructing this deep NN, a second NN, as shown in shaded green and denoted as DNN2 in Figure 7.1(C), is constructed which takes the nondimensional spatiotemporal coordinate (i.e. $[t, x, y]$) as the input and returns the nondimensional downwind velocity u, crosswind velocity v and pressure p as the output. This second NN is derived based on the first NN, by taking the derivative of the NN output of the first NN with respect to the NN input using automatic differentiation. Thus it shares the same training variables with the first NN and no new training variables are created. The output of this second NN is derived by

$$\partial\psi/\partial y = u, -\partial\psi/\partial x = v. \tag{7.3}$$

Therefore, the continuity equation

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{7.4}$$

is satisfied automatically. As LIDAR can only measure the LoS wind speed, no data about u or v is available for the NN training. To train the NN with the LIDAR data, an observation process that maps the flow state (i.e. $[u, v, p]$) to LIDAR observations (i.e. $[u^r_{los}, u^l_{los}]$) is embedded to the second NN, which is shaded in light red in Figure 7.1(C). The functions f_1 and f_2 in Figure 7.1(C) represent the right and left beam observation processes respectively, which are expressed as

$$f_1(u, v) = ucos(15°) - vsin(15°) \tag{7.5}$$
$$f_2(u, v) = ucos(-15°) - vsin(-15°) \tag{7.6}$$

The inclusion of more LIDAR beams and/or other types of flow sensors such as pressure sensors, is straightforward by embedding the corresponding observation processes in this second NN. In this work, only f_1 and f_2 are embedded as only the measurements from the left and right LIDAR beams are used for the NN training.

Next, a third NN, as shown in shaded pink and denoted as DNN3 in Figure 7.1(C), is constructed based on the second NN by taking the derivative of the NN output with respect to the NN input using automatic differentiation. This NN, which is the physics-informed part, takes the nondimensional spatiotemporal coordinate (i.e. $[t, x, y]$) as the input and returns the NS residue terms (i.e. $[e_u, e_v]$)

as the output. The NS residue terms are defined by reformulating the following nondimensional 2D NS equations

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{7.7}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}), \tag{7.8}$$

as

$$e_u = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \frac{1}{Re}(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{7.9}$$

$$e_v = \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \frac{1}{Re}(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}). \tag{7.10}$$

Here $Re = (U_\infty D)/\nu$ with ν representing the kinematic viscosity of air.

In summary, there are three deep NNs constructed in the whole NN structure. However, they are essentially just one NN in terms of training, as all of them share exactly the same training variables and only one loss function will be defined to train these training variables. The NN training and prediction details are described in the next subsection.

7.2.3 NN Training and Prediction

The deep NN is trained to minimise the loss arising from both the NS residue terms and LIDAR observations. The loss arising from NS residue terms is defined as

$$\mathscr{L}_1 = \frac{1}{N_{ns}}\sum_{i=1}^{N_{ns}} |e_u(x_i^{ns}, y_i^{ns}, t_i^{ns})|^2 + \frac{1}{N_{ns}}\sum_{i=1}^{N_{ns}} |e_v(x_i^{ns}, y_i^{ns}, t_i^{ns})|^2 \tag{7.11}$$

where $\{[x_i^{ns}, y_i^{ns}, t_i^{ns}], 1 \le i \le N_{ns}\}$ is a batch of test points which is fed to the deep NN to evaluate e_u and e_v. In practice, a set of test points in the spatiotemporal domain of interest are first collected in a data matrix $\tilde{\mathscr{D}}$ of shape $[N_{test}, 3]$, where N_{test} is the total number of test points and each row of $\tilde{\mathscr{D}}$ contains one spatiotemporal coordinate. Then $\{[x_i^{ns}, y_i^{ns}, t_i^{ns}], 1 \le i \le N_{ns}\}$ is generated by randomly sampling from the data matrix $\mathscr{D}$, which is the data matrix $\tilde{\mathscr{D}}$ nondimensionalised by the characteristic scale D and D/U_∞. In this work, a $81 \times 41 \times 101$ uniform grid points in the domain $[-240, 0]m \times [-60, 60]m \times [0, 100]$s are used to generate $\tilde{\mathscr{D}}$.

The loss arising from LIDAR observations is defined as

$$\begin{aligned}\mathscr{L}_2 =& \frac{1}{N_{d_1}} \sum_{i=1}^{N_{d_1}} |u_{los}^r(x_i^r, y_i^r, t_i^r) - u_i^r|^2 \\ &+ \frac{1}{N_{d_2}} \sum_{i=1}^{N_{d_2}} |u_{los}^l(x_i^l, y_i^l, t_i^l) - u_i^l|^2 \end{aligned} \tag{7.12}$$

where $\{[x_i^r, y_i^r, t_i^r, u_i^r], 1 \leq i \leq N_{d_1}\}$ is a batch of right beam measurement data which are randomly-sampled from the matrix $\mathscr{U}^r$, and $\{[x_i^l, y_i^l, t_i^l, u_i^l], 1 \leq i \leq N_{d_2}\}$ is a batch of left beam measurement data which are randomly-sampled from the matrix $\mathscr{U}^l$.

Finally, the deep NN is trained to minimise the total loss defined as

$$\mathscr{L} = \mathscr{L}_1 + \mathscr{L}_2, \tag{7.13}$$

by feeding the data batches $\{[x_i^{ns}, y_i^{ns}, t_i^{ns}], 1 \leq i \leq N_{ns}\}$, $\{[x_i^r, y_i^r, t_i^r, u_i^r], 1 \leq i \leq N_{d_1}\}$ and $\{[x_i^l, y_i^l, t_i^l, u_i^l], 1 \leq i \leq N_{d_2}\}$ to the NN simultaneously during each training iteration. The Adam optimisation algorithm [158] is employed in this work for the NN training.

After training, the spatiotemporal flow field in the whole flow domain, including both the downwind and crosswind velocity components, can be predicted. Specifically, given any time coordinate t_i and space coordinate $[x_i, y_i]$, the corresponding wind speed u_i and v_i can be predicted through the second NN, as shown in Figure 7.1(D). The prediction of the flow field in the whole domain at a given time instant, as shown in Figure 7.1(E), can be achieved by propagating the given time coordinate and the space coordinates of all the mesh points in the domain of interest through the second NN. The whole training and prediction procedure is summarised as Algorithm 6. In addition, after training, the short-term flow field forecasting can also be carried out in a straightforward manner by simply specifying the time t in Line 11 of Algorithm 6 as the time coordinate of interest in the future.

7.3 Numerical Results

The wind field prediction method developed above is evaluated in this section, by using the LES wind farm simulator SOWFA as the experimental platform. The simulation setups are described first, then the spatiotemporal flow field prediction is carried out and the results of the time-varying velocity fields are validated with

Algorithm 6 The NN training and prediction procedure

```
1: % The NN training
2: Load LIDAR measurement data, i.e. $\mathscr{U}^r$ and $\mathscr{U}^l$.
3: Load the time and space coordinates of NS test points $\mathscr{D}$.
4: Set training iteration number $N_{iter}$.
5: Set the batch size $N_{ns}$, $N_{d_1}$, $N_{d_2}$.
6: for i in $[1, 2, ..., N_{iter}]$ do
7:     Generate data batches of size $N_{d_1}$, $N_{d_2}$ and $N_{ns}$ from $\mathscr{U}^r$, $\mathscr{U}^l$ and $\mathscr{D}$ respectively.
8:     Train the deep NN by feeding these data batches to minimise the total loss $\mathscr{L}$.
9: end for

10: % The NN prediction
11: Set any time coordinate of interest $t$.
12: Set a mesh of dimension $N_{mesh}$ for the whole 2D domain.
13: for i in [1,2,...,$N_{mesh}$] do
14:     Set $[x_i, y_i]$ by the location of the $i^{th}$ mesh point.
15:     Propagate $[x_i, y_i, t]$ through DNN2 to predict $u$ and $v$ at the $i^{th}$ mesh point.
16: end for
17: The wind field in the whole domain at time $t$ is obtained by combining the $u$ and $v$ predictions at all the mesh points.
```

the corresponding true values (i.e. the SOWFA simulation results).

7.3.1 Simulation Setups

The LES wind farm solver SOWFA is employed here to simulate the turbulent atmospheric boundary layer. For the mesh generation, as suggested by [35], a uniform mesh of size $12 \times 12 \times 12$m is used in the whole simulation domain of size $3000 \times 3000 \times 1000$m, which is illustrated in Figure 7.2. The total number of cells is about 5.2×10^6. 400s simulations are carried out with a time step of 0.02s. From the last 100s simulations, the left and right LIDAR measurements are collected, and the corresponding wind field data at turbine hub-height is recorded for validation. In particular, the LIDAR measurement process is simulated by extracting the velocity vectors at the corresponding spatial locations, and then projecting them onto the LIDAR beam directions to obtain the LoS wind speed measurements. The turbine rotor dynamics is excluded in this work, similarly as previous studies [89, 90]. It is worth mentioning that the rotor's blockage effects have impacts on the flow field in the vicinity of the wind turbine [187, 188], but the impacts become negligible in the freestream flow further upstream which is the main interested region for this work.

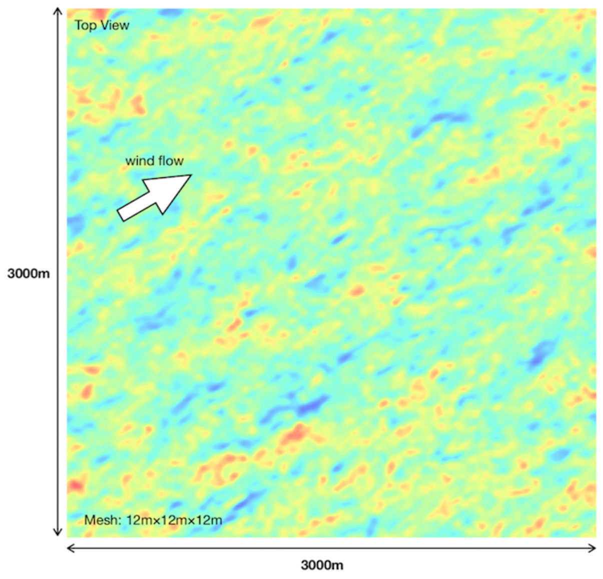


Figure 7.2: A top view of the simulation domain at the turbine hub height. The contour shows the instantaneous flow velocity magnitude.

The simulation in this work is carried out in local HPC clusters, which takes around 2 hours' computational time using 256 processors.

7.3.2 Performance Evaluation

A baseline case is used here to test the performance of the proposed method. For the turbulent atmospheric boundary layer simulations, an average freestream wind speed of 8m/s with an FSTI level of 6% is considered. For the LIDAR configurations, the range of the LIDAR beams is 220m and the distance between discrete measurement points is 20m. There are a total of 11 spatial measurement points per LIDAR beams. The LIDAR measurement is carried out every second during the whole period of 100 seconds and the measurement noise is excluded. Since the wind turbine usually operates with a yaw angle equal to 0°, the LIDAR look direction is set as the mean wind direction in this baseline case.

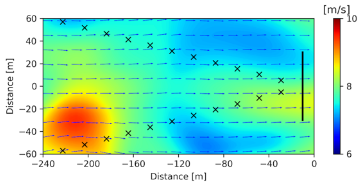
There are still some hyper-parameters in the NN structure and the NN training procedure to be determined. The tuning of the hyper-parameters is carried out by trying a set of configurations and comparing their training losses. The hyper-parameters' values used in this work are given in Table 7.1. As can be seen, the final NN used in this work has a total of 12 layers $(L+1)$ and the neuron numbers of the hidden layers are 128. This results in a total DoF of 149378. This deep

L	N_h	N_{ns}	N_{d_1}	N_{d_2}	lr
11	128	1000	1100	1100	10^{-4}

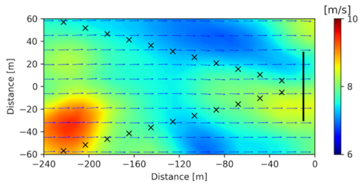
Table 7.1: The hyper-parameters in the NN structure and the NN training procedure. Here lr represents the learning rate.

structure with such a large DoF enables the NN to accurately approximate complex nonlinear PDE systems such as the NS systems in this work. The further increase of the layer number and the neuron number is tested, which has little impact on the NN's performance. Thus the parameters given in Table 7.1 are used. The NN training is carried out using the NVIDIA Tesla K80 GPU in this work with each training iteration requiring about 0.17s. After training, the prediction of the flow field at any time instant of interest requires about 0.012s. These demonstrate that the proposed method can meet real-time control requirement by pre-training and online updating. It is worth noting that training schemes based on transfer learning could possibly decrease the computation time, which needs further investigations and is outside the scope of the current work.

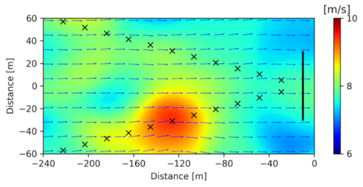
After the NN training, the unsteady velocity field during the considered period of 100s is predicted by the deep NN. Three predicted snapshots, at time $t = 50$s, $t = 60$s and $t = 70$s, along with the corresponding true snapshots (i.e. the snapshots obtained by SOWFA), are shown in Figure 7.3. The corresponding error distributions are shown in Figure 7.4. As can be seen, all the predicted snapshots agree with the true snapshots very well (the quantitative evaluation of the accuracy is given further in Table 7.2). The wind direction and magnitude have been well resolved (Cyclops' dilemma), which is achieved because the correlations between the LoS wind speed measured at different locations are taken into account implicitly through NS residue terms in the NN training procedure. Also, the downstream convection of flow structures in the incoming wind is clearly captured. As shown from the predicted flow fields in Figure 7.3(a, c, e), a high-speed flow structure enters the considered flow domain from the left at $t = 50$s, travels to the middle at $t = 60$s, and hits the wind turbine at the right side of the domain at $t = 70$s. This successful identification of the flow structure and its downstream convection are of great interest. For example it can be used for wind turbine control to mitigate the structural loads. In [77], it was shown that significant wind turbine blade load reduction was achievable by taking the coherent flow structures into account in the wind turbine control design. The paper [77] assumed that the coherent structures were known



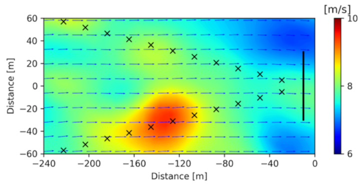
(a) $t = 50$s, prediction



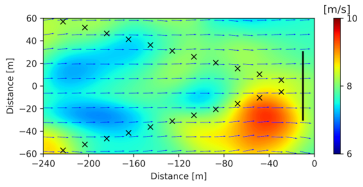
(b) $t = 50$s, true



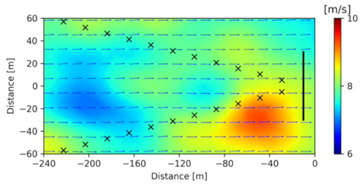
(c) $t = 60$s, prediction



(d) $t = 60$s, true



(e) $t = 70$s, prediction



(f) $t = 70$s, true

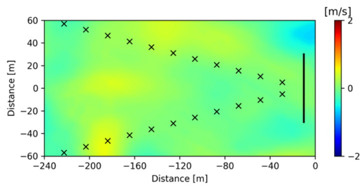
Figure 7.3: The velocity field predicted by the proposed method at the baseline case, at time (a) $t = 50$s, (c) $t = 60$s and (e) $t = 70$s. The corresponding true values are also shown for comparisons (b, d, f).

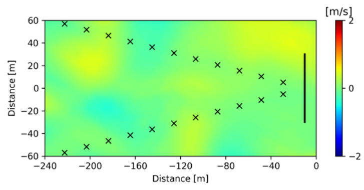
and fully measurable, and pointed out that the prediction of the detailed incoming wind information would play an important role in the level of load mitigation. Therefore, the prediction results in this work fill the research gap by providing an effective way for detailed flow predictions and flow structure detection.

The unsteady wind field visualisation is available online in the supporting materials of the published paper, including both the prediction results and the true results given by SOWFA. As shown in the video, the unsteady flow details such as the convections of high-speed/low-speed flow structures, are predicted accurately, which demonstrates the great performance of the proposed prediction method.

To further quantify the accuracy of the proposed method, the mean value of the root mean-squared errors (MRMSE) between the predicted and the true wind speed fields during the whole time period is given in Table 7.2, which is defined as

$$\epsilon_u = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{N_{test}}\sum_{i=1}^{N_{test}}(u^*_{x_i,y_i,t} - \hat{u}_{x_i,y_i,t})^2}, \tag{7.14}$$


(a) $t = 50$s


(b) $t = 60$s

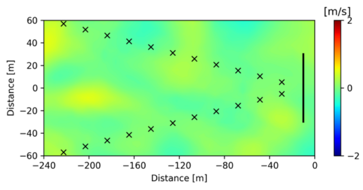

(c) $t = 70$s

Figure 7.4: The difference between the velocity field predicted by the proposed method and the corresponding true values, at the baseline case.

where the total time T is 100, the total number of test points N_{test} is 3321, $\{[x_i, y_i], 1 \leq i \leq N_{test}\}$ is the 81×41 uniform-grid test points in the considered domain, and $\hat{u}_{x_i,y_i,t}$ and $u^*_{x_i,y_i,t}$ represent the corresponding wind speed predictions and true values. Similarly, the MRMSE between the predicted and true wind direction fields is defined as

$$\epsilon_\gamma = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{N_{test}}\sum_{i=1}^{N_{test}}(\gamma^*_{x_i,y_i,t} - \hat{\gamma}_{x_i,y_i,t})^2}, \tag{7.15}$$

where $\hat{\gamma}_{x_i,y_i,t}$ and $\gamma^*_{x_i,y_i,t}$ represent the corresponding wind direction predictions and true values. As shown in Table 7.2, the prediction performance is quite satisfactory. The MRMSE is just 7.0% of the freestream wind speed range at this baseline case.

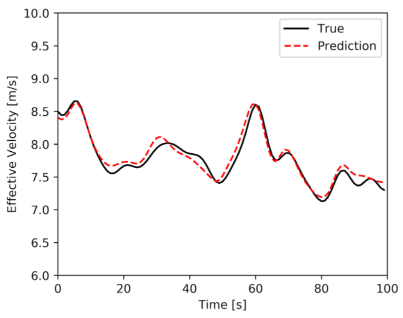
The following part demonstrates the potential use of the proposed prediction method for wind turbine control. First the effective wind speed can be extracted from the predicted spatiotemporal wind field, which is defined as the wind speed averaged over the rotor plane and calculated by

$$\bar{U}_{x,t} = \frac{1}{N_y}\sum_{i=1}^{N_y}\hat{u}_{x,y_i,t}, \tag{7.16}$$

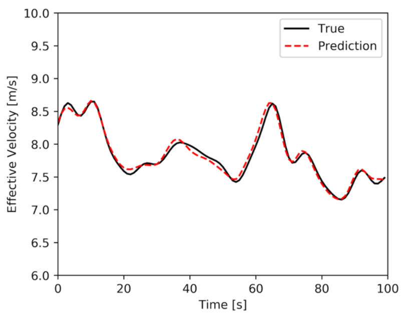
where $\{[x, y_i], 1 \leq i \leq N_y\}$ is a set of spatial points at a fixed distance before the turbine location and uniformly distributed from $-D/2$ to $D/2$ in the spanwise direction. Figure 7.5 shows the effective wind speed averaged over y direction at

Case	Quantity (units)	Range	MRMSE
(A)	Magnitude (m/s)	[6.71, 9.52]	0.198
	Direction ($^\circ$)	[-6.03, 8.28]	2.77
(B1)	Magnitude (m/s)	[6.71, 9.52]	0.208
	Direction ($^\circ$)	[-6.03, 8.28]	2.75
(B2)	Magnitude (m/s)	[6.71, 9.52]	0.236
	Direction ($^\circ$)	[-6.03, 8.28]	3.32
(B3)	Magnitude (m/s)	[6.71, 9.52]	0.387
	Direction ($^\circ$)	[-6.03, 8.28]	3.73
(B4)	Magnitude (m/s)	[6.71, 9.52]	0.523
	Direction ($^\circ$)	[-6.03, 8.28]	4.35
(C)	Magnitude (m/s)	[6.71, 9.52]	0.212
	Direction ($^\circ$)	[-6.03, 8.28]	2.85
(D)	Magnitude (m/s)	[6.71, 9.52]	0.222
	Direction ($^\circ$)	[-6.03, 8.28]	2.66
(E)	Magnitude (m/s)	[6.70, 9.73]	0.281
	Direction ($^\circ$)	[11.4, 27.8]	2.46
(F)	Magnitude (m/s)	[6.71, 8.96]	0.204
	Direction ($^\circ$)	[-6.37, 6.13]	2.69

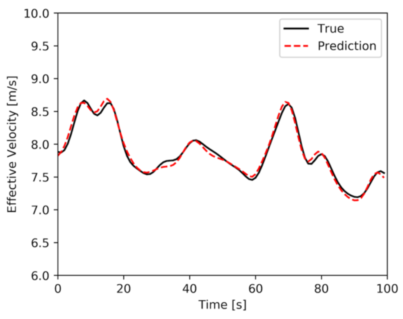
Table 7.2: The MRMSE between the predicted and the true flow fields during the whole time period, for all the scenarios considered in this work, including (A) the baseline case, (B1-B4) LIDAR measurements with various levels of noise, (C) half spatial resolution, (D) half temporal resolution, (E) 20° LIDAR look direction, and (F) FSTI level of 1%.



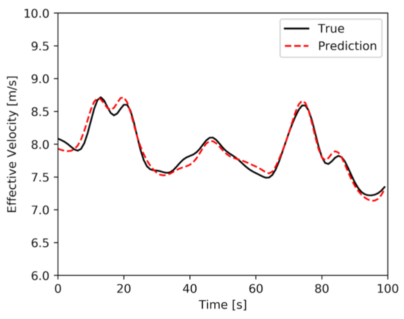
(a) x = -130m



(b) x = -90m



(c) x = -50m

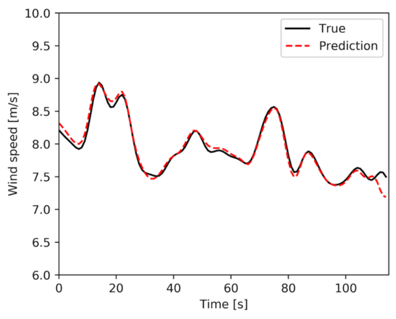


(d) x = -10m

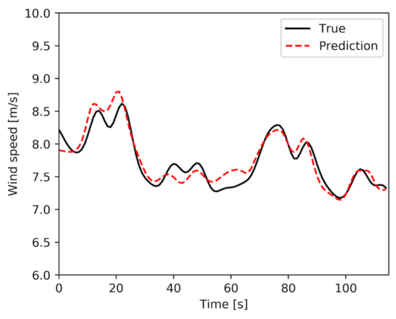
Figure 7.5: The effective wind speed predicted by the proposed method for the baseline case at distances of 120m, 80m, 40m, and 0m before the turbine location. The corresponding true values are also shown for comparisons.

$x = -130$m, $x = -90$m, $x = -50$m and $x = -10$m, which correspond to 120m, 80m, 40m and 0m before the turbine location respectively. The corresponding true values extracted from SOWFA results are also shown. The RMSEs between the predictions and true values are calculated and given in Table 7.3. As can be seen, the predicted effective wind speed matches with its true value very well. It is worth noting that the accurate prediction of the effective wind speed is not unexpected as it is calculated based on the accurately-predicted spatiotemporal wind information. This can help wind turbine control e.g. on power regulation and load reduction.

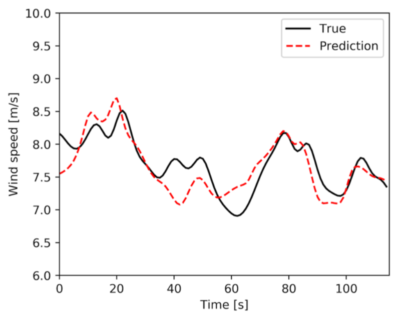
Second, the proposed method can predict the instantaneous wind speed at various turbine locations. As shown in Figure 7.6, three spanwise locations, including 0m, 15m, and 30m, are considered, which correspond to the turbine blade root, 1/2 chord length, and turbine blade tip locations. The true values are also shown in Figure 7.6 for comparisons. The corresponding RMSEs between the predictions and true values are also calculated and given in Table 7.4. As can be seen, the



(a) y = 0m



(b) y = 15m



(c) y = 30m

Figure 7.6: The instantaneous wind speed at turbine location predicted by the proposed method at the baseline case, at spanwise locations of 0m, 15m, and 30m respectively. The corresponding true values are also shown for comparisons.

predicted instantaneous wind speed matches with its true value quite well. This illustrates the great potential of the proposed prediction method in the control of smart rotors [189]. It is worth noting that the wind speed of the onset flow relative to the turbine blade, which is responsible for the turbine loading, is closely related to the instantaneous wind speed shown in Figure 7.6. However, it is not investigated here as the wind field prediction in this chapter is limited to two-dimensional space. More results are given in the next chapter where the full three-dimensional flow field is predicted.

Last but not least, the proposed method can achieve short-term wind forecasting. The extrapolation of the proposed method to future time instants is examined. In particular, the time coordinates from 100s to 115s are fed to the deep NN for predicting the 15-second ahead preview flow information. In order to test the proposed method's performance, another 15s SOWFA simulations are carried out and the wind field data are recorded. The prediction and the corresponding true

Case	$x = -130m$	$x = -90m$	$x = -50m$	$x = -10m$
(A)	0.086	0.055	0.052	0.079
(B1)	0.119	0.083	0.080	0.118
(B2)	0.182	0.137	0.092	0.128
(B3)	0.180	0.138	0.137	0.144
(B4)	0.289	0.200	0.195	0.256
(C)	0.113	0.063	0.069	0.125
(D)	0.102	0.061	0.063	0.144
(E)	0.135	0.091	0.061	0.086
(F)	0.117	0.069	0.038	0.086

Table 7.3: The RMSEs between the predicted and the true effective wind speed at several streamwise locations, for all the scenarios considered in this work, including (A) the baseline case, (B1-B4) LIDAR measurements with various levels of noise, (C) half spatial resolution, (D) half temporal resolution, (E) 20° LIDAR look direction, and (F) FSTI level of 1%.

Case	$y = 0m$	$y = 15m$	$y = 30m$
(A)	0.051	0.148	0.285
(B1)	0.151	0.162	0.264
(B2)	0.175	0.169	0.223
(B3)	0.244	0.313	0.773
(B4)	0.401	0.512	0.748
(C)	0.092	0.199	0.327
(D)	0.084	0.202	0.359
(E)	0.082	0.099	0.136
(F)	0.056	0.168	0.222

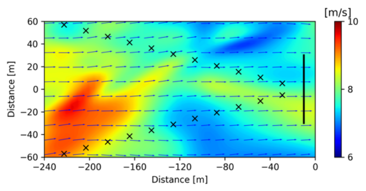
Table 7.4: The RMSEs between the predicted and the true instantaneous wind speed at several turbine locations, for all the scenarios considered in this work, including (A) the baseline case, (B1-B4) LIDAR measurements with various levels of noise, (C) half spatial resolution, (D) half temporal resolution, (E) 20° LIDAR look direction, and (F) FSTI level of 1%.

results from 100s to 115s are included in Figure 7.6. As can be seen, the overall instantaneous wind speed is predicted at satisfactory accuracy. This is because the deep NN learns the dynamics of the evolving wind field from NS equations during the training, and the learnt dynamics is retained which enables the deep NN for short-term wind forecasting without using Taylor's frozen turbulence hypothesis [185]. As the ML model in this work is continuous in time, any future time coordinate can be fed into the NN for prediction. Thus it avoids the tedious tuning of time steps, time horizons and single-step/multiple-step settings in discrete-time models. However, it is worth mentioning that as in all other wind prediction models, the prediction time horizon is still limited by the correlations between the data used for predictions and the quantities to be predicted.

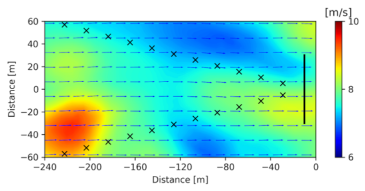
7.3.3 Sensitivity Analysis

The robustness of the proposed method is further verified by considering a wide range of scenarios including LIDAR measurements with various levels of noise and under different LIDAR spatial/temporal resolutions, different LIDAR look directions and different FSTI levels. The prediction accuracy for the whole flow field, the effective wind speed and the instantaneous wind speed at specific locations is given in Table 7.2, 7.3, and 7.4 for all the considered scenarios.

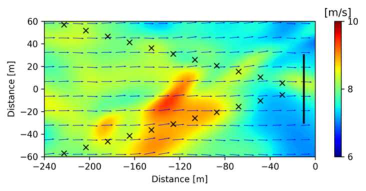
Since LIDAR measurements are subject to various error sources such as range weighting, the measurement noise must be considered in real-world applications. Here, the spatiotemporal wind field reconstruction from noisy LIDAR measurements is investigated, where random noise is added to the LoS wind speed value measured by the LIDAR at each measurement location at each time instant. The noise is drawn from the range $[-e, e]$ uniformly and independently, where a set of values of e are considered including 0.025m/s, 0.05m/s, 0.1m/s, and 0.2m/s. These cases are denoted as Case B1, Case B2, Case B3, and Case B4 respectively. For each case, the deep NN is trained with the noisy measurement data and then used for predicting the spatiotemporal wind field. The prediction MRMSEs for all the four cases are given in Table 7.2. As expected, the prediction becomes less accurate when the measurement noise increases. However, for all the cases, the errors remain quite small compared to the wind speed range, which demonstrates the method's robustness against noisy measurements. As suggested in [89], Case B3 here represents the typical noise of the commercially available pulsed LIDAR instruments. Similar measurement accuracy has been reported in the product guide of the continuous wave LIDAR devices by ZXLidars. The predicted spatiotemporal flow field for Case B3 is shown in Figure 7.7 and 7.8. As can be seen, the unsteady flow field is successfully predicted with main



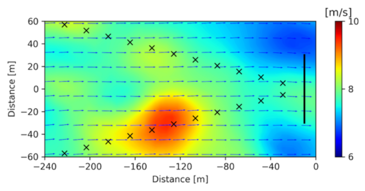
(a) $t = 50\text{s}$, prediction



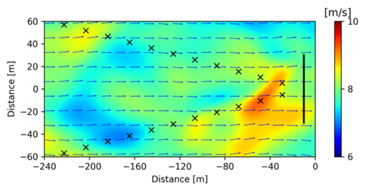
(b) $t = 50\text{s}$, true



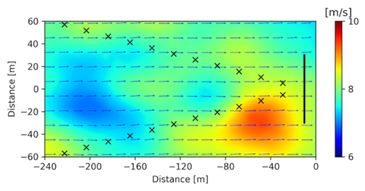
(c) $t = 60\text{s}$, prediction



(d) $t = 60\text{s}$, true



(e) $t = 70\text{s}$, prediction

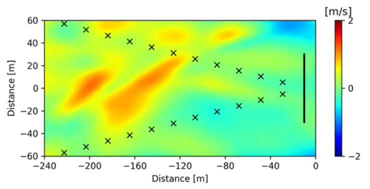


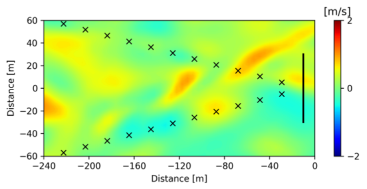
(f) $t = 70\text{s}$, true

Figure 7.7: The velocity field predicted by the proposed method for the case where the measurement noise is of typical commercial LIDAR devices, at time (a) $t = 50\text{s}$, (c) $t = 60\text{s}$ and (e) $t = 70\text{s}$. The corresponding true values are also shown for comparisons (b, d, f).

flow structures identified correctly and the MRMSE remains quite small as shown in Table 7.2, which indicate that the proposed method works well with commercial LIDAR devices. Furthermore, it is worth noting that new methods such as Bayesian PINNs [150] are under active development, which might offer new opportunities for the predictions with lower-quality measurements.

To further illustrate the proposed method's great performance in predicting the spatiotemporal information from very sparse measurements, the cases with only half spatial/temporal LIDAR measurement resolutions are investigated. For the case with half spatial resolution, the distance between the measurement points is set as 40m and only 6 measurement points per LIDAR beam are used in the prediction. For the case with half temporal resolution, the measurement frequency of the LIDAR beams is set as 2s. The prediction results are given in Figure 7.9, 7.10, 7.11, and 7.12. As can be seen, the flow field predictions are similar as in the baseline case, which demonstrates the method's robustness with various spatiotemporal


(a) $t = 50$s


(b) $t = 60$s

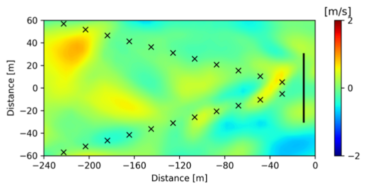
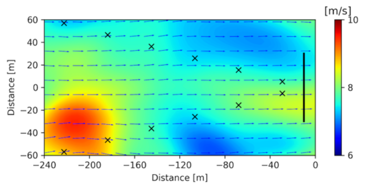
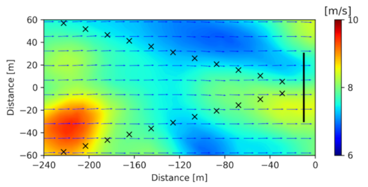

(c) $t = 70$s

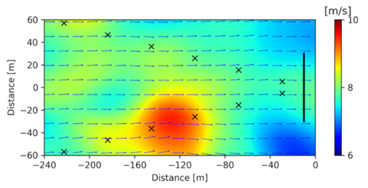
Figure 7.8: The difference between the velocity field predicted by the proposed method and the corresponding true values, for the case where the measurement noise is of typical commercial LIDAR devices.

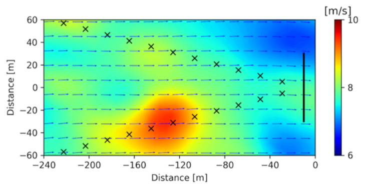
measurement resolutions. In addition, it is worth noting that the predictions here are based on the measurements at as few as 6 spatial locations per LIDAR beam, while most existing works which follow the PINNs framework have used a much larger set of measurement points. Thus the results here also demonstrate the full potential of PINNs in handling the situations of very sparse data. In addition, the prediction accuracy for the half-spatial case and half-temporal case is just slightly lower than the baseline case, which indicates the existence of data redundancy in the space and time domain for the baseline case. This problem might be solved by designing novel data acquisition strategies for the PINNs to optimally place the measurement locations, which can further increase the data quality and/or reduce the data redundancy. Such strategies might lead to the design of the optimal LIDAR configurations in wind industry, such as optimal half-angles, resolutions, scanning patterns, and even the optimal coordination among LIDAR beams. This task, however, is not trivial and requires extensive studies on the problem formulation and the method development, thus is out of the scope of the current work.

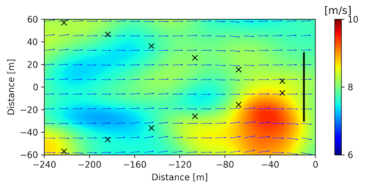
In addition, as LIDAR can only measure the LoS wind speed, the wind direction needs to be estimated (Cyclops' dilemma). A different LIDAR look direction is also considered here to further demonstrate the proposed method's ability in identifying the incoming wind direction. The incoming wind's mean direction is set as 20° from the turbine facing direction. The results are given in Figure 7.13 and 7.14. As can be seen, the incoming wind direction is correctly identified, overcoming the


(a) $t = 50$s, prediction


(b) $t = 50$s, true


(c) $t = 60$s, prediction


(d) $t = 60$s, true


(e) $t = 70$s, prediction

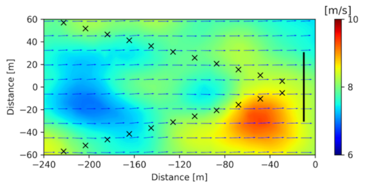
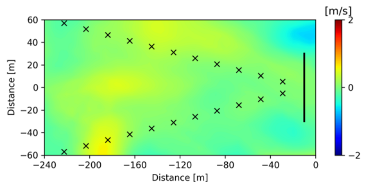
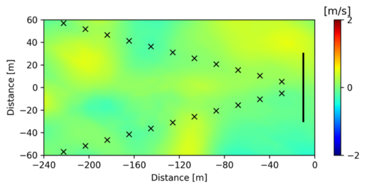

(f) $t = 70$s, true

Figure 7.9: The velocity field predicted by the proposed method for the case with half spatial measurement resolution, at time (a) $t = 50$s, (c) $t = 60$s and (e) $t = 70$s. The corresponding true values are also shown for comparisons (b, d, f).


(a) $t = 50$s


(b) $t = 60$s

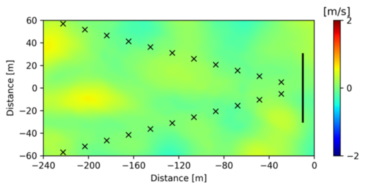
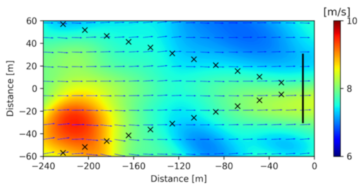
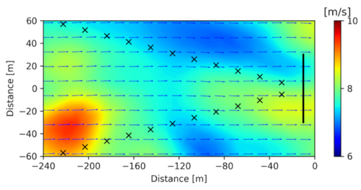

(c) $t = 70$s

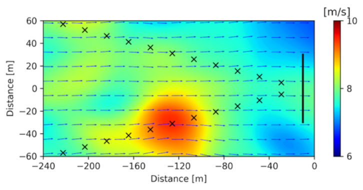
Figure 7.10: The difference between the velocity field predicted by the proposed method and the corresponding true values, for the case with half spatial measurement resolution.



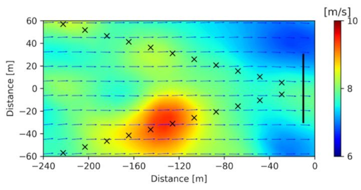
(a) $t = 50$s, prediction



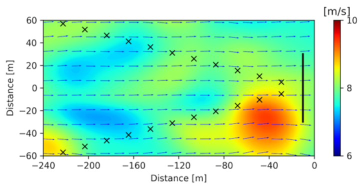
(b) $t = 50$s, true



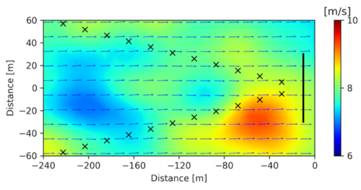
(c) $t = 60$s, prediction



(d) $t = 60$s, true

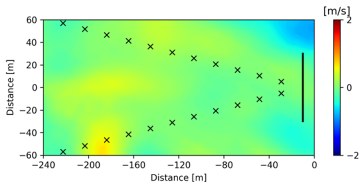


(e) $t = 70$s, prediction

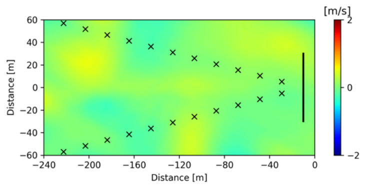


(f) $t = 70$s, true

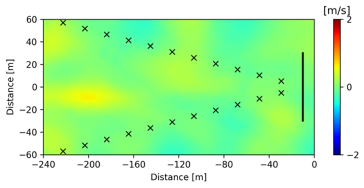
Figure 7.11: The velocity field predicted by the proposed method for the case with half temporal measurement resolution, at time (a) $t = 50$s, (c) $t = 60$s and (e) $t = 70$s. The corresponding true values are also shown for comparisons (b, d, f).



(a) $t = 50$s

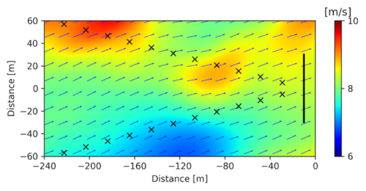


(b) $t = 60$s

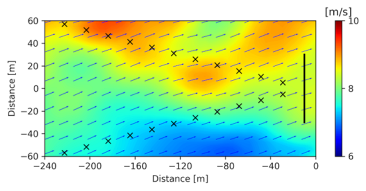


(c) $t = 70$s

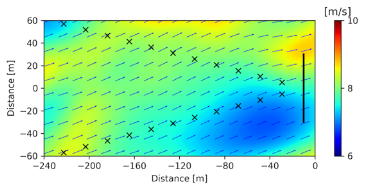
Figure 7.12: The difference between the velocity field predicted by the proposed method and the corresponding true values, for the case with half temporal measurement resolution.



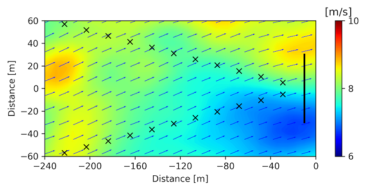
(a) $t = 50$s, prediction



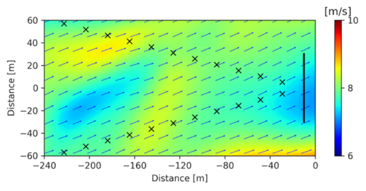
(b) $t = 50$s, true



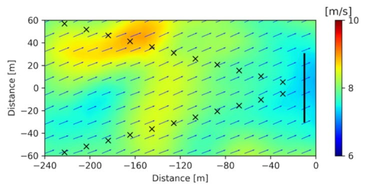
(c) $t = 60$s, prediction



(d) $t = 60$s, true

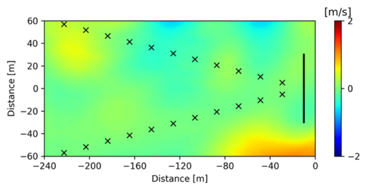


(e) $t = 70$s, prediction

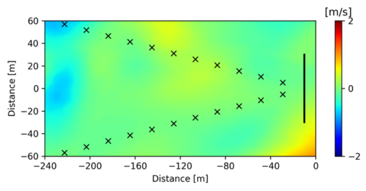


(f) $t = 70$s, true

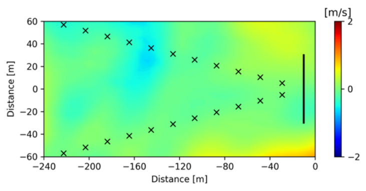
Figure 7.13: The velocity field predicted by the proposed method for the case where the LIDAR look direction is 20°, at time (a) $t = 50$s, (c) $t = 60$s and (e) $t = 70$s. The corresponding true values are also shown for comparisons (b, d, f).



(a) $t = 50$s

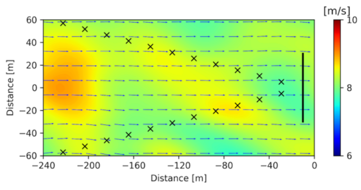


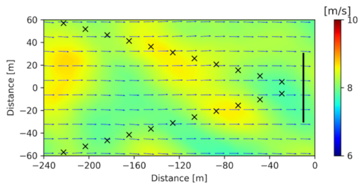
(b) $t = 60$s

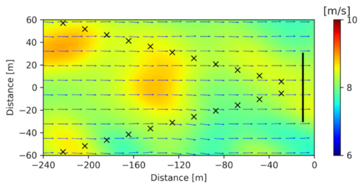


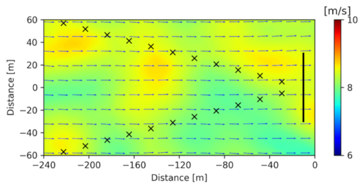
(c) $t = 70$s

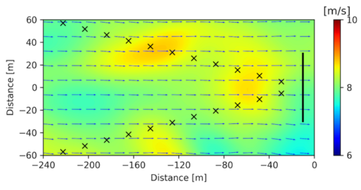
Figure 7.14: The difference between the velocity field predicted by the proposed method and the corresponding true values, for the case where the LIDAR look direction is 20°.


(a) $t = 50$s, prediction


(b) $t = 50$s, true


(c) $t = 60$s, prediction


(d) $t = 60$s, true


(e) $t = 70$s, prediction

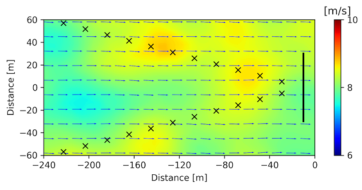
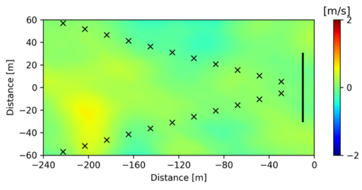
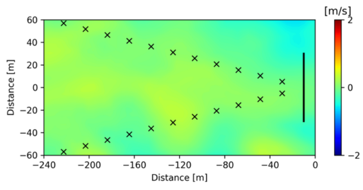

(f) $t = 70$s, true

Figure 7.15: The velocity field predicted by the proposed method for the case where the FSTI level is 1% , at time (a) $t = 50$s, (c) $t = 60$s and (e) $t = 70$s. The corresponding true values are also shown for comparisons (b, d, f).


(a) $t = 50$s


(b) $t = 60$s

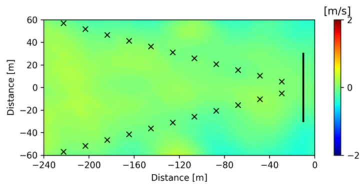

(c) $t = 70$s

Figure 7.16: The difference between the velocity field predicted by the proposed method and the corresponding true values, for the case where the FSTI level is 1%.

difficulties of estimating wind direction from only LoS wind speed data. The prediction MRMSE given in Table 7.2 shows that the prediction errors for both wind magnitude and direction remain very small.

Furthermore a different turbulence level is considered, where the FSTI of the turbulent atmospheric boundary layer is set as 1%. The prediction results are given in Figure 7.15 and 7.16. As can be seen, the predictions match well with the true flow fields. The MRMSE given in Table 7.2 also shows that the proposed method performs very well in this case, similarly as all the other considered cases.

7.4 Conclusions

In this chapter, the prediction of the spatiotemporal wind field based on sparse LIDAR wind speed measurements was investigated by using physics-incorporated deep learning techniques. In order to achieve this, a deep fully-connected NN (which has a total of 12 layers with the hidden-layer neuron number of 128) was first constructed, and then the NS equations (which provide a very good description of atmospheric boundary layer flows) were incorporated in the NN structure. The deep NN structure with a total degree of freedom of 149378 can approximate complex nonlinear systems governed by PDEs (such as the NS system in this work), while the incorporation of NS equations in the NN training empowers the deep NN with the ability to learn the dynamics of the evolving flow field over the whole domain of interest, even though the LIDAR measurements are only available at a few sparse spatial locations. To the best of the author's knowledge, this is for the first time that physical laws and data are fused in a unified manner in the training of deep learning models for wind applications.

The proposed method was evaluated based on the high-fidelity wind farm simulator SOWFA. The results showed that both the wind magnitude and direction were predicted accurately, overcoming the Cyclops' dilemma. This is because the correlations between the LoS wind speed measured at different locations were taken into account implicitly through the NS residue terms. The unsteady wind field predictions were compared with the corresponding true values. The results showed that the wind field predictions were very accurate, with the MRMSE being only 0.198m/s for wind magnitude prediction and 2.77° for wind direction predictions at the baseline case, which were just 7.0% and 19.4% of the corresponding value ranges respectively. In particular, the flow details such as the propagation of the high-speed/low-speed flow structures were captured by the proposed method. Thus it is expected that the proposed method can lead to a significant reduction of turbine blade structural loads through advanced blade control techniques which take these predicted flow details as control input, especially under the smart rotor concept [189]. To further demonstrate the potential use of the proposed method in wind turbine control, the predictions of the averaged and instantaneous wind speeds were examined. The results showed very good matches between the prediction and true values. In addition, they showed that the proposed method could achieve detailed short-term wind forecasting. These results are apparently also very useful for wind farm control and wind resource assessment. Furthermore, a wide range of scenarios were investigated to demonstrate the proposed method's robustness, which included the LIDAR measurements with various levels of noise and under different LIDAR spatial/temporal resolutions, different LIDAR look directions and different turbulence levels. The results showed that the proposed method performed very well in all these scenarios.

By fusing LIDAR measurements and NS equations, the proposed method achieved the accurate predictions of the full spatiotemporal wind field in the 2D domain in front of wind turbines. However, it is worth mentioning that its performance is still limited by the underlying physical law's ability in capturing the full dynamics of the evolving wind. For example, the 3D flow structures and the thermal effects are not captured by the current studies because the employed NS equations are 2D. Therefore, future studies considering more accurate physical models (e.g. 3D NS equations) are needed to improve the prediction performance. In addition, the wind prediction is also limited by the LIDAR measurement data's ability in characterising the essential wind information. Therefore, it is of great interest to investigate the optimal data acquisition design for the proposed method. It is expected that the data quality will increase using an optimised data acquisition design, which, in turn, will increase the prediction accuracy further. Equal weights have been applied to the training loss arising from the NS residue terms (i.e. $\mathscr{L}_1$) and LIDAR measurements (i.e. $\mathscr{L}_2$). Future works may include the investigation of the weights' impact on the training convergence and the prediction performance. Future research may also include the real-world LIDAR measurement campaign to further validate the proposed method.

As the predicted spatiotemporal flow field contains much more information about the incoming wind than the original LIDAR measurements, it is greatly useful in developing advanced strategies for the wind resource assessment and for the monitoring and control of wind turbine/farm, by using such rich flow information. This may include the usage of the predicted spatiotemporal data for wind power prediction, turbine load evaluation, extreme event forecasting, maintenance scheduling, etc. As the developed method is generic, another research direction is the application of the developed method in the state estimation and forecasting of other systems governed by PDEs such as wave/tide energy systems and other flow configurations such as wind over complex terrains.

Chapter 8

Three-Dimensional Spatiotemporal Wind Field Reconstruction Based on Physics-Informed Deep Learning

8.1 Introduction

This chapter extends the work presented in Chapter 7 to the prediction of 3D spatiotemporal wind field in front of a wind turbine, through combining the 3D NS equations and the scanning LIDAR measurements into physics-informed deep learning. In particular, a deep NN is first constructed, then the 3D NS equations are encoded into the deep NN to form the NS residue terms, by using automatic differentiation. Next, the measurement process of the scanning LIDAR is encoded into the deep NN to map the full flow state and the real-time LIDAR beam directions to LIDAR observations. The NN training is finally carried out to minimise the LIDAR observation errors and the NS residues simultaneously. Because the 3D NS equations can describe the 3D unsteady wind very well while the scanning LIDAR provides sparse yet valuable information about the incoming 3D wind, the whole 3D spatiotemporal wind field can be predicted after training. To the best of the author's knowledge, this is for the first time that the prediction of 3D spatiotemporal wind field is achieved based on real-time scattered measurements and physics. From the predicted spatiotemporal flow field, the mean wind quantities (such as the effective wind speed at different heights) and the instantaneous wind quantities (such as the wind speed at specific turbine blade locations) can be extracted.

In addition, the present work further improves the wind field reconstruction performance by taking full advantage of the physics-informed deep learning framework's ability in solving inverse problems. In particular, instead of incorporating the NS equations with pre-determined parameters (i.e the air viscosity in the transport terms), as was done in [190], the present work treats the parameters in the NS equations (i.e. the effective viscosity which is the sum of the air viscosity and turbulent viscosity) as training variables. Therefore, this work achieves the inference of the turbulent viscosity and the reconstruction of the 3D wind field simultaneously. The benefits of solving the inverse problem instead of directly specifying the NS equations with the air viscosity are two-fold. First, the accuracy of reconstructing the 3D wind field is improved, as in this way the turbulence effects are taken into account. Second, the turbulent viscosity is obtained after training, which can be used for characterising the turbulence intensity in other applications such as the modelling and numerical simulation of turbulent wind.

To evaluate the performance of the proposed method, the LES flow solver SOWFA [160] is employed to carry out high-fidelity numerical experiments. SOWFA can simulate the atmospheric boundary layer flows under various conditions and has been widely validated in many studies e.g. on the turbine dynamics [57], the control of wind farms [35] and the wind turbine load in atmospheric flows [191]. During SOWFA simulations, the LoS scalar wind speed at specific spatial locations are extracted to simulate the measurement process of the scanning LIDAR beams, while the 3D flow fields are recorded to provide ground truth for method validations.

The main contributions of this chapter are summarised as follows:

(1) The prediction of 3D spatiotemporal wind field in front of a wind turbine is achieved for the first time, by combing 3D NS equations and scanning LIDAR measurements via physics-informed deep learning. In particular, the whole 3D dynamic wind vector field is reconstructed using only the LoS LIDAR measurements at sparse spatial locations. Because LIDAR devices are becoming widely available for modern wind turbines, and, to the best of the author's knowledge, no other works can achieve similar 3D wind predictions, this work is very useful in advancing other research fields including wind turbine control & monitoring, wind resource assessment, and wind power & load forecasting.

(2) Instead of using pre-determined parameters for the NS equations [190], the proposed method treats the unknown parameters (i.e. the turbulent viscosity) in the NS equations as training variables. In this way, the inference of the turbulent

viscosity is achieved which is very useful for other wind applications such as wind modelling and simulations. It also further improves the performance of wind field reconstruction, as the turbulence effects have been taken into account through the turbulent viscosity.

(3) The proposed method is validated using large-scale high-fidelity numerical experiments, where its accuracy is evaluated in terms of predicting the whole 3D flow field as well as the main wind quantities that are closely related to wind turbine loading, to demonstrate its great importance for various wind applications [83, 189, 192].

The remaining part of this chapter is organised as follows: the spatiotemporal wind field reconstruction problem is formulated in Section 8.2. The physics-informed deep learning based method which combines the 3D NS equations and the LIDAR measurements is described in Section 8.3, where the deep NN structure and its training are given in detail. The prediction performance of the developed method is evaluated in Section 8.4, using high-fidelity CFD simulations. Finally the conclusions are drawn in Section 8.5.

8.2 Problem Formulation

Currently, LIDAR devices are becoming widely available for modern wind turbines. However, LIDAR can only measure the LoS wind speed in the laser beam direction at sparse spatial locations along the laser beams. As the incoming wind in real-world condition is not uniform, the whole 3D wind field in front of wind turbines remains unknown. In order to bridge the gap between the limitation of the current sensor technology and the need of detailed wind field information, this work develops a method to achieve the reconstruction of the whole 3D spatiotemporal wind field in front of a wind turbine, based on LIDAR measurements and 3D NS equations.

An illustration of LIDAR measurements is given in Figure 8.1, where the LIDAR beams (coloured in red) are shown in front of a wind turbine. At a given time instant, the LIDAR beams can measure the LoS wind speed at the discrete spatial locations (which are illustrated as the cross signs in Figure 8.1). The 3D spatiotemporal wind field reconstruction problem considered in this work states as, based on the LIDAR measurements at these sparse locations during certain time period T, how to predict the wind velocity (including the wind velocity components in downwind, crosswind, and vertical directions) at every locations in the 3D spatial domain in front of the wind turbine at every time instant. It is worth noting that this task is not achievable without taking flow physics into account, as only scalar

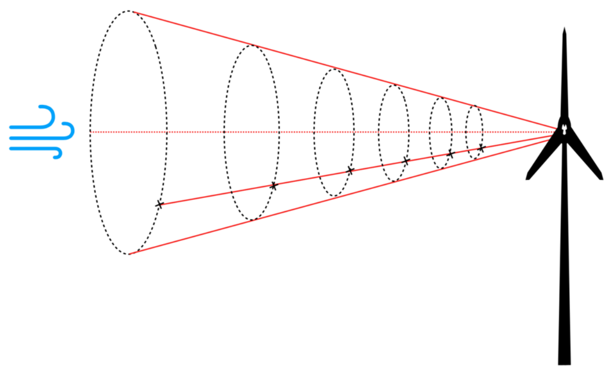


Figure 8.1: The illustration of LIDAR measurements in front of a wind turbine.

measurements (i.e. the LoS wind speed) at sparse locations are available.

8.3 3D Wind Field Reconstruction Method

A 3D spatiotemporal wind field reconstruction method is proposed in this section, where LIDAR measurements and 3D NS equations are combined via the physics-informed deep learning technique. The whole reconstruction framework is demonstrated in Figure 8.2. The NN structure and its training are described in detail in the remainder of this section.

8.3.1 Neural Network Structure

The whole NN structure includes three sub NNs i.e. the Base-NN, the LIDAR-NN and the NS-NN, as shown in Figure 8.2. The Base-NN is first constructed, based on which the LIDAR-NN and the NS-NN are then derived to incorporate LIDAR measurements and NS equations respectively.

The Base-NN is constructed to approximate the mapping between the spatiotemporal coordinates and the flow state variables. Denote the spatiotemporal coordinates as $X = [t, x, y, z]$ (representing the time coordinate and the space coordinate in the 3D Cartesian coordinate system) and the flow state variables as $Y = [u, v, w, p]$ (representing the velocity components in the x, y, z directions and the air pressure, respectively), then the Base-NN, denoted as F, can be expressed as

$$Y = F(X; W) \tag{8.1}$$

where W represents all the training variables in the Base-NN.

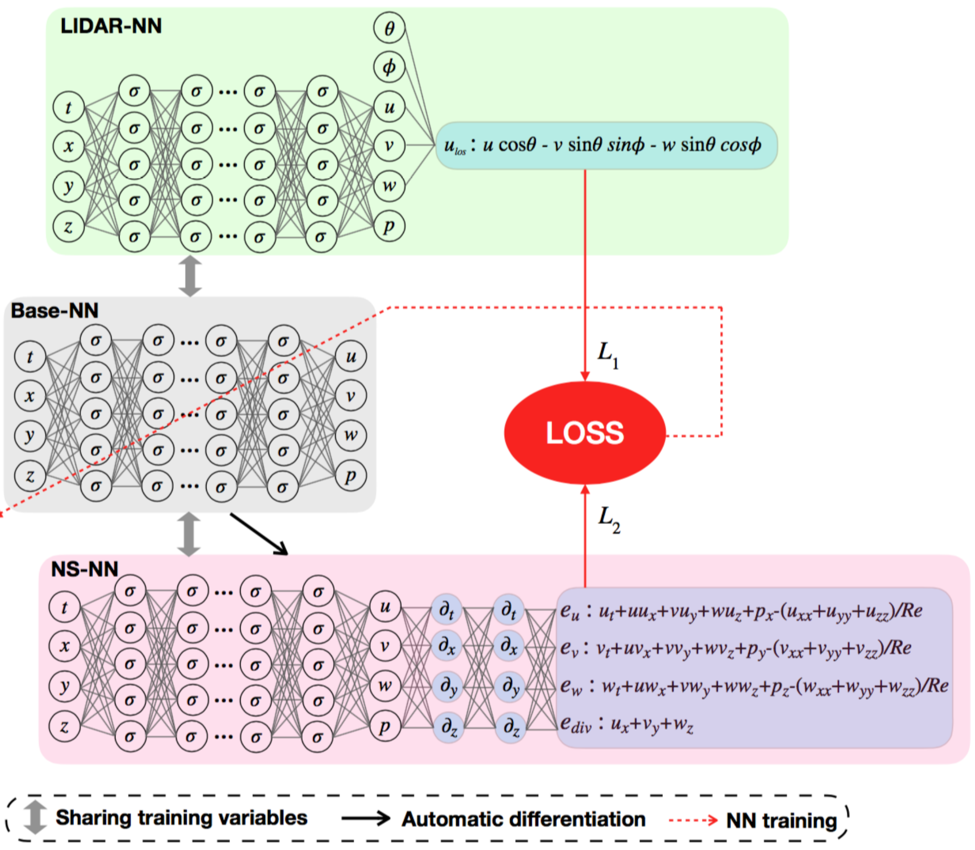

LIDAR-NN
Base-NN
NS-NN
LOSS
L_1
L_2
u_{los} : $u\cos\theta - v\sin\theta\sin\phi - w\sin\theta\cos\phi$
e_u : $u_t+uu_x+vu_y+wu_z+p_x-(u_{xx}+u_{yy}+u_{zz})/Re$
e_v : $v_t+uv_x+vv_y+wv_z+p_y-(v_{xx}+v_{yy}+v_{zz})/Re$
e_w : $w_t+uw_x+vw_y+ww_z+p_z-(w_{xx}+w_{yy}+w_{zz})/Re$
e_{div} : $u_x+v_y+w_z$
Sharing training variables
Automatic differentiation
NN training

Figure 8.2: The demonstration of the proposed 3D spatiotemporal wind field reconstruction method based on the physics-informed deep learning technique.

As the LIDAR can only measure the LoS wind speed in the LIDAR beam direction, no training target of Y is available. The LIDAR-NN, denoted as F_μ, is constructed to incorporate the LIDAR measurements. As the mapping between the flow state variables to the LoS LIDAR measurements depends on the direction of the LIDAR beam (which depends on the LIDAR configurations and can also change with time in the case of the scanning LIDAR), the LIDAR-NN takes two additional NN inputs i.e. the elevation angle θ and the azimuth angle ϕ of the LIDAR beam. Denote the NN input of F_μ as $X_\mu = [X, \theta, \phi]$. Denote the NN output of F_μ as $Y_\mu = [u_{los}]$ which represents the projection of the wind velocity vector in the LIDAR beam direction. Then the LIDAR-NN can be expressed as

$$\begin{aligned} Y_\mu =& F_\mu(X_\mu; W) \\ =& F(X;W)[1]cos(\theta) - F(X;W)[2]sin(\theta)sin(\phi) \\ & - F(X;W)[3]sin(\theta)cos(\phi). \end{aligned} \tag{8.2}$$

As LIDAR only measures the wind information at sparse locations, the whole 3D dynamic flow field in front of the wind turbine remains unknown. Here the NS-NN, denoted as F_{ns}, is derived based on the Base-NN to incorporate the NS equations which provide a very good description of the wind dynamics. The derivation is based on the physics-informed deep learning framework, a novel framework for solving forward and inverse problems involving nonlinear PDEs [145]. The applications of physics-informed deep learning in various research domains have seen great successes recently, which demonstrates the great advantage of combining physics (in terms of PDEs) and data in various scenarios. In this work, for the wind field reconstruction, the 3D NS equations

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -\frac{\partial p}{\partial x} + \frac{1}{Re}(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2})$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} = -\frac{\partial p}{\partial y} + \frac{1}{Re}(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2})$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} = -\frac{\partial p}{\partial z} + \frac{1}{Re}(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2})$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

are first reformulated as

$$e_u = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} + \frac{\partial p}{\partial x} - \frac{1}{Re}(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2})$$

$$e_v = \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + \frac{\partial p}{\partial y} - \frac{1}{Re}(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2})$$

$$e_w = \frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} + \frac{\partial p}{\partial z} - \frac{1}{Re}(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2})$$

$$e_{div} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}.$$

Here Re is defined as $U_\infty D/\nu_{eff}$ with D, U_∞, and ν_{eff} representing the turbine rotor diameter, the average freestream wind speed, and the total effective viscosity respectively. The effective viscosity is defined as $\nu_{eff} = \nu_{air} + \nu_t$, where ν_{air} and ν_t are the kinematic viscosity of air and the turbulent viscosity respectively. As the effective viscosity ν_{eff} (thus Re) is not known, it is treated as training variables and is inferred through the training process. The NS equations are then used to form the NS residue terms in the NS-NN. In particular, the differential terms in the NS equations are derived based on the Base-NN using automatic differentiation

[146]. For example, to incorporate the term $\partial u/\partial x$ in the NS-NN, the gradient of $F(X;W)[1]$ with respect to $X[2]$, denoted as $\partial F_1(X,W)/\partial X_2$, is derived using automatic differentiation. Other first-order terms are obtained similarly. Then the higher-order terms are obtained by the automatic differentiation of the lower-order terms. For example, to incorporate the term $\partial^2 u/\partial x^2$ in the NS-NN, the gradient of $\partial F_1(X,W)/\partial X_2$ with respect to $X[2]$ is derived. All the terms are finally added to form the NS residue terms $e_u(X;[W,1/Re])$, $e_v(X;[W,1/Re])$, $e_w(X;[W,1/Re])$ and $e_{div}(X;W)$. Denote the NN input and the NN output of F_{ns} as $X_{ns}=[t,x,y,z]$ and $Y_{ns}=[e_u,e_v,e_w,e_{div}]$, the NS-NN can then be expressed as

$$Y_{ns} = F_{ns}(X_{ns};[W,1/Re]). \tag{8.3}$$

As can be seen from the above construction process, the Base-NN, the LIDAR-NN and the NS-NN share the same training variables W. The training of the whole NN involves the updating of W and $1/Re$ to minimise the NN loss function, which will be described in detail in the next subsection.

8.3.2 Neural Network Training

After constructing the whole NN structure, the loss function needs to be specified for the NN training. In order to train the NN such that it satisfies the constraints imposed by the NS residue terms and fits the LIDAR measurements simultaneously, the loss function is specified to consist of two parts. The first part is defined as

$$L_1(W) = \frac{1}{N_\mu}\sum_{i=1}^{N_\mu}|F_\mu(t_i^\mu,x_i^\mu,y_i^\mu,z_i^\mu,\theta_i^\mu,\phi_i^\mu;W)-u_i^\mu|^2 \tag{8.4}$$

where $\{[t_i^\mu,x_i^\mu,y_i^\mu,z_i^\mu,\theta_i^\mu,\phi_i^\mu,u_i^\mu],1\le i\le N_\mu\}$ are the LIDAR measurement data with each sample consisting of the time coordinate, the measurement location, the elevation angle, the azimuth angle, and the corresponding value of the LoS wind speed measured by LIDAR. The second part is defined as

$$L_2(W,1/Re) = \frac{1}{N_{ns}}\sum_{i=1}^{N_{ns}}|F_{ns}(t_i^{ns},x_i^{ns},y_i^{ns},z_i^{ns};[W,1/Re])|^2 \tag{8.5}$$

where $\{[t_i^{ns},x_i^{ns},y_i^{ns},z_i^{ns}],1\le i\le N_{ns}\}$ are the randomly-sampled spatiotemporal coordinates corresponding to the spatial domain in front of the wind turbine. It is at these spatiotemporal coordinates that the NS constraints are enforced. The loss function is then defined as

$$L(W, 1/Re) = L_1(W) + L_2(W, 1/Re). \tag{8.6}$$

Finally, the proposed NN structure is trained to minimise the loss function $L(W, 1/Re)$, by updating the training variables W and $1/Re$. In this work, the Adam algorithm [158] is employed for the NN training.

After the NN training, $1/Re$ (thus the effective viscosity ν_{eff}) can be obtained and the Base-NN can be used for the prediction of the wind velocity vector at a given location in front of the wind turbine and a given time instant. The whole wind field at a given time instant can thus be obtained by first generating a 3D mesh corresponding to the flow domain in front of the turbine and then propagating the 3D mesh through the Base-NN. Furthermore, because the deep learning model learnt the spatiotemporal correlation of the wind field from the NS equations, future time instant can be directly fed into the Base-NN for predictions. Therefore, the proposed method can also achieve a short-term wind forecasting without the commonly-used Taylor's frozen turbulence hypothesis.

8.4 Results

The proposed wind field reconstruction method is evaluated in this section, by using high-fidelity numerical experiments. The simulation details and the prediction results are presented in the following subsections.

8.4.1 Simulation Setups

The numerical experiments are carried out using the high-fidelity LES solver SOWFA [160]. For the simulations in this work, a 3D mesh of $250 \times 250 \times 83$ is generated in a $3 \times 3 \times 1$km flow domain. Simulations of 400s are carried out where the freestream wind speed is set as 8m/s and the FSTI is set as 6%. During the last 100s simulations, the whole 3D wind velocity field (including the wind speed in the x, y, z directions at every spatial location) is recorded which is used as the ground truth for evaluating the proposed wind field reconstruction method. The simulations are carried out using 256 CPU cores on local HPC clusters and require around 2 hours to complete.

In addition, a virtual LIDAR device is added to extract the LoS wind speed at the LIDAR measurement locations. In particular, five LIDAR beams, with the measurement frequency of 1s, the spatial resolution of 20m and the measurement

Effective viscosity	Turbulent viscosity ratio
$0.2m^2/s$	2.0×10^4

Table 8.1: The estimation of the effective viscosity by the proposed method.

range of 220m, are included in the virtual measurement process. One of five beams is configured towards the turbine yaw direction i.e. the elevation angle equal to 0°. The other four are configured with an elevation angle of 15° and uniformly-distributed azimuth angles (i.e. the four beams are uniformly distributed in the cone surface as illustrated by the red lines in Figure 8.1). Furthermore, the virtual LIDAR beams are designed to scan over the azimuth direction in order to provide the wind information with better spatial coverage.

8.4.2 Prediction Results and Discussions

The LIDAR measurement data, which is described in Section 8.4.1, is used to train the proposed deep learning model. In this work, the structure of the Base-NN is set as 4-128-128-128-128-128-128-128-4 with the hyperbolic tangent activation for the intermediate layers and the linear activation for the last layer. The learning rate of the Adam optimiser is set as 10^{-4}. The training is carried out using NVIDIA Tesla K80 GPU and each training iteration requires around 0.14s, which illustrates the ability of the proposed approach for real-time 3D dynamic wind field reconstruction through offline training and online updating. The scanning speed of the LIDAR beams in the azimuth direction still needs to be specified. In this work, the scanning speed is set as a constant value of 15°/s, which is determined by trying out a set of different values and choosing the value with the smallest prediction RMSE.

After training, the effective viscosity is obtained. The results are given in Table 8.1, along with the turbulent viscosity ratio i.e. the ratio between the turbulent viscosity and the air viscosity. The results clearly show that the turbulent wind is characterised primarily by the turbulent viscosity, as the effective viscosity is much larger than the air viscosity. As the turbulent viscosity can be used to describe the wind turbulence, it is very useful for other wind applications such as wind modelling and simulations. For example, it can be used to specify the turbulent inflow conditions for numerical simulations of wind turbine wakes.

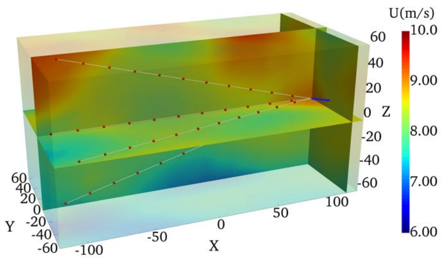
Next, the whole spatiotemporal 3D flow field is predicted. The results for three typical time instants are given in Figure 8.3 and 8.4, where the ground truth is also included for comparisons. The corresponding error distributions are shown in Figure 8.5 and Figure 8.6. Figure 8.3 shows the visualisations of the wind speed magnitude in the x-y plane at the hub height and in the x-z plane, while Figure 8.4 shows the flow visualisations in the y-z plane at various streamwise locations. As can be seen from Figure 8.3, the flow structures (e.g. the high-speed/low-speed flow zones) in both x-y and x-z planes are predicted very accurately. The wind features in the vertical direction, such as the wind shear (e.g. the increase of the wind speed with height), are also captured very well. As can be seen from Figure 8.4, the flow fields in the 2-D domain parallel to the rotor plane are predicted very accurately at various streamwise locations before the turbine. This demonstrates that the proposed method can provide detailed preview wind information for the whole 2-D rotor plane, which is of great importance for the control of wind turbines especially in the case of smart rotors [189]. In addition, the unsteady flow visualisations including both the ground truth and the prediction results are available online in the supporting materials of the published paper. As shown in the videos, the predicted flow field matches with the ground truth very well, demonstrating that the proposed approach captures the 3D spatial variation and the temporal evolution of the incoming turbulent wind successfully. The results fully reveal the great performance of the proposed approach.

To further quantify the prediction accuracy, the RMSE of the flow field prediction is calculated, which is defined as

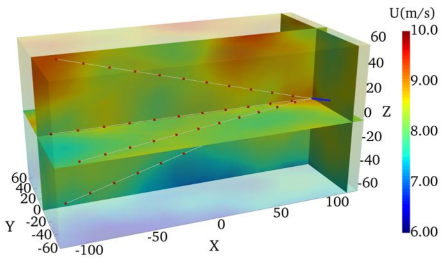
$$\epsilon_q = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{N_{test}}\sum_{i=1}^{N_{test}}(q^*_{x_i,y_i,z_i,t} - \hat{q}_{x_i,y_i,z_i,t})^2}. \tag{8.7}$$

Here the total time T is 100, $\{[x_i, y_i, z_i], 1 \leq i \leq N_{test}\}$ is the grid points corresponding to the $12 \times 12 \times 12$m uniform mesh in the 3D flow domain in front of the wind turbine, and q represents the flow quantity such as the wind velocity components in x, y and z directions (i.e. u, v, w). q^* and $\hat{q}$ represent the true value and the corresponding predicted value of q. The results are given in Table 8.2. As can be seen, the predictions for the streamwise velocity u, the spanwise velocity v and the vertical velocity w are all quite accurate, with the RMSEs equal to 6.5%, 11.9%, and 12.7% of the corresponding value ranges. Furthermore, the directional information of the wind velocity can also be predicted, by first projecting the 3D velocity vector to the 2-D plane and then calculating the angle between the 2-D vector and the reference direction. The directions of the wind vector projected in $x - y$ and $x - z$ planes, at any given location, can be calculated by

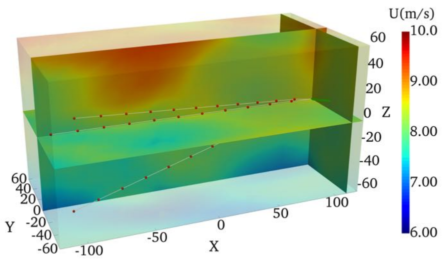
$$\gamma_y = arctan(v/u) \tag{8.8}$$



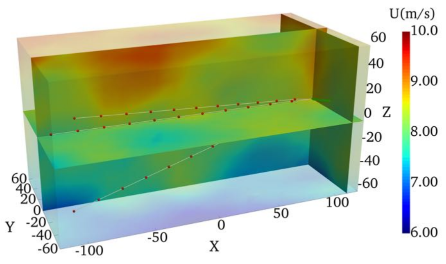
(a) $t = 60$s, prediction



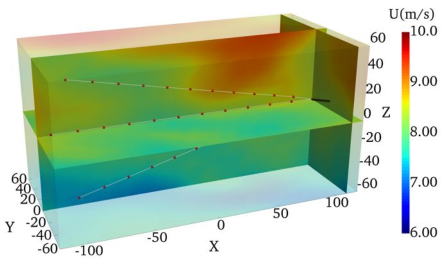
(b) $t = 60$s, true



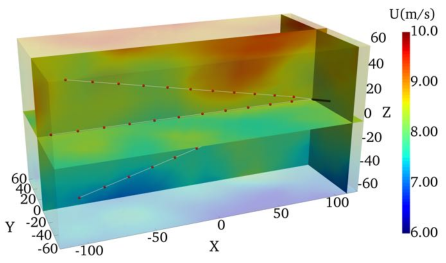
(c) $t = 70$s, prediction



(d) $t = 70$s, true



(e) $t = 80$s, prediction



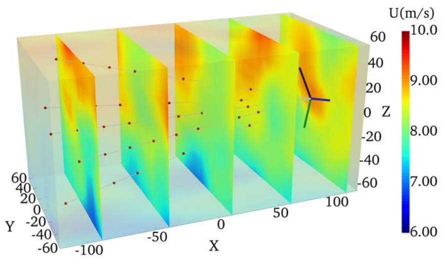
(f) $t = 80$s, true

Figure 8.3: The 3D velocity field (visualised in x-y and x-z planes) predicted by the proposed method at time (a) $t = 60$s, (c) $t = 70$s and (e) $t = 80$s. The corresponding ground truth is also shown for comparisons (b, d, f).

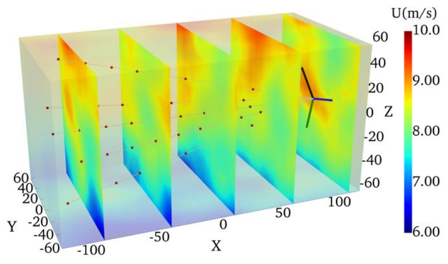
and

$$\gamma_z = arctan(w/u). \tag{8.9}$$

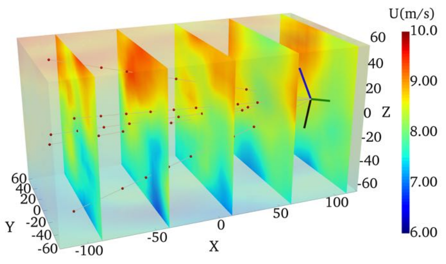
The RMSEs of the γ_y and γ_z fields are also included in Table 8.2. As shown, γ_y, which is actually the conventional wind direction, is predicted correctly. This demonstrates that the proposed approach can be used for tackling the yaw misalignment which is of great importance for improving the efficiency of wind power generations [193]. The vertical wind direction γ_z is also predicted correctly, which



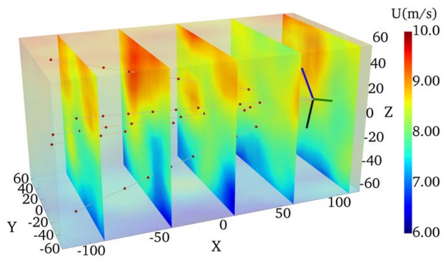
(a) $t = 60$s, prediction



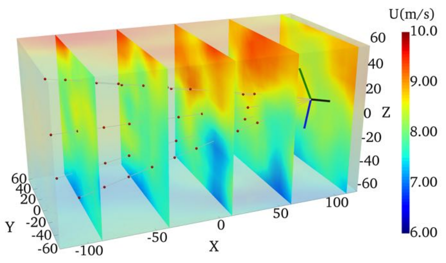
(b) $t = 60$s, true



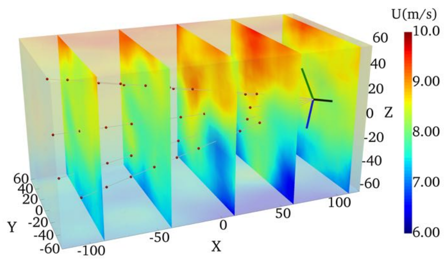
(c) $t = 70$s, prediction



(d) $t = 70$s, true



(e) $t = 80$s, prediction

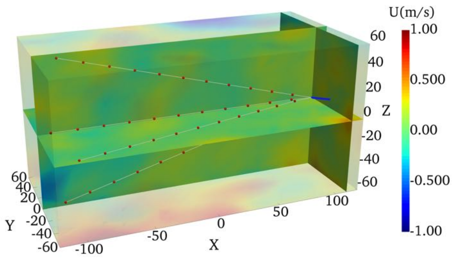


(f) $t = 80$s, true

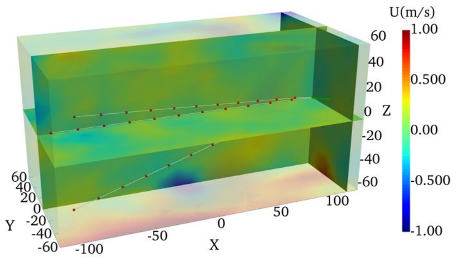
Figure 8.4: The 3D velocity field (visualised in y-z planes) predicted by the proposed method at time (a) $t = 60$s, (c) $t = 70$s and (e) $t = 80$s. The corresponding true values are also shown for comparisons (b, d, f).

shows that the proposed approach can be used for the control of turbine tilt angles [194]. It is concluded that the proposed method predicts the directional wind information very well, given that only scalar information is available in the original LIDAR measurements.

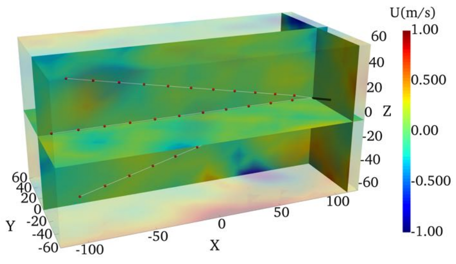
In addition, the 3D spatiotemporal wind field reconstruction is also carried out by using the pre-determined viscosity in the NS equations. The prediction RMSEs are given in Table 8.3. The results in Table 8.2 and 8.3 clearly demonstrate that by including and inferring the effective viscosity in the proposed model, the present work achieves better accuracy for all the flow quantities.



(a) $t = 60\text{s}$



(b) $t = 70\text{s}$



(c) $t = 80\text{s}$

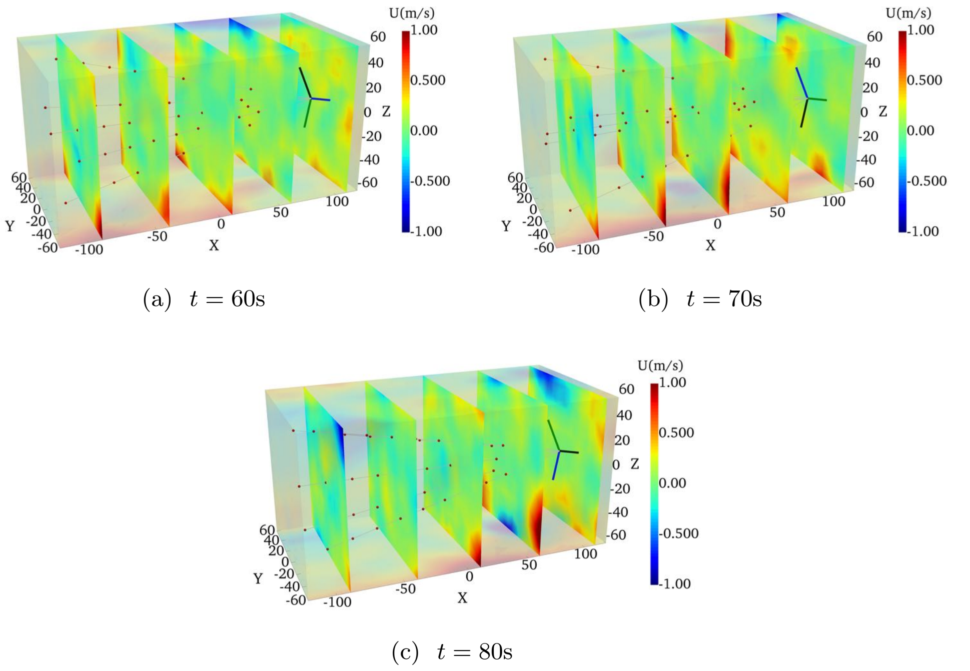
Figure 8.5: The difference between the 3D velocity field (visualised in x-y and x-z planes) predicted by the proposed method and the corresponding ground truth, at time (a) $t = 60\text{s}$, (b) $t = 70\text{s}$ and (c) $t = 80\text{s}$.

Quantity (units)	Value range	RMSE (% of range)
u (m/s)	[6.08, 10.11]	0.263 (6.5%)
v (m/s)	[-1.82, 1.53]	0.397 (11.9%)
w (m/s)	[-1.48, 1.36]	0.361(12.7%)
γ_y $(^\circ)$	[-11.4, 11.8]	2.84 (12.2%)
γ_z $(^\circ)$	[-10.1, 9.77]	2.58(13.0%)

Table 8.2: The RMSEs of the 3D flow field predictions by the method proposed in this chapter.

To further evaluate the prediction performance and to illustrate the use of the proposed method for wind turbine control and wind power & load forecasting, the effective wind speed and the instantaneous wind speed at specific turbine blade locations are extracted from the predicted full spatiotemporal wind field and then compared with the corresponding true values in the following parts.

First, the effective wind speed, which is defined here as the wind speed av-


(a) $t = 60$s

(b) $t = 70$s

(c) $t = 80$s

Figure 8.6: The difference between the 3D velocity field (visualised in y-z planes) predicted by the proposed method and the corresponding ground truth, at time (a) $t = 60$s, (b) $t = 70$s and (c) $t = 80$s.

Quantity (units)	Value range	RMSE (% of range)
u (m/s)	[6.08, 10.11]	0.276 (6.8%)
v (m/s)	[-1.82, 1.53]	0.457 (13.6%)
w (m/s)	[-1.48, 1.36]	0.364(12.8%)
γ_y (°)	[-11.4, 11.8]	3.25 (14.0%)
γ_z (°)	[-10.1, 9.77]	2.61(13.1%)

Table 8.3: The RMSEs of the 3D flow field predictions by using the pre-determined viscosity.

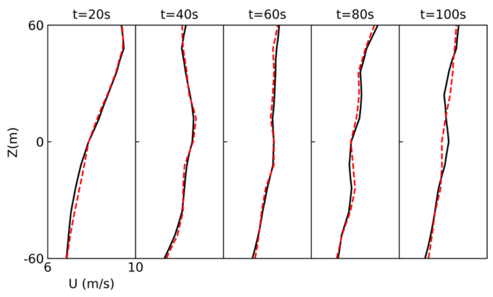
eraged over the y direction, is calculated as

$$\bar{U}_{x_0,t}(z) = \frac{1}{N_y} \sum_{i=1}^{N_y} \hat{u}_{x_0,y_i,z,t}, \tag{8.10}$$

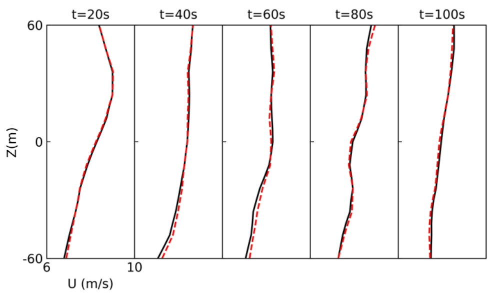
where $\{y_i, 1 \leq i \leq N_y\}$ is a set of uniformly-distributed y coordinates in the interval $[-30, 30]$m. The effective wind speed is defined here as the wind speed averaged only over the y direction instead of over the y-z rotor plane, because in this way the wind speed variation in vertical direction can be shown clearly. The results are given in Figure 8.7 for several streamwise locations i.e. $x_0 = [-50, -10, 30, 70, 110]$m. The corresponding RMSEs are [0.082, 0.086, 0.102, 0.133, 0.173]m/s for $x_0 = [-50, -10, 30, 70, 110]$m, respectively. As shown, the profiles of the effective wind speed are predicted accurately at all the streamwise locations and all time instants. The wind shear, i.e. the increase of the wind speed magnitude with height, is captured very well.

Second, the instantaneous wind speeds at the turbine blade root, 1/4 chord length, 1/2 chord length, 3/4 chord length and the blade tip are predicted and compared with the corresponding ground truth. For illustration purpose, here the rotational speed of the wind turbine is set as 60°/s, and the variable rotational speed can be applied in the same way. The results are given in Figure 8.8. As shown, the time series of the wind speed are predicted very accurately, including both the slow variations due to the freestream flow structures and the fast variations due to the turbine rotations. In particular, the effect of the wind shear increases clearly from the blade root to the blade tip location, as shown by the increase of the oscillation magnitudes from Figure 8.8(a) to 8.8(e). The magnitudes of wind speed oscillations, which are modulated by the incoming turbulent wind and differ for each turbine rotation period, are also predicted accurately. To further illustrate the use of the proposed approach in wind turbine load evaluations, the wind velocity relative to the rotating turbine blade is extracted and the results are given in Figure 8.9 and 8.10, where the velocity components in the axial direction and in the tangential direction are shown respectively. The corresponding RMSEs at 1/4 chord length, 1/2 chord length, 3/4 chord length and the blade tip are [0.115, 0.183, 0.241, 0.251]m/s and [0.340, 0.359, 0.359, 0.372]m/s for axial velocity and tangential velocity respectively. As shown, the axial wind velocity is predicted very accurately while the prediction for tangential wind velocity is less accurate. This is because LIDAR mainly measures the wind velocity component in the axial direction, providing much less information in the tangential direction.

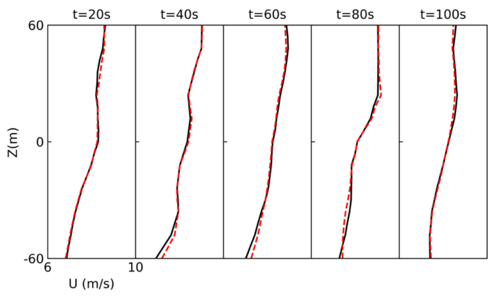
Next, as the proposed deep learning model learns the temporal correlations of the wind field from the NS equations, it can be used directly for short-term wind forecasting. This is achieved by directly feeding future time coordinates to the Base-NN. The results for 15s-ahead wind speed forecasting are included in Figure 8.8, 8.9 and 8.10. Another 15s numerical simulations are also carried out by SOWFA to obtain the corresponding ground truth for comparisons. As shown by the last 15s time series in Figure 8.8, 8.9 and 8.10, the forecasting results match with the true



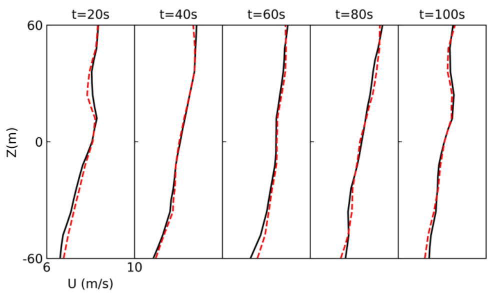
(a) x_0 = -50m



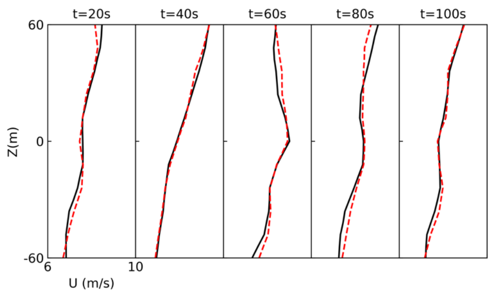
(b) x_0 = -10m



(c) x_0 = 30m



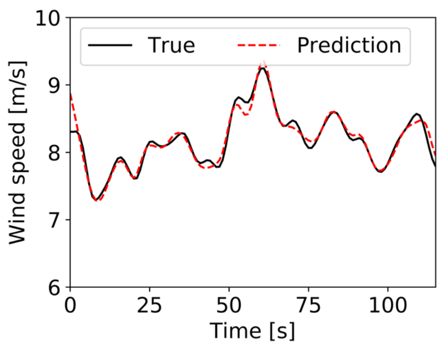
(d) x_0 = 70m



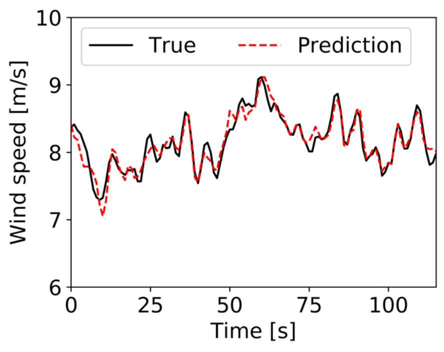
(e) x_0 = 110m

Figure 8.7: The profile of the effective wind speed predicted by the proposed method (the dashed lines) at $x_0 = [-50, -10, 30, 70, 110]$m and various time instants. The corresponding true values (the solid lines) are also shown for comparisons.

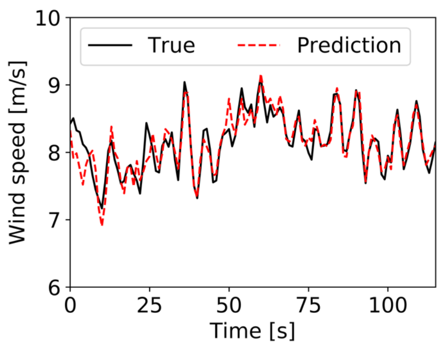
values quite well. It is worth mentioning that the proposed method does not need any prior parameter tunings to determine the forecasting time horizon. The forecasting can be achieved with good accuracy as long as the wind speed at the location and the time instant of interest is correlated with the available LIDAR measurement data. In practice, the maximum forecasting time horizon can be estimated as the virtual time of the flow convection from the upstream measurement points to the turbine location.



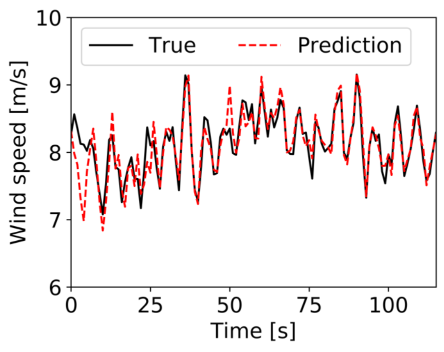
(a) blade root



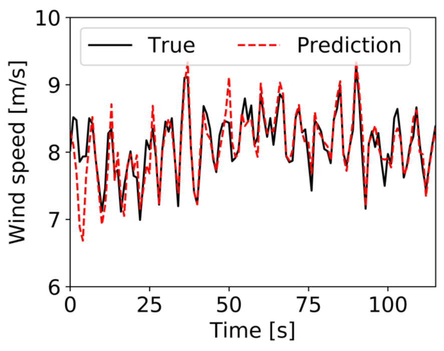
(b) 1/4 chord length



(c) 1/2 chord length

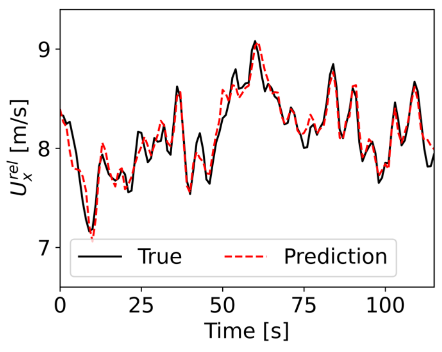


(d) 3/4 chord length

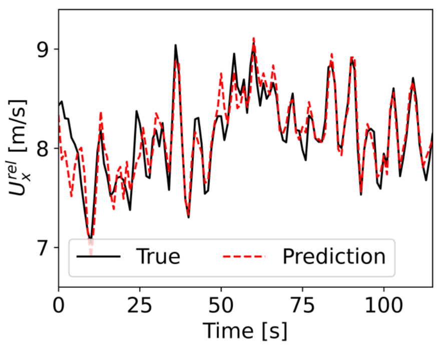


(e) blade tip

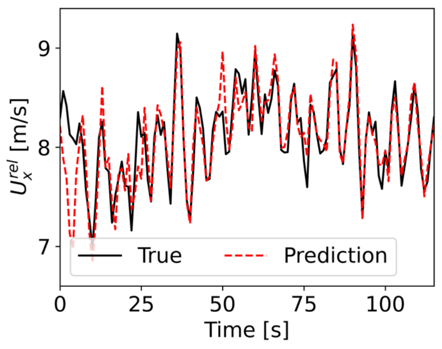
Figure 8.8: The instantaneous wind speeds predicted by the proposed method at the turbine blade root, 1/4 chord length, 1/2 chord length, 3/4 chord length, and the blade tip. The corresponding true values are also shown for comparisons.



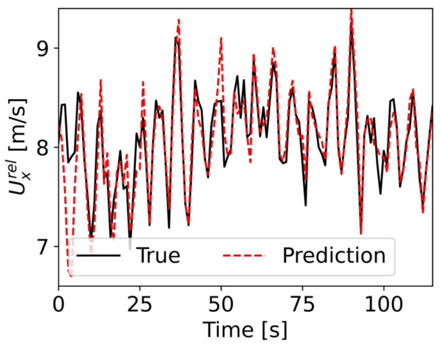
(a) 1/4 chord length



(b) 1/2 chord length



(c) 3/4 chord length

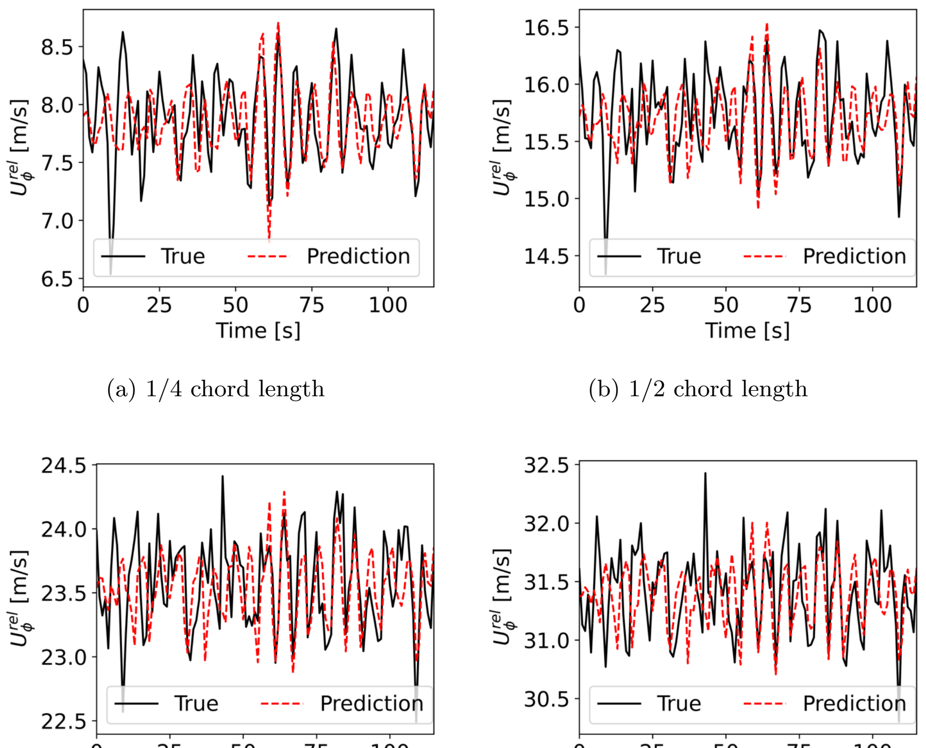


(d) blade tip

Figure 8.9: The instantaneous axial wind speeds relative to the rotating turbine blade, predicted by the proposed method at 1/4 chord length, 1/2 chord length, 3/4 chord length, and the blade tip. The corresponding true values are also shown for comparisons.

8.5 Conclusions

The 3D spatiotemporal wind field reconstruction was investigated in this chapter, where a physics-informed deep learning based method was proposed to combine the 3D NS equations and the scanning LIDAR measurements. The results showed that, by combining the physics and data, the whole 3D dynamic wind velocity vector field (including the velocity components in x, y, and z directions) in front of the wind turbine was predicted very accurately based on only the limited scalar information at very sparse spatial locations (i.e. the LoS wind speed measured by LIDAR beams). In particular, only 11 measurement points per LIDAR beam were used for the predictions. In addition, the inference of the turbulent viscosity was also achieved, which can be used for characterising the wind turbulence in other applications such



(a) 1/4 chord length (b) 1/2 chord length

(c) 3/4 chord length (d) blade tip

Figure 8.10: The instantaneous tangential wind speeds relative to the rotating turbine blade, predicted by the proposed method at 1/4 chord length, 1/2 chord length, 3/4 chord length, and the blade tip. The corresponding true values are also shown for comparisons.

as wind modelling and numerical simulations of wind turbine wakes.

The 3D wind field predictions were first examined by visualising the flow fields in x-y, x-z, and y-z planes. The results showed that the spatiotemporal flow field predicted by the proposed approach matched with the corresponding ground truth very well, where the 3D spatial variation of the incoming wind (such as the evolving flow structures and the vertical wind shear) was successfully predicted. The prediction accuracy was then quantified by the RMSEs of the reconstructed spatiotemporal wind fields. The results showed that the RMSEs were only 0.263m/s, 0.397m/s, 0.361m/s for the streamwise velocity u, the spanwise velocity v, and the vertical velocity w which were only 6.5%, 11.9%, and 12.7% of the corresponding value ranges, demonstrating the great accuracy of the proposed method. To the best of the author's knowledge, this is for the first time that this type of accurate and detailed predictions of the unsteady 3D wind field in front of a wind turbine is achieved. Furthermore, the directional wind information, including the conventional wind direction (i.e. the wind direction in the $x - y$ plane) and the vertical wind direction (i.e. the wind direction in the $x - z$ plane), was also predicted. The results showed that the RMSEs were only 2.84° and 2.58° respectively, demonstrating the great potential of the proposed method in tackling yaw misalignment and turbine tilt control, which are of great interest in improving the energy capture efficiency of wind turbines. For example, the field test in [195] showed that the annual energy production can be increased by 2.4% by applying yaw corrections, and the study in [194] showed that turbine tilt control has a great impact on the power generation of wind farms.

The predicted wind field is of vital importance for wind energy applications e.g. wind turbine control design to further increase the power generation efficiency, accurate forecasting of wind power to aid its grid integration, and detailed and reliable wind resource assessments. To illustrate these points, the effective wind speeds at various streamwise locations and the instantaneous wind speeds at various turbine blade locations were extracted from the predicted wind field and then compared with the corresponding ground truth. The results showed that the effective and instantaneous wind speeds were both predicted very accurately at all the considered locations. In particular, the wind speed oscillations due to the blade rotations and the variations of the oscillation magnitudes at each rotation period due to the freestream turbulent wind were both captured very well. The vertical wind shear was also predicted accurately. Furthermore, a short-term wind forecasting was carried out and the results showed that the accurate forecasting of the wind speeds at various locations ranging from the turbine blade root to the blade tip was achieved without the commonly-used Taylor's frozen turbulence hypothesis [185].

To further improve the prediction performance, future works include the design of the LIDAR configuration and its scanning pattern to optimally arrange the measurement points, and the incorporation of other flow sensors in the proposed method to provide more versatile flow measurements.

Chapter 9

Conclusions and Future Works

9.1 Conclusions

In this Thesis, ML-based approaches have been proposed to tackle the challenges arising from the modelling and control of wind turbine structures and wind farm wakes. First, the RL-based approach was investigated for the structural control of floating wind turbines. Then the research moved to the farm level, focusing on the ML-based modelling of wind farm wakes. Next, the prediction of the spatiotemporal wind field in front of wind turbines was investigated based on the physics-informed deep learning. The works presented in this thesis showed very promising results in the interdisciplinary research field involving high-fidelity numerical simulations (including high-fidelity structural models and CFD models), physical laws in terms of PDEs (i.e. the NS equations), and ML (including RL, supervised ML, dimensionality reduction, GAN, physics-informed deep learning).

Chapter 2 investigated the RL-based structural control of floating wind turbines, where an active TMD was installed on the floating platform to suppress the vibration of the structural system. In particular, the ADP algorithm was employed to derive the optimal control law based on the nonlinear structural dynamics, and a NN structure consisting of three sub-networks (i.e. the plant network, critic network, and action network) was designed and implemented to realise the ADP algorithm. Numerical simulations were carried out using the floating wind turbine model within FAST to evaluate the proposed RL-based controllers. The results showed that the proposed controller achieved great performance (e.g. the standard deviation of the platform pitch displacement being reduced by around 40%) for the load mitigation of floating wind turbine structures in a wide range of wave and wind conditions. A clear advantage of the developed controllers over traditional approaches was observed, especially for extreme conditions - the scenarios that must be considered seriously in offshore wind technology. Moreover, the tradeoff between the control performance and power consumption was systematically investigated.

Chapter 3 investigated the quantification of parameter uncertainty in wind farm wake models using the Bayesian UQ framework. High-fidelity CFD simulations of wind farms were first carried out using the LES flow solver SOWFA and then the generated flow field data was used to infer the posterior distributions of the parameters in the widely-used wake model FLORIS. The results showed that by taking parameter uncertainty into account, the flow field prediction was improved and a correct characteristic of uncertainty in the 'mixing zone' was captured. In addition, the predictions of the turbine power generations were improved. In particular, the FLORIS model with parameter uncertainty (called stochastic FLORIS) can predict the turbine power fluctuation much better than the original FLORIS. This demonstrates that the stochastic FLORIS model can be used to not only maximise the average power but also minimise the power fluctuation.

Chapter 4 proposed an ML-based surrogate modelling method for distributed fluid systems and then applied it to wind farm wake modelling. In the proposed surrogate modelling procedure, various dimensionality reduction techniques (e.g. POD, ICA, AE) were employed for reducing the flow field dimension and the fully-connected NN was employed for predicting the reduced representations of the flow field with the flow parameters as the input. The proposed method was designed specifically to tackle the fluid flows over distributed structures, by carrying out surrogate modelling for each subdomain and combining the flow field of each subdomain with the consideration of the matching condition at the interface. Its efficiency, accuracy, and scalability were first demonstrated by a 1-D Poisson equation case, then it was applied to build a data-based wind farm wake model. The results showed that the developed surrogate model was able to achieve accurate and real-time prediction of wind farm wakes, with the prediction RMSE for the velocity field in the whole domain being only 2% of the freestream wind speed.

Chapter 5 improved the work presented in Chapter 4, by proposing a novel ML-based surrogate modelling method based on the state-of-the-art deep learning framework DC-CGAN. The proposed method was applied to developing a novel data-based wake model. The developed wake model (called the CGAN model) was first validated against high-fidelity data and the results showed that the CGAN model was able to predict the multi-channel wind turbine wake flows (both streamwise and spanwise velocity fields) very accurately and efficiently. In particular, the prediction RMSEs for the streamwise and spanwise velocity fields were just 0.10m/s and 0.18m/s respectively which were just 1.1% and 4.1% of the corresponding value ranges. Then a comprehensive parametric study was carried out and the results showed that the CGAN model learned the qualitative features of wind turbine wake flows and generalised well to the flow scenarios that were not present in the training dataset. Furthermore, a case study for a small wind farm was carried out and the results showed that the CGAN model was able to achieve accurate, robust, and real-time predictions of wind farm wake flows.

Chapter 6 further improved the works in previous chapters on static wake modelling, by developing a novel dynamic wind farm wake model. First, a set of LES simulations were carried out to generate a high-fidelity flow field database with designed wind conditions and yaw changes. Then a deep learning model, called POD-LSTM, was trained to learn the wake dynamics from this valuable database. The POD was employed for reducing the flow field dimension and the LSTM was employed for predicting future flow fields with the historical data as the input. The results showed that the developed model was able to capture the main unsteady flow features (such as the streamwise convection of flow structures, the wake meandering, the wake's deflection with changing yaw, and the wake interactions between wind turbines) similarly as high-fidelity wake models while running as fast as the low-fidelity static wake models. The results of a 9-turbine test case further showed that the developed model was able to predict the unsteady turbine wakes in several seconds on a standard desktop while it would require tens of thousands of CPU hours on an HPC cluster if a high-fidelity model is used. As the existing wake models in the literature are either too time-consuming or unable to capture detailed wake dynamics, the developed model brings a step change in fast and accurate simulations, predictions, and control designs of wind farms. However, as the developed model only considers the interactions between subdomains through the upstream boundary (which is enough in capturing the main wake interactions), there exist discontinuities in the predicted flow fields at the interface between different rows of wind turbines. This discontinuity issue may be solved by including all the boundary conditions of each subdomain as the input in the POD-LSTM model. However, the consideration of the interactions at all the subdomain boundaries is not trivial and needs further investigation regarding the generation of the training dataset and the numerical stability. Future works may include the development of new approaches to tackle this issue.

Chapter 7 investigated the spatiotemporal wind field predictions in front of a wind turbine, based on the physics informed deep learning framework and LIDAR measurements. The deep learning model in this chapter fused data (i.e. the LIDAR measurements data) and physics (i.e. the NS equations) in the training process in a unified manner. Therefore it can predict the spatiotemporal wind information that is not present in the training data, which is not achievable using traditional ML methods. The prediction performance was evaluated based on the high-fidelity flow solver SOWFA. The results showed that the spatiotemporal wind field (including both wind speed and direction fields) in the whole domain in front of the wind turbine was predicted very accurately for a wide range of scenarios (including various measurement noises, resolutions, LIDAR look directions, and turbulence levels), based on only LoS wind speed measurements at sparse locations. In particular, the MRMSE of wind field prediction was only 0.198m/s for wind magnitude prediction and 2.77° for wind direction predictions at the baseline case, which were just 7.0% and 19.4% of the corresponding value ranges. The averaged and instantaneous wind speeds were also predicted and the results showed the effective wind speed at various streamwise locations and the instantaneous wind speed at turbine blade locations were all predicted very well. In addition, the detailed short-term wind forecasting was also achieved without relying on the widely-used Taylor's frozen turbulence hypothesis.

Chapter 8 extended the work presented in Chapter 7 to three-dimensional spatiotemporal wind field predictions, by combining the scanning LIDAR measurements and the 3D NS equations. To further improve the prediction performance, instead of incorporating the NS equations with pre-determined parameters (i.e. the air viscosity in the transport terms) as in Chapter 7, this chapter treated the parameters in the NS equations (i.e. the effective viscosity) as training variables. The results showed that the whole 3D dynamic wind velocity vector field (including the velocity components in x, y, and z directions) in front of the wind turbine was predicted very accurately based on only the limited scalar LoS wind speed measured at very sparse spatial locations by LIDAR. The RMSEs of the 3D flow field predictions were only 0.263m/s, 0.397m/s, 0.361m/s for the streamwise velocity u, the spanwise velocity v, and the vertical velocity w which were only 6.5%, 11.9%, and 12.7% of the corresponding value ranges. The main wind features were correctly captured including the turbulent viscosity, the wind directions (both the conventional and vertical wind directions), the evolving flow structures, and the vertical wind shear. In addition, the predictions of the effective and instantaneous wind speeds were examined and the results showed that the wind speed oscillations at specific turbine blade locations due to the blade rotations and the variations of the oscillation magnitudes at each rotation period due to the freestream turbulent wind were both captured very well. The prediction RMSEs for the relative wind speed at

1/4 chord length, 1/2 chord length, 3/4 chord length and the blade tip were [0.115, 0.183, 0.241, 0.251]m/s and [0.340, 0.359, 0.359, 0.372]m/s for axial and tangential velocity respectively. The predicted 3D spatiotemporal wind field (which is not available with current sensor and prediction technologies) is very useful in advancing other wind energy research fields e.g. wind turbine control & monitoring, power forecasting, and resource assessments.

9.2 Future Works

The works presented in this thesis involve control engineering, structural mechanics, fluid dynamics, scientific computing, machine learning, and their applications in renewable energy systems (in particular wind turbine/farm systems). The field is just emerging and large research gaps still exist for both method developments and applications. A few important research directions are discussed below.

- More research efforts on the control of wind turbines and wind farms are needed in order to mitigate the fatigue load of the turbine structures, maximise the power capturing efficiency of wind farms, and facilitate the integration of wind power into the power grid. Both model-based and model-free control approaches need to be explored. The former may include the use of traditional models and the ML-based models developed in this thesis in the control design process, while the latter may include model-free RL approaches which are developing fast in ML and control communities. The development of novel ML-based models for wind turbine/farm systems is needed to further improve the model accuracy, efficiency, and robustness. Such models are of great interests for both model-based control design as internal models and the model-free control design as fast simulation tools. As only the wake interaction in the upstream direction is considered in the wake models developed throughout this thesis, more works are needed to take account of the interactions between wind turbines in the lateral directions.

- To improve the performance of the ML-based models, it is very important to develop novel ML approaches that can combine domain knowledge with data. This is especially suitable for physical systems, as the underlying physics are usually known & well studied and the consideration of physics in the ML model is extremely beneficial. Furthermore, the ML field itself is developing fast. Therefore, more works are needed to take advantage of the latest development of the ML hardware (such as multi-GPU clusters), software (such as the further development of the deep learning platform including Tensorflow [155] and Pytorch [196]), and algorithms (including the ML algorithms arising from computer science, control engineering, and scientific computing fields).

- The works presented in this thesis, including the structural control of floating wind turbines, the ML-based modelling of wind farm wakes and the spatiotemporal wind predictions using physic-informed deep learning, have been extensively validated with high-fidelity numerical models. To further evaluate the proposed control and modelling methods and to further demonstrate their values in wind energy applications, future works may include experimental studies (including the structural and flow dynamics in wave tank and wind tunnel experiments) and the corresponding real-world measurement campaign.
- It is also of great interests to apply the surrogate modelling methods proposed in this thesis to other fluid flows and renewable energy systems. For example, ML-based wave models may be developed using the proposed methods based on the wave data generated by numerical simulations or experiments. In addition, it is very interesting to explore the spatiotemporal flow field reconstruction in other renewable energy systems (such as wave energy and tidal energy systems) based on physics-informed deep learning and scattered measurements, where the proximity to bounding surfaces will need further investigations.

Bibliography

[1] J. Lee, F. Zhao, A. Dutton, B. Backwell, R. Fiestas, L. Qiao, N. Balachandran, S. Lim, W. Liang, E. Clarke, *et al.*, "Global wind report 2019," *Brussels: Global Wind Energy Council (GWEC)*, 2020.

[2] A. Pullen and M. G. Eneland, "Global wind 2006 report," *Brussels: Global Wind Energy Council (GWEC)*, 2007.

[3] J. M. Jonkman, "Dynamics modeling and loads analysis of an offshore floating wind turbine," *National Renewable Energy Lab.(NREL), Golden, CO (United States)*, 2007.

[4] M. Adaramola and P.-Å. Krogstad, "Experimental investigation of wake effects on wind turbine performance," *Renewable Energy*, vol. 36, no. 8, pp. 2078–2086, 2011.

[5] W. P. Mahoney, K. Parks, G. Wiener, Y. Liu, W. L. Myers, J. Sun, L. Delle Monache, T. Hopson, D. Johnson, and S. E. Haupt, "A wind power forecasting system to optimize grid integration," *IEEE Transactions on Sustainable Energy*, vol. 3, no. 4, pp. 670–682, 2012.

[6] S. Lin, X. Zhao, and X. Tong, "Feasibility studies of a converter-free grid-connected offshore hydrostatic wind turbine," *IEEE Transactions on Sustainable Energy*, vol. 11, no. 4, pp. 2494–2503, 2020.

[7] Z. Shu, Q. Li, Y. He, and P. Chan, "Observations of offshore wind characteristics by doppler-lidar for wind energy applications," *Applied Energy*, vol. 169, pp. 150–163, 2016.

[8] X.-Y. Tang, S. Zhao, B. Fan, J. Peinke, and B. Stoevesandt, "Micro-scale wind resource assessment in complex terrain based on cfd coupled measurement from multiple masts," *Applied Energy*, vol. 238, pp. 806–815, 2019.

[9] X. Li and H. Gao, “Load mitigation for a floating wind turbine via generalized H_∞ structural control,” *IEEE Transactions on Industrial Electronics*, vol. 63, no. 1, pp. 332–342, 2016.

[10] H. Wang, Z. Lei, X. Zhang, B. Zhou, and J. Peng, “A review of deep learning for renewable energy forecasting,” *Energy Conversion and Management*, vol. 198, p. 111799, 2019.

[11] Y. Wang, D. Zhang, Y. Liu, B. Dai, and L. H. Lee, “Enhancing transportation systems via deep learning: A survey,” *Transportation research part C: emerging technologies*, vol. 99, pp. 144–163, 2019.

[12] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, “Machine learning for fluid mechanics,” *Annual Review of Fluid Mechanics*, vol. 52, 2019.

[13] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, *et al.*, “Opportunities and obstacles for deep learning in biology and medicine,” *Journal of The Royal Society Interface*, vol. 15, no. 141, p. 20170387, 2018.

[14] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[15] M. A. Lackner, “Controlling platform motions and reducing blade loads for floating wind turbines,” *Wind Engineering*, vol. 33, no. 6, pp. 541–553, 2009.

[16] H. Namik and K. Stol, “Individual blade pitch control of floating offshore wind turbines,” *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, vol. 13, no. 1, pp. 74–85, 2010.

[17] P. A. Fleming, A. Peiffer, and D. Schlipf, “Wind turbine controller to mitigate structural loads on a floating wind turbine platform,” in *ASME 2016 35th International Conference on Ocean, Offshore and Arctic Engineering*, pp. V006T09A044–V006T09A044, American Society of Mechanical Engineers, 2016.

[18] M. A. Lackner, “An investigation of variable power collective pitch control for load mitigation of floating offshore wind turbines,” *Wind Energy*, vol. 16, no. 3, pp. 435–444, 2013.

[19] X. Zhao and G. Weiss, "Strong stabilisation of a wind turbine tower model in the plane of the turbine blades," *International Journal of Control*, vol. 87, pp. 2027–2034, 2014.

[20] C. Chang and H. T. Yang, "Control of buildings using active tuned mass dampers," *Journal of engineering mechanics*, vol. 121, no. 3, pp. 355–366, 1995.

[21] G. Stewart and M. Lackner, "Offshore wind turbine load reduction employing optimal passive tuned mass damping systems," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1090–1104, 2013.

[22] V. N. Dinh and B. Basu, "Passive control of floating offshore wind turbine nacelle and spar vibrations by multiple tuned mass dampers," *Structural Control and Health Monitoring*, vol. 22, no. 1, pp. 152–176, 2015.

[23] X. Tong, X. Zhao, and S. Zhao, "Load reduction of a monopile wind turbine tower using optimal tuned mass dampers," *International Journal of Control*, vol. 90, no. 7, pp. 1283–1298, 2017.

[24] Y. Si, H. R. Karimi, and H. Gao, "Modelling and optimization of a passive structural control design for a spar-type floating wind turbine," *Engineering Structures*, vol. 69, pp. 168–182, 2014.

[25] S. Colwell and B. Basu, "Tuned liquid column dampers in offshore wind turbines for structural control," *Engineering Structures*, vol. 31, no. 2, pp. 358–368, 2009.

[26] C. Roderick, "Vibration reduction of offshore wind turbines using tuned liquid column dampers," Master's Thesis, University of Massachusetts Amherst 2012.

[27] C. Coudurier, O. Lepreux, and N. Petit, "Passive and semi-active control of an offshore floating wind turbine using a tuned liquid column damper," *IFAC-PapersOnLine*, vol. 48, no. 16, pp. 241–247, 2015.

[28] X. Tong, X. Zhao, and A. Karcanias, "Passive vibration control of an offshore floating hydrostatic wind turbine model," *Wind Energy*, vol. 21, no. 9, pp. 697–714, 2018.

[29] M. A. Lackner and M. A. Rotea, "Passive structural control of offshore wind turbines," *Wind energy*, vol. 14, no. 3, pp. 373–388, 2011.

[30] M. A. Lackner and M. A. Rotea, “Structural control of floating wind turbines,” *Mechatronics*, vol. 21, no. 4, pp. 704–719, 2011.

[31] G. M. Stewart and M. A. Lackner, “The effect of actuator dynamics on active structural control of offshore wind turbines,” *Engineering Structures*, vol. 33, no. 5, pp. 1807–1816, 2011.

[32] Y. Hu and E. He, “Active structural control of a floating wind turbine with a stroke-limited hybrid mass damper,” *Journal of Sound and Vibration*, vol. 410, pp. 447–472, 2017.

[33] X. Gao, H. Yang, and L. Lu, “Optimization of wind turbine layout position in a wind farm using a newly-developed two-dimensional wake model,” *Applied energy*, vol. 174, pp. 192–200, 2016.

[34] K. Yang, G. Kwak, K. Cho, and J. Huh, “Wind farm layout optimization for wake effect uniformity,” *Energy*, vol. 183, pp. 983–995, 2019.

[35] P. Fleming, P. M. Gebraad, S. Lee, J.-W. van Wingerden, K. Johnson, M. Churchfield, J. Michalakes, P. Spalart, and P. Moriarty, “Simulation comparison of wake mitigation control strategies for a two-turbine case,” *Wind Energy*, vol. 18, pp. 2135–2143, Oct. 2014.

[36] S. Boersma, B. Doekemeijer, P. Gebraad, P. Fleming, J. Annoni, A. Scholbrock, J. Frederik, and J.-W. van Wingerden, “A tutorial on control-oriented modeling and control of wind farms,” in *2017 American Control Conference (ACC)*, IEEE, May 2017.

[37] N. O. Jensen, *A note on wind generator interaction.* 1983.

[38] I. Katic, J. Højstrup, and N. O. Jensen, “A simple model for cluster efficiency,” in *European wind energy association conference and exhibition*, A. Raguzzi, 1987.

[39] S. Frandsen, R. Barthelmie, S. Pryor, O. Rathmann, S. Larsen, J. Højstrup, and M. Thøgersen, “Analytical modelling of wind speed deficit in large offshore wind farms,” *Wind Energy*, vol. 9, pp. 39–53, Jan. 2006.

[40] P. M. O. Gebraad, F. W. Teeuwisse, J. W. van Wingerden, P. A. Fleming, S. D. Ruben, J. R. Marden, and L. Y. Pao, “Wind plant power optimization through yaw control using a parametric model for wake effects-a CFD simulation study,” *Wind Energy*, vol. 19, pp. 95–114, Dec. 2014.

[41] H. Sun and H. Yang, "Study on an innovative three-dimensional wind turbine wake model," *Applied energy*, vol. 226, pp. 483–493, 2018.

[42] X. Gao, B. Li, T. Wang, H. Sun, H. Yang, Y. Li, *et al.*, "Investigation and validation of 3d wake model for horizontal-axis wind turbines based on filed measurements," *Applied Energy*, vol. 260, p. 114272, 2020.

[43] M. Bastankhah and F. Porté-Agel, "A new analytical model for wind-turbine wakes," *Renewable Energy*, vol. 70, pp. 116–123, 2014.

[44] A. Niayifar and F. Porté-Agel, "Analytical modeling of wind farms: A new approach for power prediction," *Energies*, vol. 9, p. 741, Sept. 2016.

[45] L. Tian, W. Zhu, W. Shen, N. Zhao, and Z. Shen, "Development and validation of a new two-dimensional wake model for wind turbine wakes," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 137, pp. 90–99, 2015.

[46] C. R. Shapiro, D. F. Gayme, and C. Meneveau, "Modelling yawed wind turbine wakes: a lifting line approach," *Journal of Fluid Mechanics*, vol. 841, 2018.

[47] D. Lopez, J. Kuo, and N. Li, "A novel wake model for yawed wind turbines," *Energy*, vol. 178, pp. 158–167, 2019.

[48] B. Dou, M. Guala, L. Lei, and P. Zeng, "Wake model for horizontal-axis wind and hydrokinetic turbines in yawed conditions," *Applied energy*, vol. 242, pp. 1383–1395, 2019.

[49] R. Brogna, J. Feng, J. N. Sørensen, W. Z. Shen, and F. Porté-Agel, "A new wake model and comparison of eight algorithms for layout optimization of wind farms in complex terrain," *Applied Energy*, vol. 259, p. 114189, 2020.

[50] M. Ge, Y. Wu, Y. Liu, and Q. Li, "A two-dimensional model based on the expansion of physical wake boundary for wind-turbine wakes," *Applied energy*, vol. 233, pp. 975–984, 2019.

[51] J. Zhang and X. Zhao, "Quantification of parameter uncertainty in wind farm wake modeling," *Energy*, vol. 196, p. 117065, 2020.

[52] K. Nilsson, S. Ivanell, K. S. Hansen, R. Mikkelsen, J. N. Sørensen, S.-P. Breton, and D. Henningson, "Large-eddy simulations of the lillgrund wind farm," *Wind Energy*, vol. 18, pp. 449–467, Feb. 2014.

[53] Y.-T. Wu and F. Porté-Agel, "Modeling turbine wakes and power losses within a wind farm using LES: An application to the horns rev offshore wind farm," *Renewable Energy*, vol. 75, pp. 945–955, Mar. 2015.

[54] M. Calaf, C. Meneveau, and J. Meyers, "Large eddy simulation study of fully developed wind-turbine array boundary layers," *Physics of Fluids*, vol. 22, p. 015110, Jan. 2010.

[55] J. Meyers and C. Meneveau, "Large eddy simulations of large wind-turbine arrays in the atmospheric boundary layer," in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, American Institute of Aeronautics and Astronautics, Jan. 2010.

[56] H. Lu and F. Porté-Agel, "Large-eddy simulation of a very large wind farm in a stable atmospheric boundary layer," *Physics of Fluids*, vol. 23, no. 6, p. 065101, 2011.

[57] M. J. Churchfield, S. Lee, J. Michalakes, and P. J. Moriarty, "A numerical study of the effects of atmospheric and wake turbulence on wind turbine dynamics," *Journal of Turbulence*, vol. 13, p. N14, Jan. 2012.

[58] B. Witha, G. Steinfeld, and D. Heinemann, "High-resolution offshore wake simulations with the LES model PALM," in *Research Topics in Wind Energy*, pp. 175–181, Springer Berlin Heidelberg, 2014.

[59] L. A. Martínez-Tossas, M. J. Churchfield, and S. Leonardi, "Large eddy simulations of the flow past wind turbines: actuator line and disk modeling," *Wind Energy*, vol. 18, pp. 1047–1060, Apr. 2014.

[60] R. J. Stevens, L. A. Martínez-Tossas, and C. Meneveau, "Comparison of wind farm large eddy simulations using actuator disk and actuator line models with wind tunnel experiments," *Renewable energy*, vol. 116, pp. 470–478, 2018.

[61] M. Sprague, S. Ananthan, G. Vijayakumar, and M. Robinson, "Exawind: A multi-fidelity modeling and simulation environment for wind energy," in *J. Phys. Conf. Series*, 2020.

[62] G. C. Larsen, H. M. Aagaard, F. Bingöl, J. Mann, S. Ott, J. N. Sørensen, V. Okulov, N. Troldborg, N. M. Nielsen, K. Thomsen, *et al.*, "Dynamic wake meandering modeling," 2007.

[63] S. Boersma, B. Doekemeijer, M. Vali, J. Meyers, and J.-W. v. Wingerden, "A control-oriented dynamic wind farm model: Wfsim," *Wind Energy Science*, vol. 3, no. 1, pp. 75–95, 2018.

[64] C. R. Shapiro, G. M. Starke, C. Meneveau, and D. F. Gayme, "A wake modeling paradigm for wind farm design and control," *Energies*, vol. 12, no. 15, p. 2956, 2019.

[65] G. S. Böhme, E. A. Fadigas, A. L. Gimenes, and C. E. Tassinari, "Wake effect measurement in complex terrain-a case study in brazilian wind farms," *Energy*, vol. 161, pp. 277–283, 2018.

[66] M. Gaumond, P.-E. Réthoré, S. Ott, A. Pena, A. Bechmann, and K. S. Hansen, "Evaluation of the wind direction uncertainty and its impact on wake modeling at the horns rev offshore wind farm," *Wind Energy*, vol. 17, no. 8, pp. 1169–1178, 2014.

[67] A. Peña, P.-E. Réthoré, and M. P. van der Laan, "On the application of the jensen wake model using a turbulence-dependent wake decay coefficient: the sexbierum case," *Wind Energy*, vol. 19, no. 4, pp. 763–776, 2016.

[68] J. Quick, J. Annoni, R. King, K. Dykes, P. Fleming, and A. Ning, "Optimization under uncertainty for wake steering strategies," in *Journal of Physics: Conference Series*, vol. 854, p. 012036, IOP Publishing, 2017.

[69] A. Rott, B. Doekemeijer, J. K. Seifert, J.-W. v. Wingerden, and M. Kühn, "Robust active wake control in consideration of wind direction variability and uncertainty," *Wind Energy Science*, vol. 3, no. 2, pp. 869–882, 2018.

[70] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 3, pp. 425–464, 2001.

[71] S. H. Cheung, T. A. Oliver, E. E. Prudencio, S. Prudhomme, and R. D. Moser, "Bayesian uncertainty analysis with applications to turbulence modeling," *Reliability Engineering & System Safety*, vol. 96, no. 9, pp. 1137–1149, 2011.

[72] W. Edeling, P. Cinnella, R. P. Dwight, and H. Bijl, "Bayesian estimates of parameter variability in the k–ε turbulence model," *Journal of Computational Physics*, vol. 258, pp. 73–94, 2014.

[73] J. Zhang and S. Fu, "An efficient approach for quantifying parameter uncertainty in the SST turbulence model," *Computers & Fluids*, vol. 181, pp. 173–187, Mar. 2019.

[74] T. A. Oliver and R. D. Moser, "Bayesian uncertainty quantification applied to rans turbulence models," in *Journal of Physics: Conference Series*, vol. 318, p. 042032, IOP Publishing, 2011.

[75] J. Zhang and S. Fu, "An efficient bayesian uncertainty quantification approach with application to k-ω-γ transition modeling," *Computers & Fluids*, vol. 161, pp. 211–224, 2018.

[76] J. Ray, S. Lefantzi, S. Arunajatesan, and L. Dechant, "Bayesian parameter estimation of ak-ε model for accurate jet-in-crossflow simulations," *AIAA Journal*, pp. 2432–2448, 2016.

[77] M. M. Hand and M. J. Balas, "Blade load mitigation control design for a wind turbine operating in the path of vortices," *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, vol. 10, no. 4, pp. 339–355, 2007.

[78] M. Harris, M. Hand, and A. Wright, "Lidar for turbine control," *National Renewable Energy Laboratory, Golden, CO, Report No. NREL/TP-500-39154*, 2006.

[79] E. Simley, L. Pao, N. Kelley, B. Jonkman, and R. Frehlich, "Lidar wind speed measurements of evolving wind fields," in *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, p. 656, 2012.

[80] D. Schlipf, P. W. Cheng, and J. Mann, "Model of the correlation between lidar systems and wind turbines for lidar-assisted control," *Journal of Atmospheric and Oceanic Technology*, vol. 30, no. 10, pp. 2233–2240, 2013.

[81] A. Scholbrock, P. Fleming, D. Schlipf, A. Wright, K. Johnson, and N. Wang, "Lidar-enhanced wind turbine control: Past, present, and future," in *2016 American Control Conference (ACC)*, pp. 1399–1406, IEEE, 2016.

[82] X. Tong and X. Zhao, "Power generation control of a monopile hydrostatic wind turbine using an $\mathcal{H}_\infty$ loop-shaping torque controller and an LPV pitch controller," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 2165–2172, 2018.

[83] J. Li, X. Wang, and X. B. Yu, "Use of spatio-temporal calibrated wind shear model to improve accuracy of wind resource assessment," *Applied energy*, vol. 213, pp. 469–485, 2018.

[84] F. Dunne, E. Simley, and L. Pao, "Lidar wind speed measurement analysis and feed-forward blade pitch control for load mitigation in wind turbines," *Golden, CO: National Renewable Energy Laboratory*, 2011.

[85] Z. Peng, S. Peng, L. Fu, B. Lu, J. Tang, K. Wang, and W. Li, "A novel deep learning ensemble model with data denoising for short-term wind speed forecasting," *Energy Conversion and Management*, vol. 207, p. 112524, 2020.

[86] J. Duan, H. Zuo, Y. Bai, J. Duan, M. Chang, and B. Chen, "Short-term wind speed forecasting using recurrent neural networks with error correction," *Energy*, p. 119397, 2020.

[87] Y. Liu, H. Qin, Z. Zhang, S. Pei, Z. Jiang, Z. Feng, *et al.*, "Probabilistic spatiotemporal wind speed forecasting based on a variational bayesian deep learning model," *Applied Energy*, vol. 260, p. 114259, 2020.

[88] B. Yan and Q. Li, "Coupled on-site measurement/cfd based approach for high-resolution wind resource assessment over complex terrains," *Energy Conversion and Management*, vol. 117, pp. 351–366, 2016.

[89] P. Towers and B. L. Jones, "Real-time wind field reconstruction from lidar measurements using a dynamic wind model and state estimation," *Wind Energy*, vol. 19, no. 1, pp. 133–150, 2016.

[90] J. Mercieca, P. Aram, B. L. Jones, and V. Kadirkamanathan, "A spatiotemporal estimation framework for real-world lidar wind speed measurements," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 4, pp. 1595–1602, 2020.

[91] S. Sun, S. Liu, J. Liu, and H. I. Schlaberg, "Wind field reconstruction using inverse process with optimal sensor placement," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 3, pp. 1290–1299, 2019.

[92] S. Sun, S. Liu, M. Chen, and H. Guo, "An optimized sensing arrangement in wind field reconstruction using cfd and pod," *IEEE Transactions on Sustainable Energy*, vol. 11, no. 4, pp. 2449–2456, 2020.

[93] P. J. Werbos, "A menu of designs for reinforcement learning over time," *Neural networks for control*, pp. 67–95, 1990.

[94] P. J. Werbos, "Approximate dynamic programming for real time control and neural modeling," *Handbook of intelligent control: neural, fuzzy and adaptive approaches*, pp. 493–525, 1992.

[95] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE circuits and systems magazine*, vol. 9, no. 3, 2009.

[96] F. L. Lewis and D. Liu, *Reinforcement learning and approximate dynamic programming for feedback control*, vol. 17. John Wiley & Sons, 2013.

[97] D. Wang, D. Liu, Q. Zhang, and D. Zhao, "Data-based adaptive critic designs for nonlinear robust optimal control with uncertain dynamics," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 11, pp. 1544–1555, 2015.

[98] D. Wang, D. Liu, H. Li, B. Luo, and H. Ma, "An approximate optimal control approach for robust stabilization of a class of discrete-time nonlinear systems with uncertainties," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 5, pp. 713–717, 2015.

[99] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li, *Adaptive dynamic programming with applications in optimal control.* Springer, 2017.

[100] Y. Jiang and Z. P. Jiang, *Robust Adaptive Dynamic Programming.* John Wiley & Sons, 2017.

[101] S. Xue, B. Luo, and D. Liu, "Event-triggered adaptive dynamic programming for zero-sum game of partially unknown continuous-time nonlinear systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 9, pp. 3189–3199, 2020.

[102] H. N. Wu and Z. Y. Liu, "Data-driven guaranteed cost control design via reinforcement learning for linear systems with parameter uncertainties," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 11, pp. 4151–4159, 2020.

[103] Q. Wei and D. Liu, "Adaptive dynamic programming for optimal tracking control of unknown nonlinear systems with application to coal gasification," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 4, pp. 1020–1036, 2013.

[104] Q. Wei, D. Liu, G. Shi, and Y. Liu, "Multibattery optimal coordination control for home energy management systems via distributed iterative adaptive dynamic programming," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 7, pp. 4203–4214, 2015.

[105] C. Mu, Z. Ni, C. Sun, and H. He, "Air-breathing hypersonic vehicle tracking control based on adaptive dynamic programming," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 584–598, 2016.

[106] D. Wang, H. He, C. Mu, and D. Liu, "Intelligent critic control with disturbance attenuation for affine dynamics including an application to a microgrid system," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 4935–4944, 2017.

[107] H. Dong, X. Zhao, and B. Luo, "Optimal tracking control for uncertain nonlinear systems with prescribed performance via critic-only ADP," *IEEE Transactions on Systems, Man, and Cybernetics: Systems (Early Access). DOI: 10.1109/TSMC.2020.3003797.*, 2020.

[108] Q. Wei, F. L. Lewis, D. Liu, R. Song, and H. Lin, "Discrete-time local value iteration adaptive dynamic programming: Convergence analysis," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 6, pp. 875–891, 2016.

[109] B. Luo, D. Liu, T. Huang, and J. Liu, "Output tracking control based on adaptive dynamic programming with multistep policy evaluation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 10, pp. 2155–2165, 2019.

[110] B. Luo, Y. Yang, H. N. Wu, and T. Huang, "Balancing value iteration and policy iteration for discrete-time control," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 11, pp. 3948–3958, 2020.

[111] H. Li and D. Liu, "Optimal control for discrete-time affine non-linear systems using general value iteration," *IET Control Theory & Applications*, vol. 6, no. 18, pp. 2725–2736, 2012.

[112] Q. Wei, D. Liu, and Y. Xu, "Neuro-optimal tracking control for a class of discrete-time nonlinear systems via generalized value iteration adaptive dynamic programming approach," *Soft Computing*, vol. 20, no. 2, pp. 697–706, 2016.

[113] D. Liu, Q. Wei, *et al.*, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems.," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 3, pp. 621–634, 2014.

[114] D. Liu, Q. Wei, and P. Yan, "Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 12, pp. 1577–1591, 2015.

[115] D. Amsallem, J. Cortial, and C. Farhat, "Towards real-time computational-fluid-dynamics-based aeroelastic computations using a database of reduced-order information," *AIAA Journal*, vol. 48, pp. 2029–2037, Sept. 2010.

[116] M. Fossati, "Evaluation of aerodynamic loads via reduced-order methodology," *AIAA Journal*, vol. 53, pp. 2389–2405, Aug. 2015.

[117] R. Dupuis, J.-C. Jouhaud, and P. Sagaut, "Surrogate modeling of aerodynamic simulations for multiple operating conditions using machine learning," *AIAA Journal*, vol. 56, pp. 3622–3635, Sept. 2018.

[118] Q. Wang, J. S. Hesthaven, and D. Ray, "Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem," *Journal of Computational Physics*, vol. 384, pp. 289–307, May 2019.

[119] R. Swischuk, B. Kramer, C. Huang, and K. Willcox, "Learning physics-based reduced-order models for a single-injector combustion process," *AIAA Journal*, vol. 58, pp. 2658–2672, June 2020.

[120] X. Guo, W. Li, and F. Iorio, "Convolutional neural networks for steady flow approximation," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, Aug. 2016.

[121] X. Jin, P. Cheng, W.-L. Chen, and H. Li, "Prediction model of velocity field around circular cylinder over various reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder," *Physics of Fluids*, vol. 30, p. 047105, Apr. 2018.

[122] G. Berkooz, P. Holmes, and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," *Annual Review of Fluid Mechanics*, vol. 25, pp. 539–575, Jan. 1993.

[123] A. Chatterjee, "An introduction to the proper orthogonal decomposition," *Current science*, vol. 78, no. 7, pp. 808–817, 2000.

[124] Y. Liang, H. Lee, S. Lim, W. Lin, K. Lee, and C. Wu, “Proper orthogonal decomposition and its applications—part i: Theory,” *Journal of Sound and Vibration*, vol. 252, pp. 527–544, May 2002.

[125] J. Hesthaven and S. Ubbiali, “Non-intrusive reduced order modeling of nonlinear problems using neural networks,” *Journal of Computational Physics*, vol. 363, pp. 55–78, June 2018.

[126] M. Guo and J. S. Hesthaven, “Reduced order modeling for nonlinear structural analysis using gaussian process regression,” *Computer Methods in Applied Mechanics and Engineering*, vol. 341, pp. 807–826, Nov. 2018.

[127] M. Guo and J. S. Hesthaven, “Data-driven reduced order modeling for time-dependent problems,” *Computer Methods in Applied Mechanics and Engineering*, vol. 345, pp. 75–99, Mar. 2019.

[128] R. Swischuk, L. Mainini, B. Peherstorfer, and K. Willcox, “Projection-based model reduction: Formulations for physics-based machine learning,” *Computers & Fluids*, vol. 179, pp. 704–717, Jan. 2019.

[129] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014.

[130] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.

[131] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, July 2017.

[132] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” *arXiv preprint arXiv:1701.07875*, 2017.

[133] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2017.

[134] Y. Xie, E. Franz, M. Chu, and N. Thuerey, “Tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow,” *ACM Transactions on Graphics*, vol. 37, pp. 1–15, Aug. 2018.

[135] Z. Deng, C. He, Y. Liu, and K. C. Kim, “Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework,” *Physics of Fluids*, vol. 31, p. 125111, Dec. 2019.

[136] M. Cheng, F. Fang, C. Pain, and I. Navon, “Data-driven modelling of nonlinear spatio-temporal fluid flows using a deep convolutional generative adversarial network,” *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113000, 2020.

[137] A. Subramaniam, M. L. Wong, R. D. Borker, S. Nimmagadda, and S. K. Lele, “Turbulence enrichment using physics-informed generative adversarial networks,” *arXiv preprint arXiv:2003.01907*, 2020.

[138] J. Ling, A. Kurzawski, and J. Templeton, “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance,” *Journal of Fluid Mechanics*, vol. 807, pp. 155–166, 2016.

[139] K. Duraisamy, G. Iaccarino, and H. Xiao, “Turbulence modeling in the age of data,” *Annual Review of Fluid Mechanics*, vol. 51, pp. 357–377, 2019.

[140] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[141] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Data-driven discovery of partial differential equations,” *Science Advances*, vol. 3, no. 4, p. e1602614, 2017.

[142] J. Sirignano and K. Spiliopoulos, “Dgm: A deep learning algorithm for solving partial differential equations,” *Journal of Computational Physics*, vol. 375, pp. 1339–1364, 2018.

[143] Y. Zhu and N. Zabaras, “Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification,” *Journal of Computational Physics*, vol. 366, pp. 415–447, 2018.

[144] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, “Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data,” *Journal of Computational Physics*, vol. 394, pp. 56–81, 2019.

[145] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[146] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5595–5637, 2017.

[147] Y. Yang and P. Perdikaris, "Adversarial uncertainty quantification in physics-informed neural networks," *Journal of Computational Physics*, vol. 394, pp. 136–152, 2019.

[148] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," *Journal of Computational Physics*, vol. 404, p. 109136, 2020.

[149] X. Meng and G. E. Karniadakis, "A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems," *Journal of Computational Physics*, vol. 401, p. 109020, 2020.

[150] L. Yang, X. Meng, and G. E. Karniadakis, "B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data," *arXiv preprint arXiv:2003.06097*, 2020.

[151] M. Raissi, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, "Deep learning of vortex-induced vibrations," *Journal of Fluid Mechanics*, vol. 861, pp. 119–137, 2019.

[152] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, 2020.

[153] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

[154] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds.), vol. 27, pp. 2672–2680, Curran Associates, Inc., 2014.

[155] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems." https://www.tensorflow.org/, 2015. Software available from tensorflow.org.

[156] J. M. Jonkman, M. L. Buhl Jr, *et al.*, "Fast user's guide," *National Renewable Energy Laboratory, Golden, CO, Technical Report No. NREL/EL-500-38230*, 2005.

[157] B. Jonkman and J. Jonkman, "Fast v8. 16.00 a-bjj," *National Renewable Energy Laboratory*, 2016.

[158] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[159] B. Jonkman and L. Kilcher, "Turbsim user's guide: Version 1.06. 00, tech. rep.," 2012.

[160] M. Churchfield and S. Lee, "Nwtc design codes-sowfa," *URL: http://wind.nrel.gov/designcodes/simulators/SOWFA*, 2012.

[161] M. Churchfield, S. Lee, P. Moriarty, L. Martinez, S. Leonardi, G. Vijayakumar, and J. Brasseur, "A large-eddy simulations of wind-plant aerodynamics," in *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, American Institute of Aeronautics and Astronautics, Jan. 2012.

[162] H. Haario, E. Saksman, J. Tamminen, *et al.*, "An adaptive metropolis algorithm," *Bernoulli*, vol. 7, no. 2, pp. 223–242, 2001.

[163] R. C. Team *et al.*, "R: A language and environment for statistical computing," 2013.

[164] C. Chivers, "Mhadaptive: general markov chain monte carlo for bayesian inference using adaptive metropolis-hastings sampling," 2012.

[165] J. Jonkman, S. Butterfield, W. Musial, and G. Scott, "Definition of a 5-mw reference wind turbine for offshore system development," *National Renewable Energy Lab.(NREL), Golden, CO (United States)*, 2009.

[166] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis.* John Wiley & Sons, Inc., May 2001.

[167] G. E. Hinton, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, July 2006.

[168] S. Tou, C. Tso, and X. Zhang, "3-d numerical analysis of natural convective liquid cooling of a 3×3 heater array in rectangular enclosures," *International Journal of Heat and Mass Transfer*, vol. 42, pp. 3231–3244, Sept. 1999.

[169] M. S. Mon and U. Gross, "Numerical study of fin-spacing effects in annular-finned tube heat exchangers," *International Journal of Heat and Mass Transfer*, vol. 47, pp. 1953–1964, Apr. 2004.

[170] D. P. Rizzetta and M. R. Visbal, "Direct numerical simulations of flow past an array of distributed roughness elements," *AIAA Journal*, vol. 45, pp. 1967–1976, Aug. 2007.

[171] J. Liu, J. Srebric, and N. Yu, "Numerical simulation of convective heat transfer coefficients at the external surfaces of building arrays immersed in a turbulent boundary layer," *International Journal of Heat and Mass Transfer*, vol. 61, pp. 209–225, June 2013.

[172] G. V. Iungo, C. Santoni-Ortiz, M. Abkar, F. Porté-Agel, M. A. Rotea, and S. Leonardi, "Data-driven reduced order model for prediction of wind turbine wakes," *Journal of Physics: Conference Series*, vol. 625, p. 012009, June 2015.

[173] N. Hamilton, B. Viggiano, M. Calaf, M. Tutkun, and R. B. Cal, "A generalized framework for reduced-order modeling of a wind turbine wake," *Wind Energy*, vol. 21, pp. 373–390, Jan. 2018.

[174] J. Annoni, P. Gebraad, and P. Seiler, "Wind farm flow modeling using an input-output reduced-order model," in *2016 American Control Conference (ACC)*, IEEE, July 2016.

[175] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Nov, p. 2825–2830, 2011.

[176] F. Chollet *et al.*, "Keras." `https://github.com/fchollet/keras`, 2015.

[177] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 13, pp. 411–430, June 2000.

[178] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[179] J. Zhang and X. Zhao, "Machine-learning-based surrogate modeling of aerodynamic flow around distributed structures," *AIAA Journal*, vol. 59, no. 3, pp. 868–879, 2021.

[180] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[181] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry.* Cambridge University Press, 2009.

[182] X. Yang, C. Milliren, M. Kistner, C. Hogg, J. Marr, L. Shen, *et al.*, "High-fidelity simulations and field measurements for characterizing wind fields in a utility-scale wind farm," *Applied Energy*, vol. 281, p. 116115, 2021.

[183] J. Zhang and X. Zhao, "A novel dynamic wind farm wake model based on deep learning," *Applied Energy*, vol. 277, p. 115552, Nov. 2020.

[184] M. F. Howland and J. O. Dabiri, "Wind farm modeling with interpretable physics-informed machine learning," *Energies*, vol. 12, no. 14, p. 2716, 2019.

[185] G. I. Taylor, "The spectrum of turbulence," *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences*, vol. 164, no. 919, pp. 476–490, 1938.

[186] G. K. Batchelor, *An Introduction to Fluid Dynamics.* Cambridge University Press, 2000.

[187] D. Medici, S. Ivanell, J.-Å. Dahlberg, and P. H. Alfredsson, "The upstream flow of a wind turbine: blockage effect," *Wind Energy*, vol. 14, no. 5, pp. 691–697, 2011.

[188] F. Zhao, T. Wang, X. Gao, H. Sun, H. Yang, Z. Han, *et al.*, "Experimental study on wake interactions and performance of the turbines with different rotor-diameters in adjacent area of large-scale wind farm," *Energy*, p. 117416, 2020.

[189] T. K. Barlas and G. A. van Kuik, "Review of state of the art in smart rotor control research for wind turbines," *Progress in Aerospace Sciences*, vol. 46, no. 1, pp. 1–27, 2010.

[190] J. Zhang and X. Zhao, "Spatiotemporal wind field prediction based on physics-informed deep learning and lidar measurements," *Applied Energy*, vol. 288, p. 116641, 2021.

[191] P. Doubrawa, M. J. Churchfield, M. Godvik, and S. Sirnivas, "Load response of a floating wind turbine to turbulent atmospheric flow," *Applied Energy*, vol. 242, pp. 1588–1599, 2019.

[192] D.-Y. Kim, Y.-H. Kim, and B.-S. Kim, "Changes in wind turbine power characteristics and annual energy production due to atmospheric stability, turbulence intensity, and wind shear," *Energy*, vol. 214, p. 119051, 2021.

[193] D. Choi, W. Shin, K. Ko, and W. Rhee, "Static and dynamic yaw misalignments of wind turbines and machine learning based correction methods using lidar data," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 2, pp. 971–982, 2018.

[194] J. Annoni, A. Scholbrock, M. Churchfield, and P. Fleming, "Evaluating tilt for wind plants," in *2017 American Control Conference (ACC)*, pp. 717–722, IEEE, 2017.

[195] P. A. Fleming, A. Scholbrock, A. Jehu, S. Davoust, E. Osler, A. D. Wright, and A. Clifton, "Field-test results using a nacelle-mounted lidar for improving wind turbine power capture by reducing yaw misalignment," in *Journal of Physics: Conference Series*, vol. 524, p. 012002, IOP Publishing, 2014.

[196] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.