Hybrid Computational Fluid Dynamics (CFD) and Machine Learning for Airflow Simulation Over Coastal Terrain.

A Thesis Submitted in Partial Fulfilment of the Requirements for the MSc. Applied Data Science Degree.

James Quigley

June 2024

Advisors: **Dr. Saeb Faraji Gargari** and **Oriol Pomarol Moya**

Abstract

In recent years, Physics Informed Neural Networks (PINNs) have emerged as a powerful tool for solving complex partial differential equations (PDEs) governing physical phenomena. While numerous studies have explored the theoretical aspects of PINNs using benchmark problems, their application to real-world data and simulations remains limited. In this paper, we employ the PINNs method to simulate fluid flow over coastal dunes in the Netherlands, addressing a real-world problem. We aim to test the usability and performance of Physics Informed Loss by comparing it to similar models without this loss. Our results demonstrate that models incorporating Physics Informed Loss do not improve performance within the training data bounds but do show enhanced generalization to unseen data.

The source code used in this project will be made available on Github.

Contents

1 Introduction

Understanding wind patterns around sandy beaches and their dunes is vital. It helps us predict how sand moves, which in turn affects the size and shape of the dunes. Sandy beaches are a major feature of our planet, covering 31% of ice-free coastlines Luijendijk et al. (2018). These beaches and dunes play a crucial role in coastal protection. They provide habitat for plants and animals, support tourism, and help purify water (Edward et al., 2011; Everard et al., 2010). Much of the Netherlands lays below sea level and thus coastal dunes also play a key role in preventing storm water from flooding the land and separating seawater from fresh drinking water. Regular exposure to high-speed wind can quickly change the size and shape of sand dunes and eliminate their protective functions.

The ability to predict the wind speed and direction can enable researchers to protect against erosion and deposition of sand and therefore protect the functions of the dunes (Van Koningsveld et al., 2007). Protective actions have been taken at multiple sites in the Netherlands to strategically help grow sand dunes and ensure their long-term existence. Foredune notching is one example where deep ridges are cut into the first row of dunes, known at the sea strip, allowing wind carrying sand to travel further inland and deposit it’s load on the dunes behind. This has been performed at Kennemerduinen, NL, between 2011 and 2013. The foredune notching resulted in the main row of dunes behind the foredunes growing at a rate of 30-50m per year, becoming larger and firmer (Ruessink, 2021). According to PWN Waterleidingbedrijf Noord-Holland, the public organization responsible for the works, the larger-more inland dunes are responsible for keeping the water back. Similar work is planned for the dunes at Castricum, NL, during 2024 and 2025.

The use of high-fidelity Computational Fluid Dynamics (CFD) simulations remains a reliable but time consuming and computationally expensive method for predicting the surface wind speed and direction over coastal dunes for a given atmospheric wind speed. The accuracy of traditional CFD is limited by the computational resources available. In the pursuit of making faster predictions, surrogate CFD models using Machine Learning (ML) have been developed. Such CFD-ML model have been shown to reduce computation time and to solve inverse problems (determining the underlying physical equations)(Raissi, Perdikaris, et al., 2019; Raissi, Wang, et al., 2019), and to upscale the resolution of CFD simulations (Gao et al., 2021).

Hybrid CFD-ML Surrogate models are a class of ML models which are trained on data from traditional CFD simulations with the hope that the ML models can learn the patterns and non-linear interactions of CFD. While slow and costly to train, surrogate CFD-ML models aim to make predictions quick and efficiently. Many possible surrogate CFD-ML models exist. Convolutional Neural Networks (CNNs) being designed specifically for processing images are a natural network architecture for CFD since a 2D or 3D vector field can be presented as 2 or 3 input channels (images), which can then be convolved and deconvolved within a CNN(Guo et al., 2016). Recurrent Neural Networks(RNNs) have also shown promising results in turbulence modelling(Zhang et al., 2023).

Physics Informed Neural Networks (PINNs) are another potential surrogate model for solving systems of differential equations. PINNs involve training a neural network to minimize the loss function defined by residual of the modelled equations. In their seminal paper, Raissi et al. (2017) showed that Neural Networks could be trained to learn differential equations and in their follow up paper Raissi, Perdikaris, et al. (2019) showed that PINNs could lean the 2D Navier-Stokes(NS) equations for an incompressible fluid flow around a cylinder. By encoding the continuity NS equation and the momentum NS equations into the loss function, they were able to train the NN to make predictions which satisfy the governing laws.

However, training PINNs to model CFD simulations remains mostly in the realm of theoretical research. The use of Automatic Differentiation (AD) of the outputs of the NN to compute the partial derivatives of a PINN remains a barrier to the development of the field. AD relies on the ability to decompose a function into a series of operations (primitives) for which the derivatives are known and to which the chain rule can be applied (Merriënboer et al., 2018). Most modern ML frameworks including PyTorch, TensorFlow and JAX include the ability to perform AD. Nevertheless, the use of AD is computationally and memory intensive as large graphs employing the chain rule are generated to calculate the derivate of the NN outputs with respect to the inputs of the NN and its parameters.

When designing a PINN to model the NS equations, a Deep Neural Network (DNN) architecture is often chosen as these provide sufficient network depth and complexity to model the interactions of the fluid. However, DNNs have a high number of parameters even for low-resolution, small-scale fluid simulations. Moving from simulating the NS equations from 2D to 3D domain dramatically increases the size of the network required as well as additional derivatives and an additional NS equation. There are notably few articles which apply PINNs to the problem of real-world 3D fluid flows. Studies which do attempt to solve the problem do so by modelling on extremely small scales (Arzani et al., 2021), use multi-scaling techniques (Suo & Zhang, 2023) or employing Physics Informed Graph Neural Networks (Shao et al., 2023). The use of Graphs instead of regular lattices in PINNs are an obvious choice since this is standard practice in traditional CFD simulations. However, introducing an unstructured mesh introduces additional complexity to modelling and the results would require interpolation back to the regular lattice for comparison. Due to these drawbacks, a regular lattice structure will be used for the model proposed in this paper.

In their paper on charactering the failure modes of PINNs, Krishnapriyan et al. (2021) suggest using curriculum regularization to improve performance. Curriculum regularization is achieved by gradually increasing the PIL parameters as the model trains until the PINN converges correctly. Before starting curriculum regularization, they suggest creating a NN without physics informed loss while using all available training data to show that the model has sufficient capacity/expressivity . They then argue that the cause of PINN failure is because of optimization difficulties and curriculum regularization is the solution they present.

Hybrid CFD-ML requires training an ML model on CFD data. Although it is possible to develop PINNs without any data (Stiasny et al., 2021; Sun et al., 2020), this is not usually performed due to extremely slow convergence. The creation of an effective surrogate model for determining wind flow over coastal dunes could enable land management agencies to replace time consuming CFD simulations with quick feed-forward neural networks. Such a model could be used to inform erosion and deposition models to predict how dunes will develop over time and find the quickest or most cost-effective foredune notching solution if required.

In this study, we attempt to create a deep neural network to model wind flow over coastal terrain. We aim to characterize the performance of Physics Informed Learning(PIL) in simulating wind velocity and investigate whether NNs can improve the computation efficiency compared to classical CFD simulations.

Chapter 2.1 introduces the terrain available for simulation and presents a brief overview of the CFD performed, including the assumptions, limitations and boundary conditions. Chapter 2.2 presents the conceptual model of our PINN and Chapter 2.3 introduces data scaling and the a Fourier Transformation Layer to improve model performance. Chapter 2.4 demonstrates how to optimize performance by tuning the model parameters. In Chapter 3 we present the model accuracy, training time, and test its generalizability based on unseen data. In Chapter 4 we discuss the results and conclude the research question.

2 Data and Methods

2.1 CFD Simulation

For this study data, including velocity vector and pressure, is generated using OpenFOAM, an open-source package for CFD simulation ("OpenFOAM," 2024; Weller et al., 1998). The simulation assumes an incompressible fluid flow in isothermal equilibrium. A no slip wall boundary condition is imposed on dune ground. Specific hyperbolic velocity profile is used at inlet boundary. Zero pressure gradient is used on the outlet boundary and symmetry condition are imposed on the side boundaries. For the top boundary, slip-wall boundary is assumed. The simulation was performed using a high-resolution graph (mesh) stretching a region of 700m x 300m and the results were interpolated to a regular X-Y grid with 1m spacing. Figure 1 illustrates the terrain used in the simulation. The air velocity and pressure were reported after two thousand simulation time iterations at a height (Z-direction) of 1 m above the surface. The simulation was repeated for wind inlet angles ranging from -70° to +50° at 10° intervals., with the 0° angle being defined as the direction the wind makes if it is pointing directly into the coast.

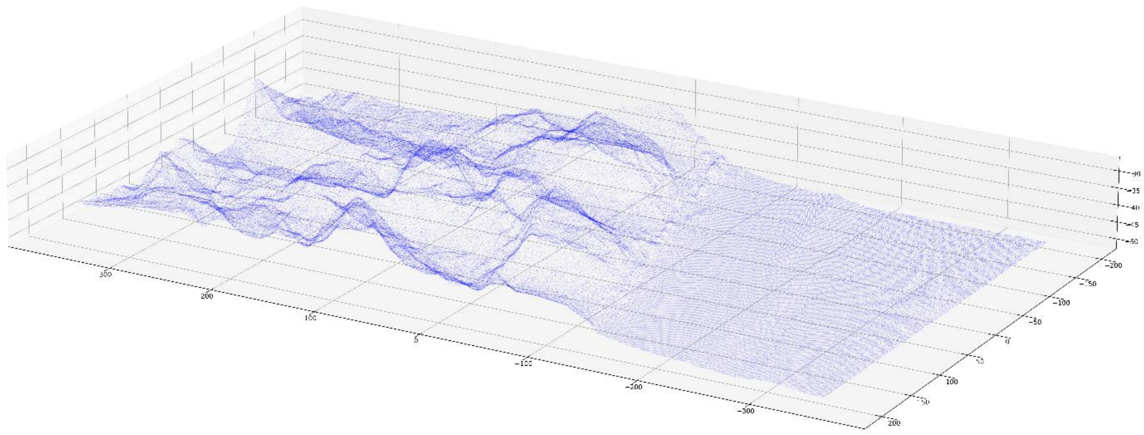

Figure 1: Simulated Terrain. All axis units are in meters, but the scale of the z-axis is exaggerated for illustrative purposes.

It should be noted that the transient fluid flow was reported at one arbitrary snapshot in time. It is assumed that the air reached an almost steady state condition by the end of the simulation and therefore the use of the steady state Navier-Stokes equations when training a PINN is reasonable. In addition, the angle between the normal surface vector $\vec{n}$ and the wind inlet direction φ is calculated as $\theta = \cos\,(\varphi - \vec{n})$. Figure 2 illustrates the boundary conditions as well as the calculated normal surface angle, θ.

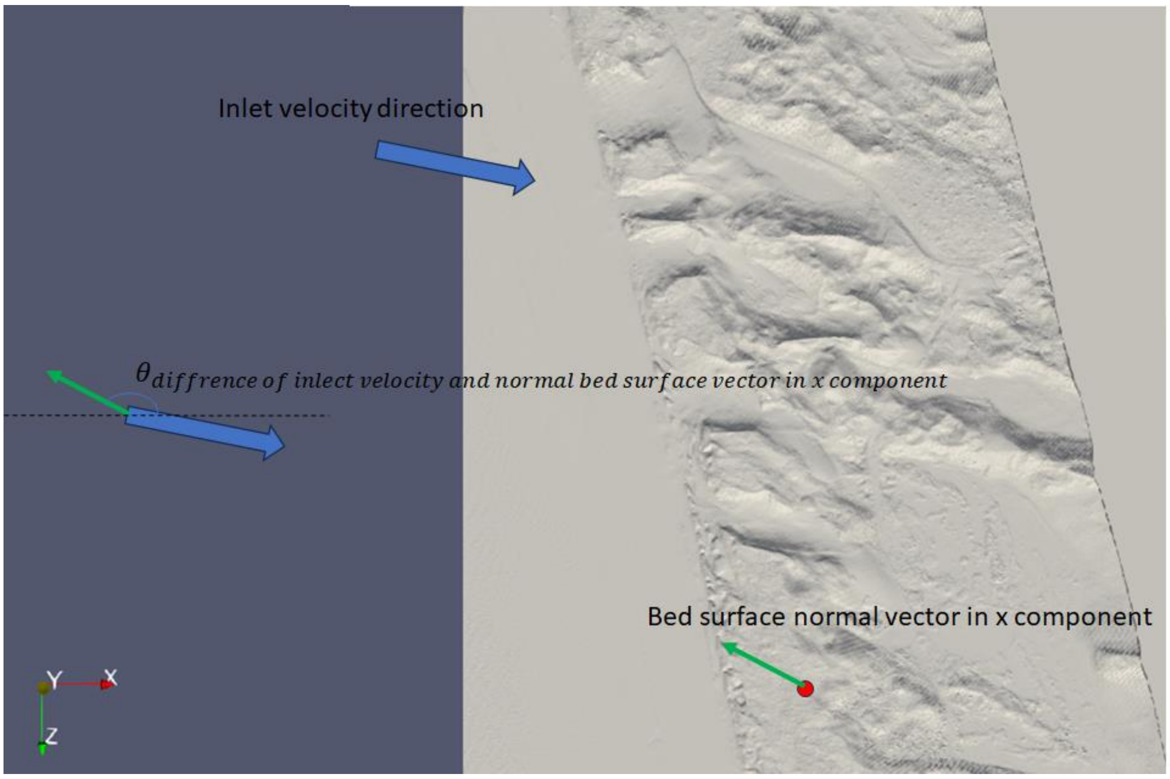
Inlet velocity direction
$\theta_{diffrence\ of\ inlect\ velocity\ and\ normal\ bed\ surface\ vector\ in\ x\ component}$
Bed surface normal vector in x component
Y
X
Z

Figure 2: Illustration of CFD boundary conditions and the calculated normal surface angle, θ

2.2 Physics Informed Loss

Normally NN optimizers minimize a loss function that is an error metric between observations and predictions. Back propagation using labelled data (supervised learning) is traditionally used by the optimizer when training a NN. For brevity the details of backpropagation are not covered in this paper but can be found in the literature (Hecht-Nielsen, 1992). PINNs add another term to the loss function corresponding to the physical laws, in our can the Navier Stokes equations. Using Automatic Differentiation(AD) to compute the partial derivatives with respect to the input coordinates it is possible to train a NN to simultaneously minimize residuals of the governing equations while also minimizing the loss due to data. The steady state, incompressible Navier Stokes equations in 3D under the influence of a gravity force g, with fluid viscosity ν and time dependent turbulent viscosity ν_t are given in Eq 2.1 - 2.4. The residuals of the NS equations are labelled as e_1, e_2, e_3, e_4. Note u, v and w are the fluid velocities in the x, y and z directions, respectively. The modified pressure term p^* is related to the real pressure p by $p^* = \frac{P_{dynamic}}{\rho} = \frac{P}{\rho} + g\vec{z}$, where ρ is the fluid density and g is the acceleration due to gravity.

Incompressibility $$e_1 = u_x + v_y + w_z \qquad 2.1$$

Momentum in x $$e_2 = u * u_x + v * u_y + w * u_z + p_x^* - (\nu + \nu_t)(u_{xx} + u_{yy} + u_{zz}) \qquad 2.2$$

Momentum in y $$e_3 = u * v_x + v * v_y + w * v_z + p_y^* - (\nu + \nu_t)(v_{xx} + v_{yy} + v_{zz}) \qquad 2.3$$

Momentum in z $$e_4 = u * w_x + v * w_y + w * w_z + p_z^* - (\nu + \nu_t)(w_{xx} + w_{yy} + w_{zz}) - g \quad 2.4$$

We define the loss function as the summation of the absolute value of the residuals e_i, plus the sum of the absolute difference between the CFD data and the model prediction. The parameter λ_p controls how much weight is given to the loss of the pressure term, p* while the parameters $\lambda_c$and $\lambda_m$control how much weight is given to the incompressibility and momentum transfer conditions.

$$\mathrm{Loss}_{data} = \sum_{i=u,v,w} \left| \boldsymbol{i}_{pred} - \boldsymbol{i}_{CFD} \right| \quad 2.5$$

$$\mathrm{Loss}_{pressure} = \lambda_p \left| {p^*}_{pred} - {p^*}_{CFD} \right|$$

$$\mathrm{Loss}_{continuity} = \lambda_c |\mathbf{e}_1| \quad 2.6$$

$$\mathrm{Loss}_{momentum} = \lambda_m (|\mathbf{e}_1| + |\mathbf{e}_2| + |\mathbf{e}_4|) \quad 2.7$$

$$\mathrm{Loss}_{total} = \mathrm{Loss}_{data} + \mathrm{Loss}_{pressure} + \mathrm{Loss}_{continuity} + \mathrm{Loss}_{momentum} \quad 2.8$$

A diagram of the NN model and the optimizer is shown in Figure 3.

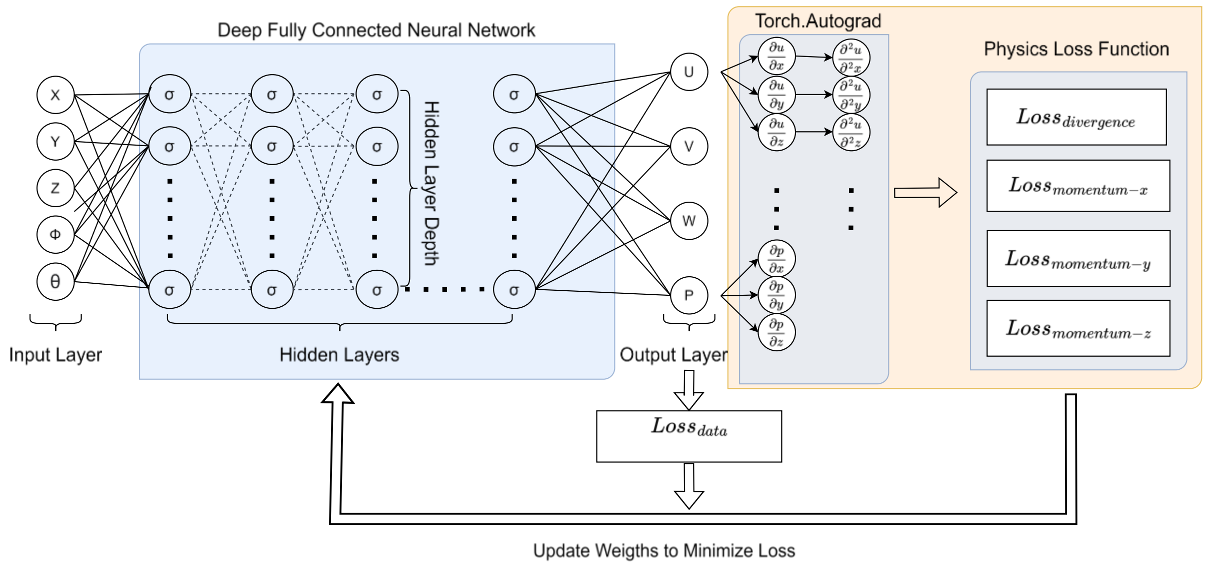
Deep Fully Connected Neural Network
X
Y
Z
Φ
θ
σ
Hidden Layer Depth
Input Layer
Hidden Layers
U
V
W
P
Output Layer
Torch.Autograd
$\frac{\partial u}{\partial x}$ $\frac{\partial u}{\partial y}$ $\frac{\partial u}{\partial z}$
$\frac{\partial^2 u}{\partial^2 x}$ $\frac{\partial^2 u}{\partial^2 y}$ $\frac{\partial^2 u}{\partial^2 z}$
$\frac{\partial p}{\partial x}$ $\frac{\partial p}{\partial y}$ $\frac{\partial p}{\partial z}$
Physics Loss Function
$Loss_{divergence}$
$Loss_{momentum-x}$
$Loss_{momentum-y}$
$Loss_{momentum-z}$
$Loss_{data}$
Update Weigths to Minimize Loss

Figure 3: Schematic of Neural Network and Optimizer used in this model.

2.3 Data Processing

When designing a NN to work with CFD data on real world scales, normalization of the data is important. In this study we will follow two approaches and create separate models for each. This first model will employ mean standardization and scaling as described by (Xu et al., 2024). However, such approaches have been shown to over prioritize low frequency functions and under prioritize high frequency patterns in the data (Wang et al., 2022). Therefore, the second model will instead employ a Fourier Feature Map (Tancik et al., 2020). Both NNs will be trained with and without Physics Informed Loss for a total of 4 models of comparison.

2.3.1 Mean Standardization and Scaling

The equation for the Neural Network(NN) trained in this paper is given in Eq 2.9.

$$NN(X, Y, Z, \varphi, \theta) = (U, V, W, p^*) \qquad 2.9$$

The NN takes as input the 3 vectors of space coordinates (X, Y, Z) as well as the wind inlet velocity direction θ and the surface normal to simulation wind direction φ. The NN returns the velocities (U, V, W) and the modified pressure term p^*. While there are many approaches to normalization including Min-Max scaling, Z-score normalization, and Decimal Scaling normalization, it has been found from early testing of this model that mean subtraction followed by parameterized scaling results in the quickest descent of the loss function. The transformation is given in Eq 2.11 and the restoring equation is given in equation Eq 2.12 with $\bar{x}$ and x' being the mean and the transformed coordinates of field x. The parameter $\hat{x}$ is the predicted value from the NN in the scale of the original coordinates. The scaling values, α_x, are given in the Results section. The purpose of this custom scaling function is to parameterize the scale on which the NN operates as we have found from early testing that a deep NN with tanh activation functions performs well on a domain greater than Z-score normalization allows.

$$x' = \frac{x - \bar{x}}{\alpha_x} \quad \forall x \in [X, Y, Z, \varphi, \theta, U, V, W, P^*] \qquad 2.10$$

$$\hat{x} = x'\alpha_x + \bar{x} \quad \forall x \in [X, Y, Z, \varphi, \theta, U, V, W, P^*] \qquad 2.11$$

In terms of scaled coordinates, the equation for the NN is given Eq. 2.12

$$NN(X', Y', Z', \varphi', \theta') = (U', V', W', P^{*\prime}) \qquad 2.12$$

Therefore, the data transformation process of the PINN and FCNN models in this paper are presented in 2.13.

$$\{X, Y, Z, \varphi, \theta\} \rightarrow^{Eq.\ 2.10} \{X', Y', Z', \varphi', \theta'\} \rightarrow^{Eq.\ 2.12} \{U', V', W', P^{*\prime}\} \rightarrow^{Eq. 2.11} \{U, V, W, P^*\} \qquad 2.13$$

It is also required to adjust the governing physical equations (residual Eqs. 2.2-2.5). Applying the general rule of calculus (Eq 2.14), we obtain the residual equations in terms of the transformed coordinates x' (Eq 2.15 – Eq 2.18).

$$\frac{\partial^n(\mathrm{ay} + \mathrm{b})}{\partial(\mathrm{cx} + \mathrm{d})^n} = \frac{a\,\partial^n y}{\mathrm{c}^n\,\partial \mathrm{x}^n} \qquad 2.14$$

Incompressibility

$$e_1 = u'_{x'}\frac{\alpha_u}{\alpha_x} + v'_{y'}\frac{\alpha_v}{\alpha_y} + w'_{z'}\frac{\alpha_w}{\alpha_z} \tag{2.15}$$

Momentum in x

$$e_2 = \hat{u} * u'_{x'}\frac{\alpha_u}{\alpha_x} + \hat{v} * u'_{y'}\frac{\alpha_u}{\alpha_y} + \hat{w} * u'_{z'}\frac{\alpha_u}{\alpha_z} + p^*_x\frac{\alpha_p}{\alpha_x} - \nu\left(u'_{x'x'}\frac{\alpha_u}{\alpha_x^2} + u'_{y'y'}\frac{\alpha_u}{\alpha_y^2} + u'_{z'z'}\frac{\alpha_u}{\alpha_z^2}\right) \tag{2.16}$$

Momentum in y

$$e_3 = \hat{u} * v'_{x'}\frac{\alpha_v}{\alpha_x} + \hat{v} * v'_{y'}\frac{\alpha_v}{\alpha_y} + \hat{w} * v'_{z'}\frac{\alpha_v}{\alpha_z} + p^{*\prime}_{y'}\frac{\alpha_p}{\alpha_y} - \nu\left(v'_{x'x'}\frac{\alpha_v}{\alpha_x^2} + v'_{y'y'}\frac{\alpha_v}{\alpha_y^2} + v'_{z'z'}\frac{\alpha_v}{\alpha_z^2}\right) \tag{2.17}$$

Momentum in z

$$e_4 = \hat{u} * w'_{x'}\frac{\alpha_w}{\alpha_x} + \hat{v} * w'_{y'}\frac{\alpha_w}{\alpha_y} + \hat{w} * w'_{z'}\frac{\alpha_w}{\alpha_z} + p^{*\prime}_{z'}\frac{\alpha_p}{\alpha_z} - \nu\left(w'_{x'x'}\frac{\alpha_w}{\alpha_x^2} + w'_{y'y'}\frac{\alpha_w}{\alpha_y^2} + w'_{z'z'}\frac{\alpha_w}{\alpha_z^2}\right) - g \tag{2.18}$$

where the model estimates $\hat{u}$, $\hat{v}$ and $\hat{w}$ are obtained from restoring the model predictions to their original scales using Eq 2.12. It should be noted that training with PIL need not be on the same coordinates as the data. Indeed, if data is scarce or not available under certain regimes it is still possible to train using physics in these regions.

2.3.2 Fourier Feature Mapping

In the second model type, we insert a Fourier Feature mapping layer as the first layer of our FCNN and PINN models, hereafter referred to as FF-FCNN and FF-PINN. This choice is motivated by the theoretical and practical benefits of such a mapping in enhancing the network's ability to learn the complex NS equations, particularly regions with high-frequency components. Note, the use of the term frequency in this paper refers to the rate of change of the NN output with respect to the input coordinates. The Fourier feature mapping $\gamma(x)$ transform the input data **X** into 'Fourier space' using sinusoidal functions. In our case, we map the X, Y and Z coordinates as well as the input angles φ, θ to the frequency domain by multiplying each data point by a set of 30 random normally distributed frequencies with variances $\sigma^2{}_i$. Importantly, scaling the $\sigma^2{}_i$ parameters are crucial for controlling the convergence of the NN. If the $\sigma^2{}_i$ are too large we will not capture small details in the data. If $\sigma^2{}_i$ are too small, we risk underfitting the data by not giving the model the finer details between coordinates. We take the sine and cosine of each coordinate times frequency and pass this layer to the NN as described in Eq. 2.19 - 2.22.

$$\mathbf{X} = [\mathrm{X}, \mathrm{Y}, \mathrm{Z}, \varphi, \theta\ ] \qquad 2.19$$

$$\mathbf{B}_i \in \mathbb{R}^{1\times30}, \qquad \mathrm{B}_i \sim \mathrm{N}(0, \sigma^2{}_i) \qquad 2.20$$

$$\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4, \mathbf{B}_5]^T \qquad 2.21$$

$$\boldsymbol{\gamma}(\mathbf{X}) = [\sin(\mathbf{BX}), \cos(\mathbf{BX})] \qquad 2.22$$

The Fourier feature mapping becomes the first layer of the NN. The equation of the FF-PINN and FF-FFCN is given in Eq 2.23.

$$NN(\mathrm{X}, \mathrm{Y}, \mathrm{Z}, \varphi, \theta) = (U, V, W, p^*) \qquad 2.23$$

Unlike the FCNN and PINN, we chose to incorporate the Fourier Feature layer and the data scaling layer within the bounds of the FF-FCNN and FF-PINN models. This last layer simply scales the output data (U, V, W, P*) as per Eq. 2.11. This allows the use of Automatic Differentiation (AD) with respect to the untransformed input vectors (X, Y & Z).Therefore the we use the untransformed residual Eqns 2.1-2.4 for the physics informed loss function of this model.

2.4 Parameter Tunning

The PINN and FF-PINN models contain a considerable number of parameters. Tuning the λ_c, λ_m and λ_p parameters is performed by varying each parameter over a range of values and observing the values which resulted in the lowest validation loss achieved over 2,000 epochs of training. This is performed to find an optimal fixed value for each parameter.

Further optimization can be achieved using curriculum regularization as described by Krishnapriyan et al. (2021). Curriculum regularization is achieved by gradually increasing the PIL parameters as the model trains until the PINN converges correctly.

3 Results

3.1 Model Parameters

The optimal values of λc, λm, λp are given in Table 1 (see Figure 11 - Figure 13 in Appendix 7.2 for their optimization curves). These values are application specific, and it is not reasonable to assume they would optimize PIL in other contexts. It also was found that the total viscosity, ν in the PINN did not help reduce the loss in the validation set, thus this parameter was set to 0 (see Figure 14 and the accompanying text in Appendix 7.2).

The scaling factors α_i, presented in Table 3: FCNN & PINN Scale Factors resulted in the quickest descent of the loss function. It is noteworthy that the three spatial dimensions, (x, y, z), all had a range of approximately greater than -6 and less than +6 after optimal scaling, while the velocity components all had ranges strictly between +1 and -1.

Mass Conservation loss Weight(Continuity)	λ_c	35
Momentum Loss Weight	λ_m	.3
Pressure Loss Weight	λ_p	0.32

Table 1: Physics Informed Weights

Loss Function	MSE
Optimizer	LBFGS
LBFGS Learning Rate	1.0
LBFGS History Size	100
LBFGS Line Search Fn	"strong_wolfe"
Neural Net Size	60
Neural Net Depth	10

Table 2: Model design parameters

x Scale Factor	α_u	32
y Scale Factor	α_y	16
z Scale Factor	α_z	1
θ Scale Factor	α_θ	1
φ Scale Factor	α_φ	1
u Scale Factor	α_u	32
v Scale Factor	α_v	50
w Scale Factor	α_w	30
P^* Scale Factor	α_{P^*}	200

Table 3: FCNN & PINN Scale Factors

x Frequency Variance	$\sigma^2{}_x$	0.0016
y Frequency Variance	$\sigma^2{}_y$	0.0016
z Frequency Variance	$\sigma^2{}_z$	0.0048
θ Frequency Variance [deg^{-1}]	$\sigma^2{}_\theta$	0.0027
φ Frequency Variance[rad^{-1}]	$\sigma^2{}_\varphi$	0.026
u Scale Factor	α_u	32
v Scale Factor	α_v	50
w Scale Factor	α_w	30
P^* Scale Factor	α_{P^*}	200

Table 4: FF-FCNN & FF-PINN parameters

To test the efficacy of Curriculum Regularization, Figure 4 compares two models, one with constant Physics Loss weight and another with a gradually increasing weight. The PINN with Curriculum Regularization sees a large spike at epoch 1000 due to the sudden increase in λ_c from 0 to 10. It is clear the PINN with the gradually increasing weight learns faster and achieves a lower overall loss over the first 5k epochs. Therefore, this strategy is employed for all models hereafter.

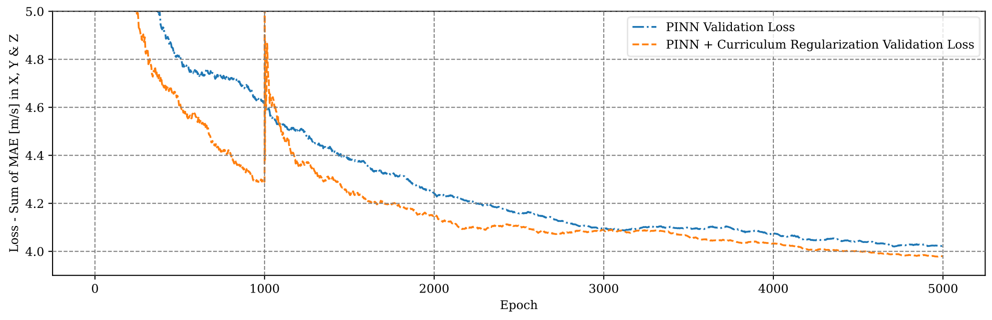

Figure 4: Validation Loss of PINNs with and without Curriculum Regularization. The angles between -40° and +40° were used for training and all available angles (-70° to +50°) were used for validation while the models were training. Both Models were stopped at Epoch 5000. Curriculum Regularization was performed by increasing λ_c by 10 every 1,000 Epochs starting from 0 and ending with $\lambda_c = 50$ and the PINN without Curriculum Regularization had a constant $\lambda_c = 50$.

Figure 5 provides the relationship between the grid resolution of the model and the model performance on extrapolated angles. The purpose of this graph is to illustrate the relationship between data availability and model performance in a fixed number of epochs. The data is sampled randomly, and the proportion of data sampled is equivalent to having a grid made from square cells with a side length equal to the grid resolution. The lowest grid resolution (5m) contains 36x as many points as the highest grid resolution (30m). Over this range the performance of the PINN and FCNN appears to drop by ~10%. Each simulation was terminated after 5k epochs; this was enough time for the FCNN validation loss to plateau, but the PINN was still training. Thus, the PINN might achieve better performance if more epochs were available for training.

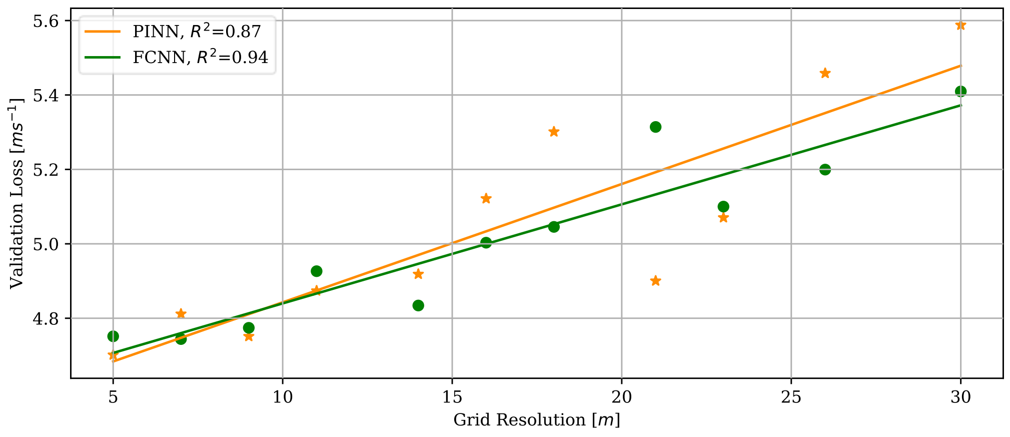

Figure 5 Comparison of Training Data resolution to Validation Loss (MAE) of training for PINN and FCNN models. The wind inlet angles, -40°, -30°, -20°, -10°, 0°, 10°, 30°, 40° were used for training and every angle (-70°... +50°) was used for validation. The lowest validation loss achieved after 5k Epochs is reported. For a given Grid Resolution, GR, the proportion of sampled data points was 1/GR^2.

3.2 Model Evaluation

A grid resolution of 10m corresponding to 1% of available training data was selected for the final FCNN, PINN, FF-FCNN and FF-PINN. The choice of this resolution is founded in practical constraints; Figure 5, shows a minor decrease of approximately 4% in accuracy from 5m resolution to a 10m resolution for both training set despite the former being four times larger, and almost four times slower to train. For the final comparison all models were trained until the validation loss could not be reduced any further. The angles -40°, -30°, -20° , -10°, 0°, 10°, 30°, 40° were used for training. The angles -70°, -60°, -50°, +20°, +50° were reserved for testing. The choice of the test/train split was to test the model's ability to interpolate between trained angles and extrapolate in both training directions with deep extrapolation of the negative test angles.

In addition to the 4 models described in this paper, we present the performance of a convolutional neural network (CNN) model designed by Thijs Modderman for a similar research question. The model employs feature pyramid network(FPN) architecture utilizing the efficientnetb7 backbone and was implemented for pixelwise regression. The model was trained using two input channels that reflect the dune topography and the same cosine wind inlet difference θ used in our models.

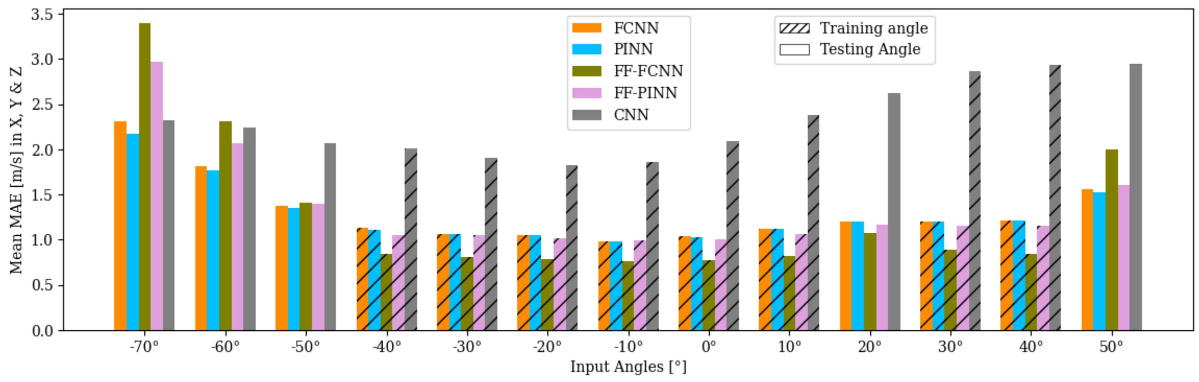

*Figure 6: Final Comparison of performance. Reported are the validation accuracy of each model across training and testing angles. Every model was allowed to train until convergence. The Physics Informed models employed Curriculum Regularization upto the parameter weights defined in Table 1. All other parameters are taken as constants from in Table 1-**Error! Reference source not found.** Table 4.*

Training the PINN and FF-PINN models for six thousand epochs took ~23 minutes on Google Colab's T4 TPU using 8GB of TPU RAM(Google Collaboratory, 2024). The FCNN and FP-FCNN models took 5 minutes on the same device using 0.5GB of TPU RAM. The sum of Mean Absolute Errors in X, Y & Z each for each model was divided by three thus the reported values are an average MAE for the U, V & W components.

Figure 6 illustrates the performance of each model across angles that were used for testing and training. The most performant model in the training angles is the Fourier Feature layer fully connected neural network (FF-FCNN). The model has a large number of parameters in the first training layer (150), and this could be a possible source of over fitting as this model is the worst in the deep extrapolation (-70°) test case. Comparatively the same model with PIL (FF-PINN) does not learn the training angles as well but outperforms the FF-FCNN in the extrapolated angles suggesting PIL maybe eliminating some degree of overfitting.

Figure 6 also illustrates the similarity between the FCNN and PINN models in the training angles suggesting that the Physics Informed Loss does not contribute a lot when data is plentiful. We do observe the PINN with the lowest MAE of all models in the -70° case and the +50° degree case suggesting that PIL helps a model to extrapolate beyond the training regime. A full comparison of the PINN's performance against the -70° CFD simulated data is provided in Figure 7 and Figure 8. For comparison the same result of the FF-FCNN is available in Appendix 7.3 and the PINN is also tested on the training angle of 0° in Figure 9 and Figure 10.

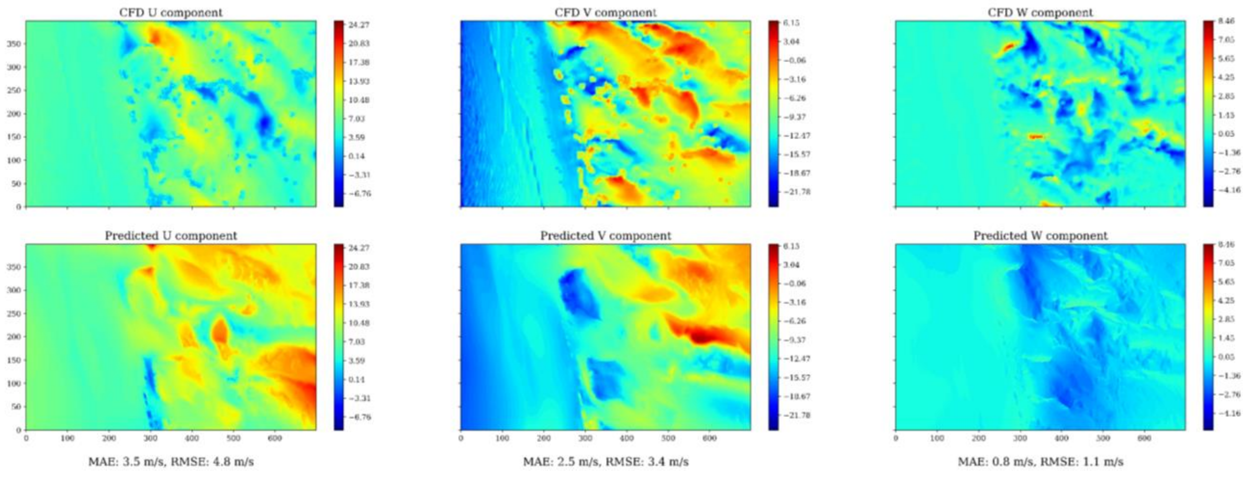

Figure 7: PINN model extrapolation test case for -70° angle. The model was trained with 1% of the data from the angles -40°, -30°, -20°, -10°, 0°, 10°, 30°, 40°. An additional 5% of the available coordinates from all angles were supplied to the model for physics only training (without data).

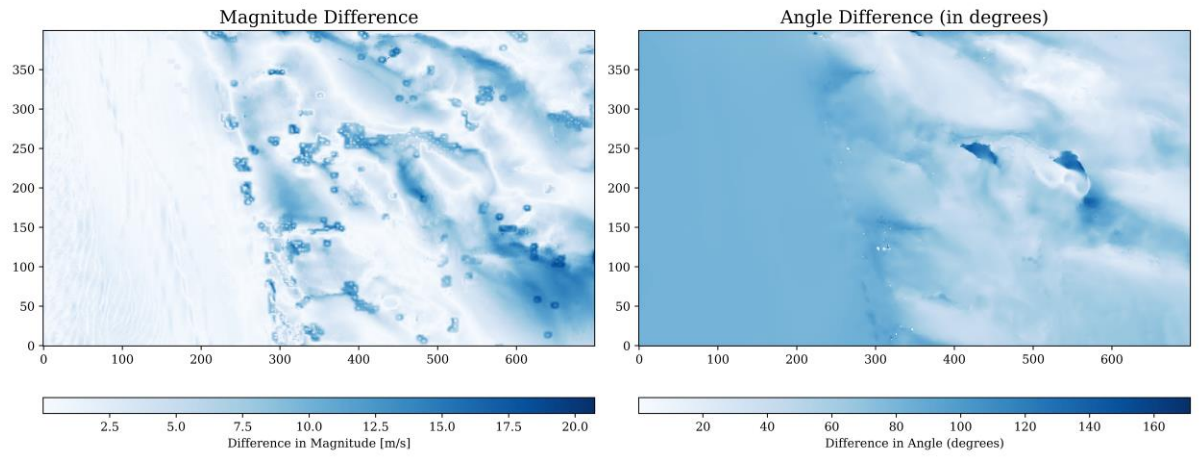

Figure 8: Wind Speed Difference and Wind Angle Difference test case for -70° angle for the same PINN model as Figure 7. Wind Angle Difference is the angle between the predicted wind velocity by the CFD and the predicted velocity by the PINN.

The small circular features in the Magnitude Difference graphs of Figure 8 and Figure 10 show us that the PINN trained in this paper does not capture the vortices in the training data and does not predict them in the extrapolation test case. Furthermore, the prediction in wind angle direction in Figure 8 shows the failure of the PINN model to extrapolate to this direction.

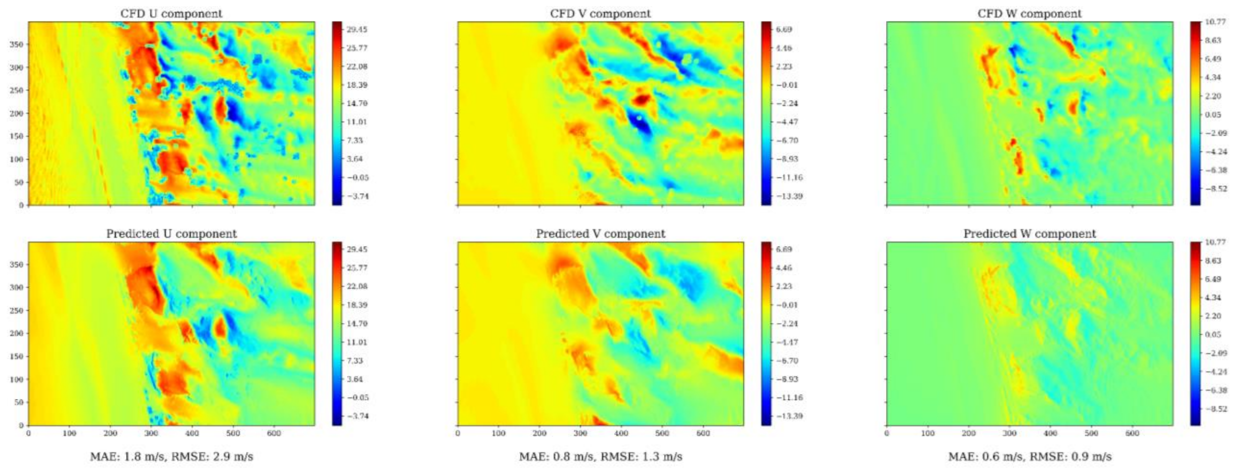

Figure 9: Training Angle 0° Prediction for the same PINN model as in Figure 7.

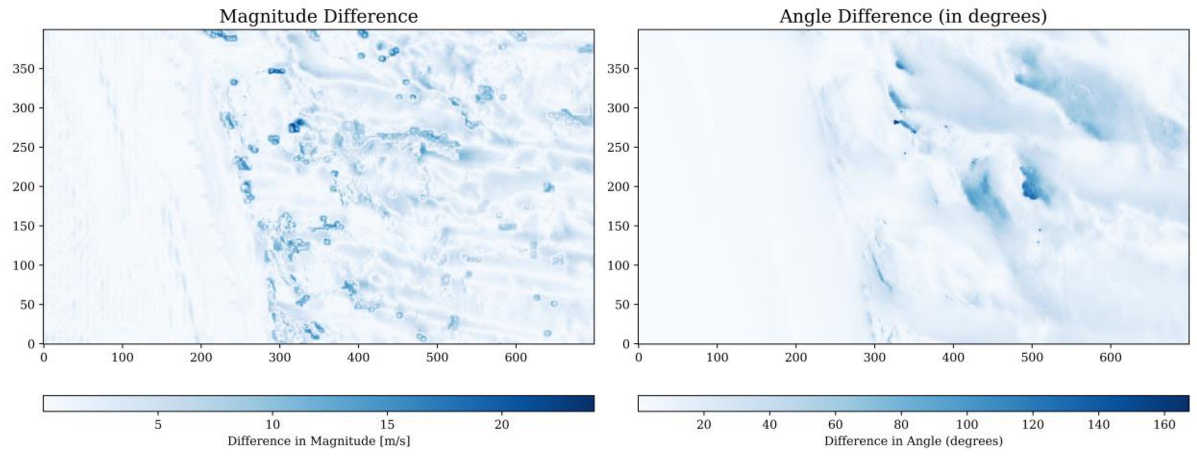

Figure 10: Wind Speed Difference and Wind Angle Difference test case for -70° angle for the same PINN model as Figure 7.

Figure 9 and Figure 10 demonstrate the ability of the PINN to learn training angles and reproduce results that are plausible even when trained on just 1% of the available data.

4 Conclusions and Discussion

In this paper, we designed a neural network to predict wind velocity across a beach and sandy terrain, employing Physics Informed Learning (PIL) to enhance model performance. Our results indicate that PIL marginally improves the neural network's ability to extrapolate beyond the training regime, as evidenced by comparing the PINN to FCNN and comparing the models incorporating a Fourier Feature layer. We failed to see an improvement of the PINN over the FCNN in the interpolation case. Notably, the mean velocity error across all spatial dimensions was generally no more than 5 m/s in the 70° test case.

Neural networks have demonstrated the capability to accurately predict wind velocity across sand dunes using relatively sparse, random data. The prediction at a 1m resolution on a 700m x 400m grid takes approximately 0.002 seconds once the model is trained, comparable for both physics-informed and non-physics-informed neural networks. Thus, we have illustrated a positive trade-off between computational cost and accuracy for PINNs and FCNNs compared to classical CFD simulations.

However, if CFD results are unavailable or NN training time is a limiting factor, the benefits of using PIL are less clear. Another conclusion of this research is that Physics Informed Learning can enhance the extrapolation accuracy of surrogate CFD models for real-world problems, though the improvement is marginal and requires significant tuning. This presents a barrier to widespread deployment. Additionally, while our extrapolation test case was performed on the same terrain as the training case, real-world applications necessitate a model capable of performing well on new, unseen terrains.

The Convolutional Neural Network (CNN) architecture showed worse performance in reproducing velocity fields from trained angles but demonstrated comparable performance in the deep extrapolation test case (-70°). The lack of degradation in performance for the extrapolated test case is promising, suggesting potential research avenues were combining PIL with convolutional architectures could enhance surrogate modelling.

Some errors in the final models are expected due to the dynamic nature of Computational Fluid Dynamics (CFD) simulations and the fact that the training data consisted of only a single snapshot in time. Therefore, predicting the velocity of turbulent flow will inherently produce some inaccuracies.

Interestingly, while the Fourier Feature layer models were the best learners in the training data, they were the worst performers on the deep extrapolation test case. It is promising that FF layer models are expressive enough to reproduce the training data and it is probable that in the current network design overfitting is likely occurring. Physics Informed Fourier Neural Operator (PINO) models, which begin with a Fourier Feature layer, have shown strong adaptability in multi-scale dynamic systems and it would be interesting to test their ability on the simulation of wind over sand dunes.

5 Acknowledgements

I would like to thank Dr. Saeb Faraji Gargari for his guidance and encouragement in pursuing this project on Physics Informed Learning and for producing the CFD data used as part of the training of these models. I would also like to thank Oriol Pomarol Moya for his valuable feedback and suggestions in Machine Learning and scientific writing. Finally, I would like to thank Prof. Derek Karssenberg and the Department of Geosciences at Utrecht University for enabling and supporting this research project.

6 References

Arzani, A., Wang, J.-X., & D'Souza, R. M. (2021). Uncovering near-wall blood flow from sparse data with physics-informed neural networks. *Physics of Fluids*, *33*(7). https://doi.org/10.1063/5.0055600

Edward, B. B., Sally, D. H., Chris, K., Evamaria, W. K., Adrian, C. S., & Brian, R. S. (2011). The value of estuarine and coastal ecosystem services. *Ecological Society of America*.

Everard, M., Jones, L., & Watts, B. (2010). Have we neglected the societal importance of sand dunes? An ecosystem services perspective. *Aquatic Conservation: Marine and Freshwater Ecosystems*, *20*, 476-487. https://doi.org/10.1002/aqc.1114

Gao, H., Sun, L., & Wang, J.-X. (2021). Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. *Physics of Fluids*, *33*(7). https://doi.org/10.1063/5.0054312

Google Colaboratory. (2024). Google LLC. https://colab.research.google.com/

Guo, X., Iorio, F., & Lei, W. (2016). Convolutional Neural Networks for Steady Flow Approximation. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939738

Hecht-Nielsen, R. (1992). III.3 - Theory of the Backpropagation Neural Network**Based on “nonindent” by Robert Hecht-Nielsen, which appeared in Proceedings of the International Joint Conference on Neural Networks 1, 593–611, June 1989. © 1989 IEEE. In H. Wechsler (Ed.), *Neural Networks for Perception* (pp. 65-93). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-741252-8.50010-8

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., & Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, *34*, 26548-26560.

Luijendijk, A., Hagenaars, G., Ranasinghe, R., Baart, F., Donchyts, G., Aarninkhof, S., Luijendijk, A., Hagenaars, G., Ranasinghe, R., Baart, F., Donchyts, G., & Aarninkhof, S. (2018). The State of the World’s Beaches. *Scientific Reports 2018 8:1*, *8*(1). https://doi.org/10.1038/s41598-018-24630-6

Merriënboer, B., Breuleux, O., Bergeron, A., & Lamblin, P. (2018). *Automatic differentiation in ML: Where we are and where we should be going*.

OpenFOAM. (2024). In (Vol. The OpenFOAM Foundation). https://openfoam.org/: OpenFoam.org.

Raissi, M., Paris, P., & Em, K. G. (2017). Machine learning of linear differential equations using Gaussian processes. *Journal of Computational Physics*, *348*. https://doi.org/10.1016/j.jcp.2017.07.050

Raissi, M., Perdikaris, & Karniadakis. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*. https://doi.org/10.1016/j.jcp.2018.10.045

Raissi, M., Wang, Z., Triantafyllou, M. S., & Karniadakis, G. E. (2019). Deep learning of vortex-induced vibrations | Journal of Fluid Mechanics | Cambridge Core. *Journal of Fluid Mechanics*, *861*. https://doi.org/10.1017/jfm.2018.872

Ruessink, G. (2021). *Why do these Dutch dunes have gaps? The play between wind, sand and biodiversity in Bloemendaal*. Faculty of Geosciences Utrecht University. https://youtu.be/6MIVty2ENos?si=GIQdLdWogHsmUJch

Shao, X., Liu, Z., Zhang, S., Zhao, Z., & Hu, C. (2023). PIGNN-CFD: A physics-informed graph neural network for rapid predicting urban wind field defined on unstructured mesh. *Building and Environment*, *232*, 110056. https://doi.org/https://doi.org/10.1016/j.buildenv.2023.110056

Stiasny, J., Chevalier, S., & Chatzivasileiadis, S. (2021). *Learning without Data: Physics-Informed Neural Networks for Fast Time-Domain Simulation*. https://doi.org/10.1109/SmartGridComm51999.2021.9631995

Sun, L., Gao, H., Pan, S., & Wang, J.-X. (2020). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, *361*, 112732. https://doi.org/https://doi.org/10.1016/j.cma.2019.112732

Suo, W., & Zhang, W. (2023). A novel paradigm for solving PDEs: multi scale neural computing. *arXiv preprint arXiv:2312.06949*.

Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., & Ng, R. (2020). Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *ArXiv*, *abs/2006.10739*.

Van Koningsveld, M., Otten, C., & Mulder, J. (2007). Dunes: the Netherlands soft but secure sea defences. Proceedings 18th World dredging congress, Florida USA, WODA,

Wang, S., Yu, X., & Perdikaris, P. (2022). When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, *449*, 110768. https://doi.org/https://doi.org/10.1016/j.jcp.2021.110768

Weller, H. G., Tabor, G., Jasak, H., & Fureby, C. (1998). A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computer in Physics*, *12*(6), 620-631. https://doi.org/10.1063/1.168744

Xu, S., Chang, Y., Sun, Z., Huang, R., Dilong, G., & Yang, G. (2024). *On the Preprocessing of Physics-informed Neural Networks: How to Better Utilize Data in Fluid Mechanics*.

Zhang, Y., Zhang, D., Jiang, H., Zhang, Y., Zhang, D., & Jiang, H. (2023). Review of Challenges and Opportunities in Turbulence Modeling: A Comparative Analysis of Data-Driven Machine Learning Approaches. *Journal of Marine Science and Engineering 2023, Vol. 11, Page 1440*, *11*(7). https://doi.org/10.3390/jmse11071440

7 Appendix

7.1 Optimization Curves

In this Appendix Chapter we determine the optimal values of the λ loss weights by repeating simulations for a finite number of epochs over a range of the scaling values. A quadratic curve is fitted to each graph as an optimal value of λ should result is a curve with a local minimum.

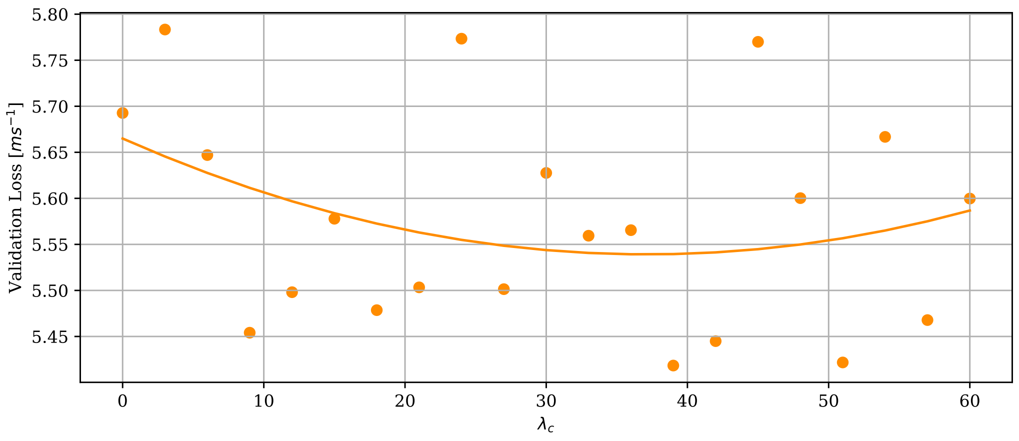

Figure 11: Epoch 1500 validation loss of 20° Wind Inlet Angle for PINN trained on -40°, -30°, -20° , -10°, 0°, 10°, 30°, 40° angles. Data points were sampled from a 10mx10m grid. The parameters λ_m =0.3 and λ_p= 0.3 were held constant.

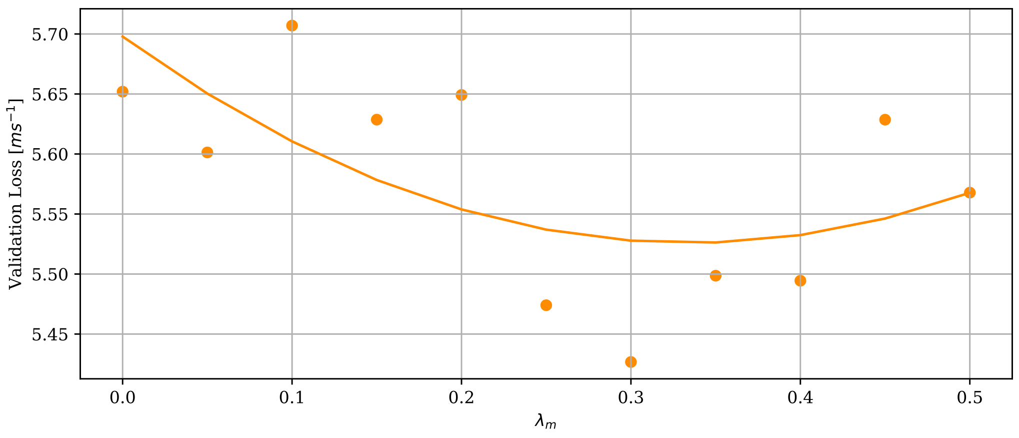

Figure 12: Epoch 1500 validation loss of 20° Wind Inlet Angle for PINN trained on -40°, -30°, -20° , -10°, 0°, 10°, 30°, 40° angles. Data points were sampled from a 10mx10m grid. The parameters λ_c =35 and λ_p=0.3 were held constant.

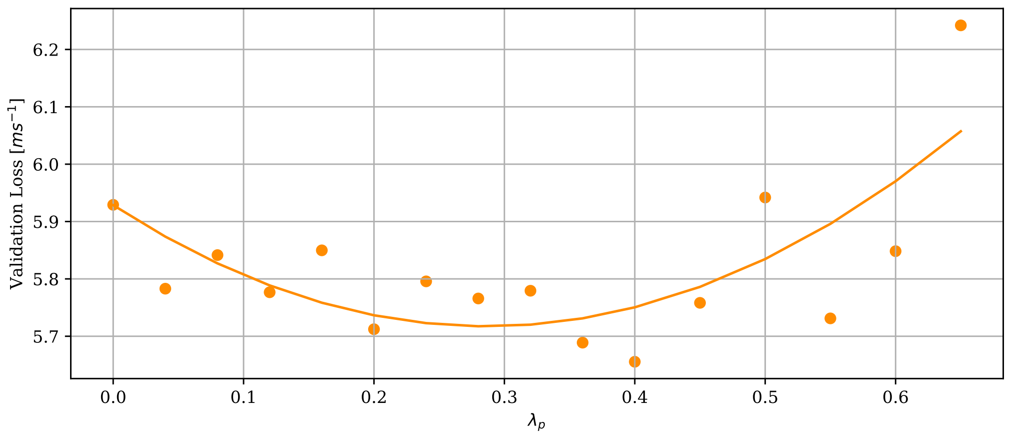

Figure 13: Epoch 1500 validation loss of 20° Wind Inlet Angle for PINN trained on -40°, -30°, -20° , -10°, 0°, 10°, 30°, 40° angles. Data points were sampled from a 10mx10m grid. The parameters λ_m =0.3 and λ_c=30 was held constant.

7.2 Investigation into fluid viscosity

Fluid Viscosity is one potential source of error in this model which warrants investigation. The CFD simulations model the turbulent viscosity, ν_t as a scaler field value. Our model does not predict the viscosity nu throughout the field therefore we must pick an average value of the total viscosity which is the sum of the turbulent viscosity and the kinematic viscosity. The average value of ν at the end of the simulation is 0.11 $kg \cdot m^{-1} \cdot s^{-1}$. Varying the total viscosity ν from 0 to 0.11 yields no clear indication of improved performance, thus we select $\nu = 0$ for all further models.

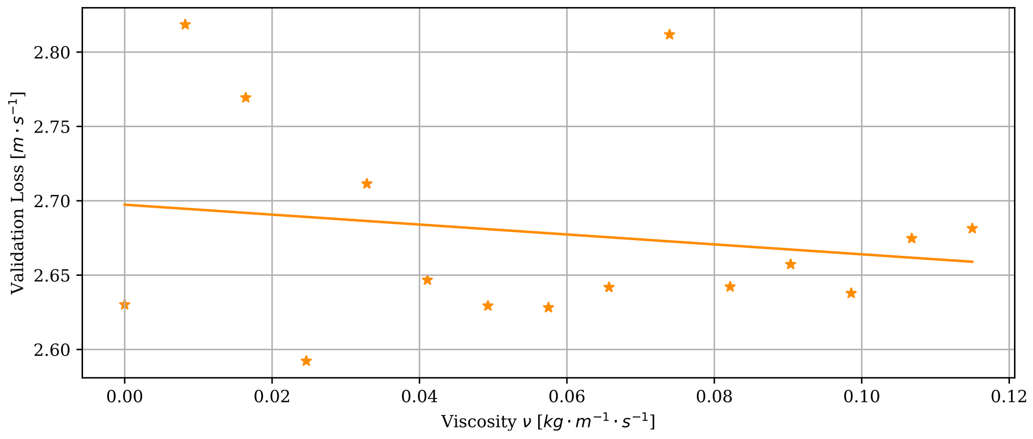

Figure 14: Investigation into varying the total viscosity of the fluid. Note: An error in the code was discovered after production of this figure resulting in the validation loss in the y-component not being added to the mean validation loss. This error caused the total validation loss to be artificallly lower but should not affect the conclusion.

7.3 Fourier Feature layer PINN Model Training Performance

The PINN model with a Fourier Feature layer is the best learner of the training data while the Fourier Feature layer models perform poorly on extrapolation. It is therefore interesting to observe the learning set performance. This model was exposed to just 1% of the training data yield's remarkable ability to reproduce the entire data set.

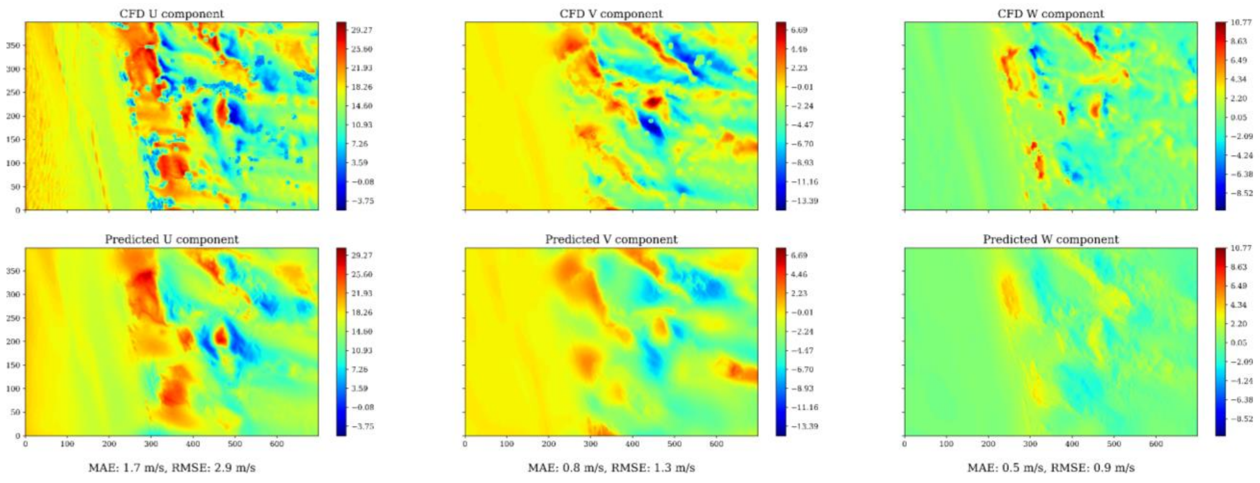

Figure 15: Final FF-PINN Model Prediction of 0°(trained) wind Inlet Angle