The 9th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2024
3–7 June 2024, Lisboa, Portugal

INVESTIGATING THE USABILITY OF PHYSICS INFORMED MACHINE LEARNING APPROACHES FOR WIND FARM PLANNING

Zahra Lakdawala[1] AND Muhammad Waasif Nadeem[1] AND Hassan Kassem AND Martin Dörenkämper[1]

[1] Fraunhofer Institute for Wind Energy Systems
Küpkersweg 70
26129 Oldenburg, Germany
e-mail: zahra.lakdawala@iwes.fraunhofer.de

Key words: PINNS, CFD, atmospheric flow,neural networks, wind resource assessment

Summary. Determining wind energy yield and predicting wind farm performance is challenging due to the uncertain behavior of wind and the need for multiple CFD simulations across various scenarios and parameters, such as tree height, roughness, and wind direction. These simulations are computationally expensive and time-consuming, making Monte-Carlo methods impractical for extensive site assessments.

This work explores the feasibility of using two types of neural networks—data-driven and physics-informed—for wind resource assessment. One network learns from CFD simulation data, while the other is trained directly with the physical equations (RANS and Navier-Stokes). Both networks are configured as feed-forward convolutional neural networks, with a loss function that incorporates residuals from either the data or the physical equations. The L-BFGS optimization algorithm is employed to minimize this loss and determine the network hyperparameters. The network predictions are compared with CFD simulations for two cases: flow over complex terrain and flow over a hill.

The accuracy and limitations of these networks are evaluated by examining the multi-objective loss function, which includes errors from both data and physical equations. The trained networks demonstrate promise in accurately simulating wind resource assessments and offer potential improvements in accelerating CFD setups.

1 Introduction

A Computational Fluid Dynamics (CFD) software tool is a typical workhorse used for predictive simulations in site assessment for predicting the performance of wind farms and understanding the influence of parameter or environmental variations on wind resource, namely wind speed and/or wind energy yield [1, 2]. The CFD-based software is deterministic, in the sense that a single case (with a specified set of input parameters, physics, initial and boundary conditions for a specific terrain) setup will generate a case-specific simulation result [3]. Any change in the case setup will require a new simulation run. Typically, to understand the influence of input variation of parameters, one would require solving for multiple cases using a CFD simulator. Most often, for the real complex terrains, the CFD simulations are computationally expensive,

due to which its use in practice is rather limited. Due to time and cost constraints, practitioners are often assessing sites and wind resource based on inaccurate and/or limited number of cases predictive runs. Industry is constantly on the lookout for better, faster workflows that can assess wind resource[4]. Keeping this in mind, this work investigates AI-driven approaches, particularly training of two different neural networks, for enhancing CFD work flows for evaluating wind resource for complex terrains. This work is driven towards faster and reliable wind resource assessment facilitated through the training of neural networks.

This work looks into two different kinds of training mechanisms for neural networks using a feed-forward convolutional network [5, 6], namely data-driven and physics-informed neural networks[7, 8]. data-driven network relies on training data that come from simulation results from CFD, whereas the physics-informed network directly incorporates physical models used for CFD instead of the simulation results. The feasibility of the two networks is investigated for two benchmarks that are introduced in Section 2 for site assessment. Section 3 describes how different mechanisms of training of networks can be incorporated in the loss functions. Configurations for both networks is described are data-hungry and rely on the network configurations. The question on how predicted estimates from trained networks compare to CFD simulation results is addressed in Section 4. We shed some light on ways in which the trained networks can be used to facilitate the CFD toolchain. Section 5 discusses the potential use of trained networks such as generating precursors/initial states for CFD simulation speed up to enhance run-time for new cases/changes in input parameters.

2 Problem setup

In this section, we set up two benchmarks to investigate the feasibility of physics informed networks. Both benchmarks use OpenFOAM-based CFD setup/simulations for generating training data/physics and providing reference solutions (for wind speed) for validating the networks described in Section 3.

The first use case is constructed to assesses the impact of forest height variation and wind direction on wind speeds on a complex terrain in the region of Baden Wuerttemburg, Germany. The objective is to use a subset of CFD simulated estimates of wind speed to train the neural network and assess the feasibility of the data-driven neural network, discussed in Section 3.1.

The second use case serves as a a classical benchmark for wind flowing through a flat terrain including a small bump. The objective is to use this as a validation benchmark to assess the feasibility of the physics-informed neural network, discussed in Section 3.2.

2.1 Use case 1

The use case presented here serves as an validation benchmark for the data-informed network described in Section 3.1. This use case is set up to understand how canopy foliage, particularly its height, impacts the variability in wind speed in complex terrains. Several cases with a forest height variation, for example five heights varying from $5 - 25$m, each with 36 states/wind directions $0 - 36\,\text{deg}$, is overlay on the terrain. The time-averaged Reynolds averaged Navier-Stokes equations are used to describe the flow in the terrain. The workflow is shown in Figure 1. In this work Fraunhofer IWES Wind Simulation Environment (FIWIND) has been used based on OpenFOAM-v2306 version. The core model physical model was first introduced in [1].

FIWIND has been used and validated in several recent studies [2, 9, 10] [1].FIWIND is used to generate results for $36 \times 5 = 180$ cases of CFD simulations to assess the forest height variation and wind direction on wind speeds. The results from CFD simulations are then post processed by a customized interface for assessing wind resource.

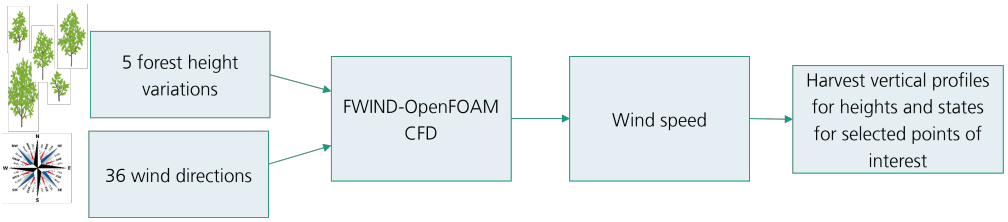


Figure 1: A deterministic workflow for CFD simulation tool to study the impact of forest height on wind speed in a chosen complex terrain.

The results from the CFD-setup are full flow field variables (wind speed and pressure) on 2D/3D domain. One could potentially select certain points of interest and harvest vertical profiles to closely assess the variability of wind speed for different forest height simulations for different wind directions.

2.2 Use case 2

The use case presented here serves as an validation benchmark for the physics-informed network described in Section 3.2. CFD simulations are use to estimate flow in a simple channel-like domain. Since the physical model and boundary conditions are used to set up the training of the physics-informed neural network, we will formally sketch out the set up of this benchmark in detail.

The two-dimensional steady state, time-independent Navier-Stokes equation governing laminar, incompressible, isothermal fluid flow is used to govern the flow in the interior of the domain Ω:

$$e_1 : \quad \nabla \cdot \vec{u} = 0 \tag{1}$$

$$e_{2,3} : \quad -\nabla \cdot (\mu \nabla \vec{u}) + (\rho \vec{u} \cdot \nabla)\vec{u} + \nabla p = f \tag{2}$$

Here e_1 denotes the continuity equation, e_2 and e_3 denote the momentum equations for velocity components $\vec{u} = (u, v)$. p denotes the pressure, whereas ρ and μ denote the density and viscosity of the fluid respectively. The equations are equipped with boundary conditions, usually of Dirichlet or Neumann type. A 2D channel with a small bump is considered as the domain of interest as shown in Figure 2. At the inlet boundary, Dirichlet inflow velocity $\vec{u} = (u_{in}, 0)$ is prescribed. At the outflow and top boundary Dirichlet pressure $p = 0$ and Neumann velocity $\frac{\delta \vec{u}}{\delta \mathbf{n}}$ is prescribed. The bottom boundary has no-slip conditions, i.e. $\vec{u} = (0, 0), p = 0$ prescribed.

[1] FIWind is developed in Fraunhofer IWES. It includes pre and post processing interfaces to have more flexibility in setting up site assessment problems (domain, mesher, and setting up the input parameters/boundary conditions and doing some precursor runs). The CFD kernel is based on OpenFOAM - the main physical models are NS, RANS and LES.

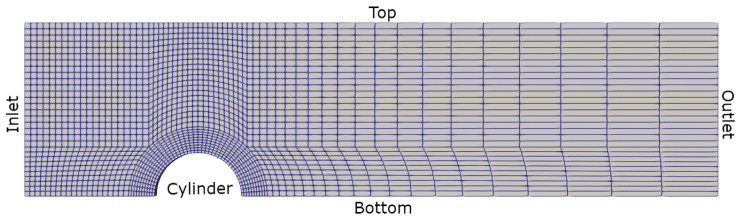


Figure 2: Mesh defined with number of cells; Internal: 2000, cylinder: 40, input: 30, output: 30, top: 60, bottom: 60

3 Feed-forward convolutional neural network

This section introduces a simple forward-feed convolution neural network (FFCNN) used for training both data-driven and physics-informed networks. Further details on those networks are provided in the subsequent subsections.

A multilayer FFCNN is primarily defined by three components: multiple hidden layers consisting of neurons, and a global architecture that included both input and output layers. If the weight w_{jk}^l connects the k-th neuron in the $(l-1)$th layer to the j-th neuron in the l-th layer, the relationship for the output $\tilde{u}_j^l$ can be expressed as follows:

$$\tilde{u}_j^l = \sigma\left(\sum_k w_{jk}^l \tilde{u}_k^{l-1} + b_j^l\right) = \sigma\left(z_j^l\right).$$

Each neuron in a layer is connected to each neuron of the next layer, with each connection associated with a weight (w), a bias term (b), and an activation function(σ). The activation function is applied to a signal at each layer before passing it to the next layer.[2] The choice of the activation function typically depends on factors such as explosion or vanishing gradients, ensuring clear predictions, and achieving computational efficiency. In our case studies *tanh* activation function delivered the best estimates.

Figure 3 shows a data-driven feed forward network with a 2-node input layer and an output layer. In principle, a network can be trained using data, rules/physics or a combination of both data and physics.

The goal is for the network to learn how to classify solutions/patterns/relationships based on the information in the provided input-output pairs. To train a network using a backpropagation algorithm, its performance must be quantifiable and measurable. This is done by defining a loss function that evaluates how changes in network parameters (weight and biases) affect the output error, comparing the network's output to the desired target solution (whether from training data or physical model). Training of a neural network is thus a multi-objective optimization problem, where the cost or loss function, typically expressed as mean squared error, needs to be minimized. A neural network requires an optimization routine to adjust its weights and biases in a way that minimizes the loss function defined on its output. This optimization is typically performed using a (stochastic) gradient descent algorithm or a quasi-Newton method. Training is considered complete when the network finds parameters that achieve the desired accuracy of the loss function.

[2]For instance, a sample input u with a *sigmoid* activation function undergoes a transformation of $\sigma(u) = \frac{1}{1+e^{-u}}$ before reaching the next layer.

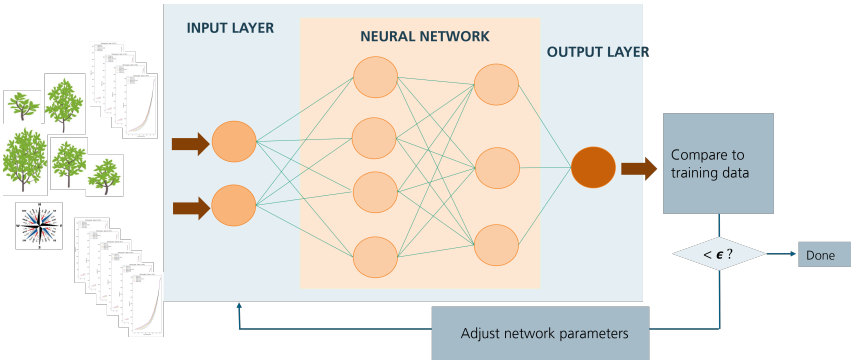


Figure 3: A simple feed forward convolution neural network configured with 2-node input layer, 2 hidden layers (with 4 and 3 nodes per layer) and a single node output layer.

Simply put, the network parameters θ (weights and biases) are optimized so that the network's output closely matches the the desired target solution. The loss functions that account for the error terms in both data-driven and physics-informed are described in the following subsections 3.1 and 3.2.

3.1 Data-driven neural network

A data-driven is denoted by $U(\mathbf{x}, \mathbf{u}^*; \theta)$, where $\mathbf{x}$, u^* and θ represent a vector of spatial coordinates, the known values and the network parameters respectively. The network U is trained using the input-output pair $(\mathbf{x}, \mathbf{u}^*)$, with the goal of learning to classify patterns between these pairs.

The objective of the training process is to find the network parameters that minimize the error or loss between the network's predicted values and the provided training data. This error is typically expressed as the mean squared error (MSE), which is calculated by taking the difference between the network's output $\tilde{\mathbf{u}}(\mathbf{x})$ and the desired target value $\mathbf{u}^*(\mathbf{x})$):

$$\mathcal{L}_{\text{data}} = \mathcal{L}_{\text{data}}\left(\theta \mid (\mathbf{x}, \mathbf{u}^*)\right) = \frac{1}{N}\sum_{i}^{N} \|\tilde{u}(x_i) - u^*(x_i)\|_2^2 \,. \tag{3}$$

Here, $\mathcal{L}_{\text{data}}$ denotes the loss function for a network trained using N input-output data pairs. The predicted output of the network $\tilde{u} = \tilde{u}(\mathbf{x}, \theta)$ is expressed as a function of the input $\mathbf{x}$ and the network parameters θ. The training process aims to find θ, given the input $(\mathbf{x}, \mathbf{u}^*)$, by solving the following optimization problem

$$\text{argmin}_\theta \mathcal{L}_{\text{data}}\left(\theta \mid (\mathbf{x}, \mathbf{u}^*)\right),$$

where the function $\mathbf{u}^*$ maps $\mathbf{x}$ to measurements/true solution at those coordinates. The learned function is based just on the training input data $\mathbf{u}^*$ and $\mathbf{x}$.

3.2 Physics informed neural network

Similar to the data-driven network, a physics-informed network can be represented by $U(\mathbf{x}, physics; \theta)$, where the network is trained using physical models or rules, rather than data, defined for the

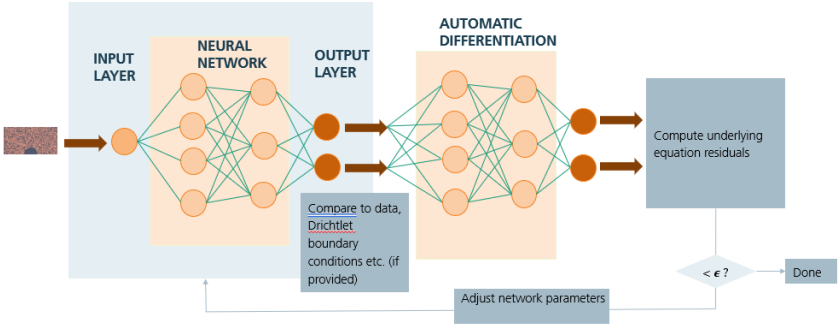


Figure 4: A simple feed forward convolution neural network configured with 2-node input layer, 2 hidden layers (with 4 and 3 nodes per layer) and a single node output layer.

spatial coordinates. When seeking the steady state solution of a boundary value problem, the partial differential equations and boundary conditions defined over the spatial domain (both interior and external boundaries) can be incorporated into the loss function. In this approach, the physical equations evaluated at interior points along with boundary conditions, serve as the training data. The function that the network learns is the solution to the physics model, denoted by $\tilde{\mathbf{u}}(\mathbf{x})$.

The FFCN for physics-informed training is similar to the the one described for data-driven networks in Section 3.1. However, because partial differential equations and boundary conditions involve derivatives, the FFCN is integrated with a gradient layer. This gradient layer takes the entire data-driven network as input and produces derivatives of the network's approximation to the solution of the partial differential equation. The derivatives of the output $\tilde{\mathbf{u}}$ are then used to calculate the strong residuals of the partial differential equation. The norm of these residuals, along with norms of residuals of the boundary conditions, contributes to the loss functions. Figure 4 illustrates a data-driven two-layer FFCN with a 2-node input layer and an output layer, coupled with a gradient layer.

The goal of the training process is to adjust the network parameters so that the network accurately learns the solution to the given physcial model. This involves incorporating the residuals of the partial differential equations and boundary conditions into the loss function, ensuring that the error between the network's predictions and the physical model is minimized. The partial differential equations are expressed in parameterized form, and the loss function is formally defined as follows:

$$\mathcal{L}_{physics} = \mathcal{L}_{physics}(\theta|\ (\mathbf{x}, physics, \mathbf{u}^*)) = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}}. \tag{4}$$

The first term of the loss function, $\mathcal{L}_{\text{PDE}}$ calculates the norms of the equation residuals. Specifically, $\mathcal{L}_{\text{PDE}}$ represents the mean squared error between the equation residuals and zero, as all the equation terms are collected on the left side. The goal is to minimize these residuals towards zero. A finite set of residual points, where the equations are penalized, can vary in number and be located throughout the domain, whether structured or unstructured. These points are known as collocation points. Let N denote the number of collocation points for a set

of N_{eqs} equations. Then,

$$\mathcal{L}_{\text{PDE}} = \sum_{j=1}^{N_{eqs}} \frac{1}{N} \sum_{i=1}^{N} \|e_j(x_i)\|_2^2. \tag{5}$$

Dirichlet boundary conditions are included in $\mathcal{L}_{\text{BC}}$ which measures the mean squared of error between the prescribed boundary conditions, denoted by u^*, on each boundary of the domain and the model predictions, denoted by $\tilde{u}$. Let there be N_{bdry} boundaries (such as solid, inlet, outlet, etc.), denoted by Γ_j, with each boundary having M_j points. Then

$$\mathcal{L}_{\text{BC}} = \sum_{j=1}^{N_{bdry}} \frac{1}{M_j} \sum_{x=1}^{M_j} \|\tilde{u}(x_i) - u^*(x_i)\|_2^2 \,.$$

Neumann or Robin boundary conditions, which involve derivatives, are incorporated as residual penalties in a manner how equations are inccluded in Equation 5).

DDNN uses a single-objective loss function, whereas PINNs optimize training parameters by minimizing a multi-objective loss functional $\mathcal{L}_{physics}$.The training process of PINNs addresses the following optimization problem:

$$\text{argmin}_\theta \mathcal{L}(\theta \mid (\mathbf{x}, physics, \mathbf{u}^*)).$$

4 Results

The results for the two setups described in Section 2 are presented here for trained networks discussed in Section 3. In the first use case, CFD simulation results are used to train a data-driven neural network. In contrast, the second use case employs CFD physics to train a physics-informed neural network to solve the steady-state Navier-Stokes equations.

4.1 Use Case 1

A site in Baden-Württemberg, featuring an elevation map ranging from 0 to 800m is considered, as shown in Figure 5 (left), is considered. With a fixed mesh resolution of 60m, the terrain is overlaid with a forest canopy model that has a constant leaf area density of 0.07, as shown in Figure 5 (middle). The red-shaded area represents the forested region. The forest height variation is sampled using a Gaussian distribution with a mean of 15m and standard deviation of 5m.

The FIWIND toolchain is used to simulate the flow over this terrain for five different forest heights and 36 distinct wind directions. Figure 5 (right) shows the wind speed estimated for one configuration (forest height = 15m and wind direction = 225 degrees). Four points of interest within the domain, marked by red dots in Figure 5 (right), are selected, and the wind speed profiles are extracted for heights ranging from 0-400m. Figure 6 displays the wind speed profiles at one of these points for four different wind directions (45, 135, 225 and 315 degrees). Each plot shows five different profiles corresponding to forest heights of 5,10,15,20 and 25m. The state-forest profiles are divided into training and testing sets, with the testing data comprising profiles corresponding to 15m forest height and wind directions of 45, 135, 225, 315 degrees. The remaining data is used to train the data-driven network described in Section 3.1.

Various architectures, as shown in Figure 7, are evaluated to determine the optimal configuration for this training data. The best-performing network architecture with the least mean

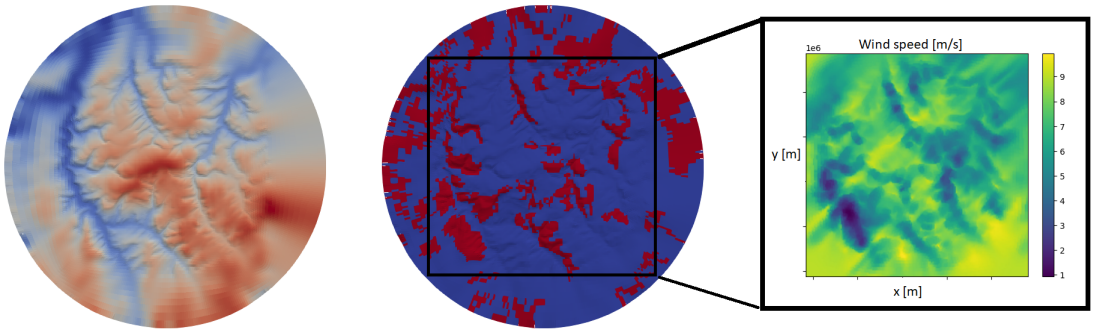


Figure 5: Left: Elevation map for a site in Baden Württemberg. Middle: Frorest cover marked in red. Right: CFD estimated wind speed at 100m hub height.

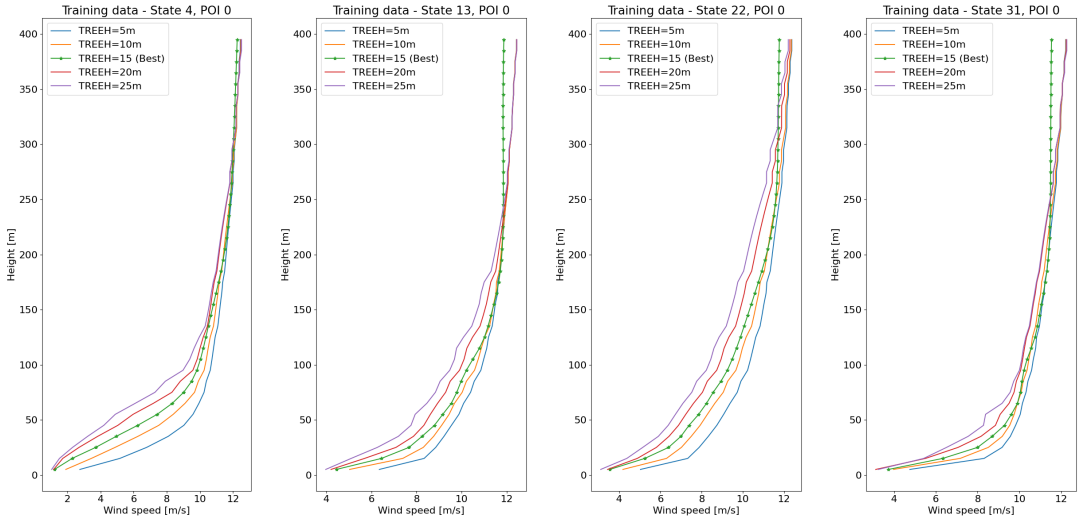


Figure 6: CFD simulated wind speed profiles along a height of 0-400m. Each plot shows five profiles, each corresponding to a forest height. A subset of these column profiles are used to train the DDNN.

squared error, labelled 'Arch 1', consists of 9 hidden layers with 50 nodes each and uses $\sigma = tanh$ activation function.

Wind speed estimates from the best-performing DDNN are validated against CFD results for the testing set with a forest height of 15m, as shown in Figure 8. The DDNN predictions closely match the CFD simulations, with an error bound of $O(10^{-1})$.

4.2 Use Case 2

A channel with dimensions of 80x20m, featuring a semi circular hill with a 5m radius located at 20m from the inlet, is set up to replicate flow over a flat terrain with a hill. The governing physics of laminar flow in this domain is described by the Navier-Stokes equations, as discussed in Section 2.2. The specific boundary conditions with inlet velocity of $\vec{u} = (0, 1)$ m/s, outlet

Model	Layers	Nodes	σ
Arch 1	9	50	tanh
Arch 2	5	100	relu
Arch 3	7	10	relu
Arch 4	5	10	tanh
Arch 5	3	10	tanh

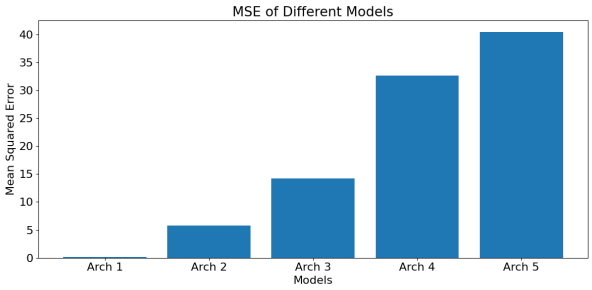


Figure 7: Mean squared error for different network architectures.

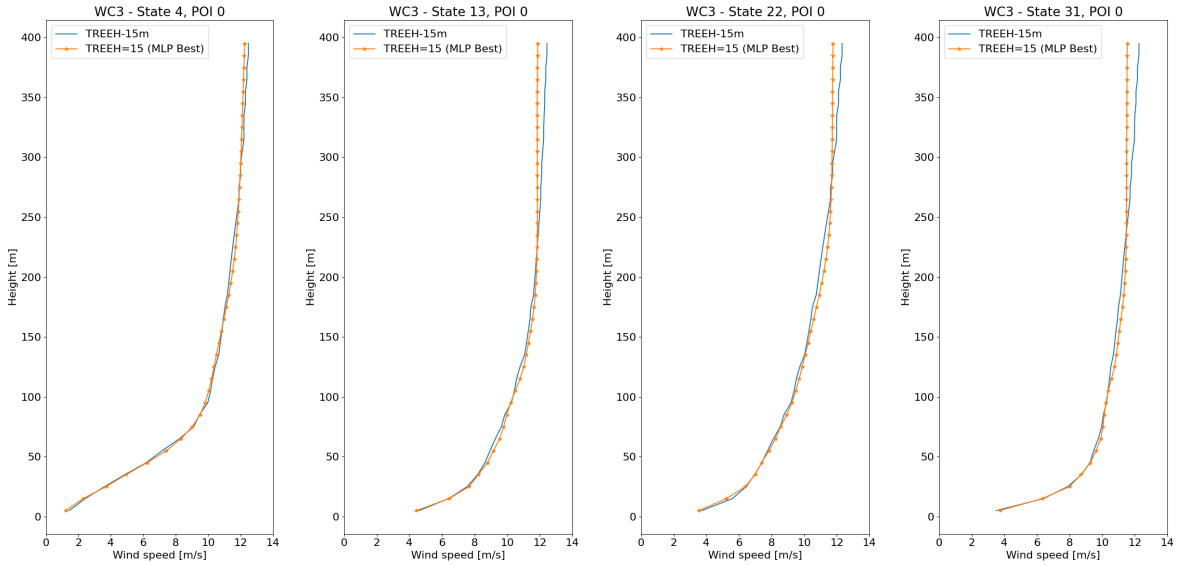


Figure 8: Predictions from data driven neural network are compared against CFD solutions for a selected point of interest for four states and forest height 15m.

zero-pressure and no-slip conditions are prescribed for the bump and the top and bottom walls. The equations together with the boundary conditions are used to formulate the residual loss function for training the network described in Section 3.2. The training data consisted of spatial grid points within the domain's interior and on its boundaries. For the results shown in Figure 9 (left), 6400 points were uniformly distributed throughout the channel interior, with 640 points on both the inlet and outlet, and 1280 points on both the top and bottom boundaries. NVIDIA's Modulus framework [11] is used to set up the domain, and the physical model and boundary conditions are imposed on the collocation points.

The loss function is crucial for assessing the performance and accuracy of the trained network. In PINNs, the loss is residual-based and its expected that the loss function would reach zero. The loss functions are formulated and the physics-informed solver in Modulus is invoked to train the network. A closer examination of the loss function, as shown in Figure 9 (right), provides valuable insight into the learning process for PINNs.

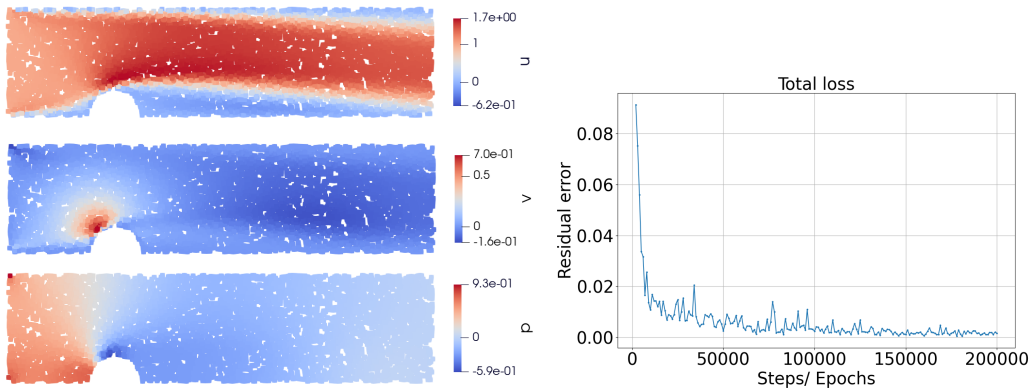
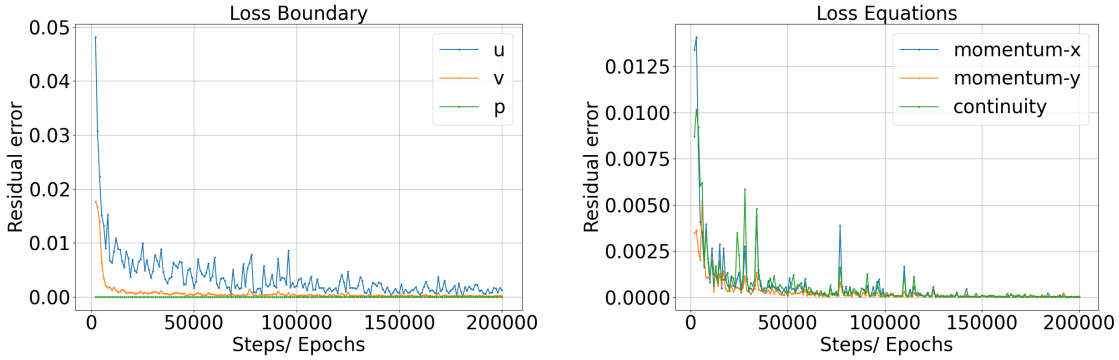


Figure 9: Left: Velocity and pressure profile estimates using PINNs. Right: Convergence of the loss function that is used for training the network.

Recall that each equation in the physical setup, along with its boundary conditions, contributes its own loss term, and the sum of these terms forms the overall loss function. Figures 10a and 10b show the convergence of the residual loss function over 200000 epochs for the Dirichlet boundary conditions and the physical equations. Constraining the horizontal velocity (at the inlet and walls) and ensuring continuity proved to be more challenging compared to the other terms.



(a) Loss for boundary points

(b) Loss for PDEs

Figure 10: Monitoring network training for use case 2 through loss functions. Left: Convergence of the residuals for the Dirichlet boundary conditions on inlet, walls and outlet. Right: Convergence of the residuals for each of the equations (momentum and continuity).

For validation, OpenFOAM (icoFoam) software is used to simulate the same setup, utilizing the computational mesh described in Section 2.2. The simulated velocity and pressure profiles from OpenFOAM are shown in Figure 11 (left). The errors between the OpenFOAM and PINN estimates, shown in Figure 11 (right), indicate that the velocity and pressure profiles predicted by the PINN closely match the CFD simulation results, with errors within $O(10^{-1})$. Additionally, the 'wake' effect and recovery caused by the hill are well captured by the PINN.

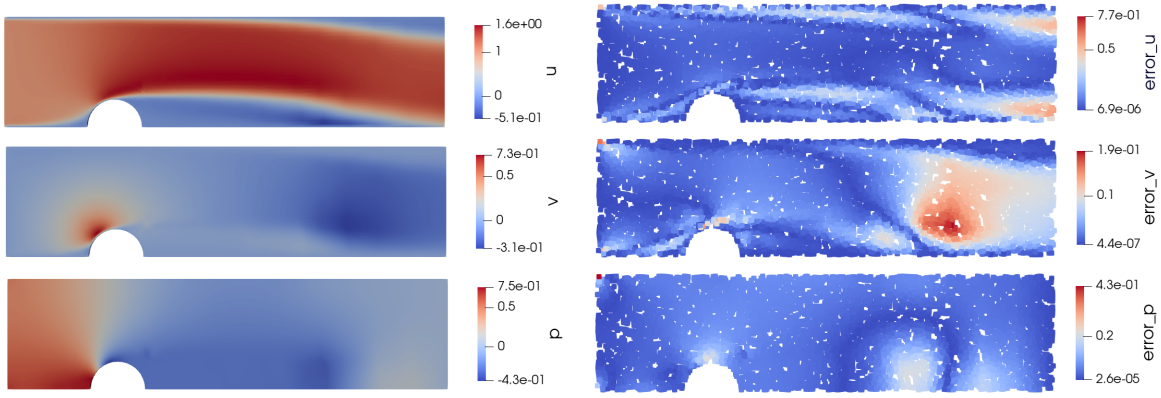


Figure 11: Left: Velocity and pressure profile estimates using OpenFOAM setup. Right: Relative error between the OpenFOAM (reference) against PINNs estimates.

5 Conclusion

This work explores the feasibility of using physics-informed machine learning approaches for wind resource assessment. Two benchmarks are established to validate the performance of both data-driven and physics-informed neural networks. The network predictions are then compared with CFD simulation results for validation.

The data-driven benchmark utilizes CFD simulations from the FIWIND toolchain as training data. The CFD results include 180 simulations for studying the impact of forest height variation on wind speed, out of which 160 are used to train the network. The trained network is used to estimate the wind speed for the 20 cases that are not included in the training data set.

The physics-informed benchmark involves a setup with flat terrain and a hill, where the flow is governed by the Navier-Stokes equations. This setup, along with the physical model, is implemented within NVIDIA's Modulus framework. The network estimates are compared to CFD simulations performed using OpenFOAM.

For both benchmarks, the network predictions were validated against the CFD simulation results, achieving accuracy within 10^{-1}. It can be concluded that network predictions are influenced by the amount of training data, specifically the number of input-output pairs used during training. The results suggest that AI-powered network estimates offer significant benefits. In the wind industry, accurately assessing wind resources typically requires a large number of CFD simulations, which can be computationally expensive. Neural networks can be trained on specific benchmarks or historical data from legacy sites. Once fully trained, these networks can provide real-time results comparable to those of conventional CFD simulations. This capability could enable resource assessment teams and wind farm planners to run more simulations in shorter timeframes, providing quicker insights and improving efficiency.

Acknowledgement

The authors would like to thank and acknowledge the support of Federal Ministry for Economic Affairs and Climate Action (BMWK). This work is done as part of the "Simulation uncertainties for the detailed assessment of wind energy yield (SUnDAY)" project funded by BMWK (grant no. 03EE2010). The simulations were partly performed at the HPC Clus-

ter STORM, located at the University of Oldenburg (Germany) funded by BMWK (grant no. 03EE3067A-B).

REFERENCES

[1] Chi-Yao Chang, Jonas Schmidt, Martin Dörenkämper, and Bernhard Stoevesandt. A consistent steady state cfd simulation method for stratified atmospheric boundary layer flows. *Journal of Wind Engineering and Industrial Aerodynamics*, 172:55–67, 2018.

[2] Siamak Akbarzadeh, Hassan Kassem, Renko Buhr, Gerald Steinfeld, and Bernhard Stoevesandt. Adjoint-based calibration of inlet boundary condition for atmospheric computational fluid dynamics solvers. *Wind Energy Science*, 4(4):619–632, 2019.

[3] Randall J. LeVeque. *Numerical methods for conservation laws*. Lectures in Mathematics ETH Zürich. Birkhäuser Verlag, Basel, second edition, 1992.

[4] Stefan Barber, Alexander Schubiger, Sandro Koller, Daniel Eggli, Alessandro Radi, Andrea Rumpf, and Heinrich Knaus. The wide range of factors contributing to wind resource assessment accuracy in complex terrain. *Wind Energy Science*, 7(4):1503–1525, 2022.

[5] Guangle Yao, Tao Lei, and Jiandan Zhong. A review of convolutional-neural-network-based action recognition. *Pattern Recognition Letters*, 118:14–22, 2019. Cooperative and Social Robots: Understanding Human Activities and Intentions.

[6] Anamika Dhillon and Gyanendra K. Verma. Convolutional neural network: a review of models, methodologies and applications to object detection. *Prog Artif Intell*, 9:85–112, 2020.

[7] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 2019.

[8] Souvik Chakraborty. Transfer learning based multi-fidelity physics informed deep neural network. *Journal of Computational Physics*, 426(109942), 2021.

[9] Julia Gottschall, Alkistis Papetta, Hassan Kassem, Paul Julian Meyer, Linda Schrempf, Christian Wetzel, and Johannes Becker. Advancing wind resource assessment in complex terrain with scanning lidar measurements. *Energies*, 14(11), 2021.

[10] Zahra Lakdawala, Hassan Kassem, and Jonas Schulte. Enhancing standard cfd based practices for site assessment through polynomial surrogates for estimating uncertainty in wind speed. *Journal of Physics: Conference Series*, 2745(1):012016, apr 2024.

[11] Modulus Contributors. Nvidia modulus: An open-source framework for physics-based deep learning in science and engineering, 2024.