Delft University of Technology

CFD Acceleration for Perforated Monopile Design

Machine-learning based methods to accelerate design optimization of perforated monopiles

Ruben Dekeyser

For the TU Delft MSc in Offshore & Dredging Engineering (SDA)

TU Delft

CFD Acceleration for Perforated Monopile Design

Machine-learning based methods to accelerate design optimization of perforated monopiles

by

Ruben Dekeyser

to obtain the Master of Science degree in
Offshore & Dredging Engineering
(Track: Structural Design and Analysis)
at Delft University of Technology,

Author:	Ruben Dekeyser (student ID: 4804252)	
Thesis committee:	Dr. Oriol Colomés Gené	TU Delft, Chairman & supervisor
	Dr. Alexander Heinlein	TU Delft, supervisor
	Prof. Dr. Andrei Metrikine	TU Delft, quality control
	Dr. Eliz-Mari Lourens	TU Delft, cum laude supervisor
	Ir. Jan Modderman	TU Delft, supervisor
	Ir. Marco Vergassola	TU Delft, supervisor
Project Duration	November 2022 – June 2023	

Cover picture: Oil & Gas Today, 2020

TU Delft

Preface

This thesis was written in order to obtain the Master of Science degree in Offshore & Dredging Engineering (Track: Structural Design and Analysis) at Delft University of Technology. The goal is to get a solid understanding of the possibilities that Julia (Gridap.jl) based finite element methods for CFD provides in order to understand the mechanics of turbulent fluid flow around perforated offshore wind turbine monopiles, and to which extent surrogate, machine learning based, models can help in optimizing calculation costs. The report is written for readers who are looking for a mainly technical analysis of the problem, and the associated computational challenges and opportunities.
The thesis was written under the supervision of Dr. O. Colomés, as well as Dr. A. Heinlein, Ir. J. Modderman and Ir. M. Vergassola.

Ruben Dekeyser
Delft, 2023

Abstract

To meet global green energy targets, the bottom founded offshore wind industry is looking for ways to economically expand markets to deeper waters. A reduction of the hydrodynamic load is necessary to achieve this. One option is to perforate the monopile around the splash zone. Here the related work of Q. Star (2022) is continued through the implementation of a 2D Navier-Stokes based LES CFD model with VMS closure in Gridap.jl. The numerical simulations are gathered to create a dataset on which a machine learning surrogate model is trained. The analysis does not include secondary design aspects such as manufacturing, installation, noise mitigation, scour, corrosion, acidification, or marine growth. Neither does it require secondary fluid phenomena such as 3D simulations, FSI, VIV, and fatigue.

Analysis of 2D perforated cylinders, built up parametrically using the wall thickness, angle of attack, diameter, inflow velocity, number of perforations and porosity as input parameters shows that the first two are of negligible influence, and number three and four can, at least for their mean values, be modelled accurately using Morison equations. A non-scaled diameter does prove important in reducing the random nature of frequency-based effects. The number of perforations and the porosity provide complex interactions both from a design-force mean and variability perspective, as well as for the frequencies and vibrations they generate. A 2D LES-VMS model gives the perfect trade-off between cost and accuracy, predicting a possible drag reduction of more than 50% compared to a closed cylinder with large diameter.

Different machine learning surrogate models are analysed with the goal of massively speeding up the analysis in the future. This is achieved by a factor of 350 thousand using Random Forests, Gaussian Processes and Neural Networks. Although the Gaussian Processes preliminary show the best accuracy, below 6% error for millisecond predictions, further work on Neural Networks could give them the upper hand in future analyses.

Table of Contents

List of Figures

List of Tables

1 Introduction

In a changing geopolitical climate the need for self-sufficient, green energy generation has become a major point of focus. Offshore wind energy is one of the key drivers in this process. As of December 2021 the worldwide offshore wind capacity is around 56 GW (Statista, 2022), with 28 GW installed in European waters (Wind Europe, 2021). 81% of these wind turbines are installed on monopile foundations. Additionally the EU has pledged to increase its capacity to over 300 GW by 2050 (European Commission, 2022; Wind Europe, 2021). This ambition includes foreseen problems with economic exclusion zones, in the Netherlands limiting the allowable development area to just 4% of its national waters (Samira Keivanpour, 2017), and including a 30 km-to-shore logistics limit. Also internationally the offshore wind energy market is getting more and more traction with the Biden administration for example having pledged 30 GW of installed capacity by 2030 (US Department of Energy, 2021). Developments in the far-east are bound to follow soon as well.

One of the main challenges of new developments are the issues associated with moving further offshore and into deeper waters. The worldwide potential is estimated to be 10 times the current worldwide energy demand when disregarding depth related issues (Wind Europe, 2021). This assumes the availability of floating wind options. Other sources go even further stating that the energy demand can be met with depth limits of 60 m within 60 km from the nearest shoreline (IEA, 2019). Technical limitations to go fully green then follow from production facilities and financial aspects. Current short-term production of monopiles has box limits of around 2500 tons of weight, 120 m of single-tube length, 15 m of diameter and 250 mm wall thickness (Steelwind Nordenham, 2022; Davis, 2019). Lifting vessels have similar boundaries. Financial limitations are driven by the increasing cost for deeper and larger foundations, where floating options are expected to become interesting for large scale investments from a LCoE of 60$/MWh or lower, which is unlikely to happen in the near future (ORE Catapult, 2021; Merrifield, 2018). Especially for deeper waters the foundation structure becomes a major cost driver contributing, depending on the source, between 8 and 15% to the LCoE (Jeanne Yacoubou, 2022; Wang et al., 2018; Lozano-Minguez et al., 2011), with scaling often following an exponential relation of factor 2.3. Andersen et al. (2020) additionally state that for traditional monopile designs the costs become uneconomical at 50 m of water depth, with depths of up to 60 m then still being eligible for jacket and tripod structures. However their manufacturing process is labour heavy, increasing their cost significantly.

It becomes clear that in order to economically increase the offshore wind capacity the cost of bottom founded turbines for deeper waters should be reduced. The generally cheapest and most used option, namely monopiles, suffers from large hydrodynamic loads in deeper waters, eventually requiring so much steel weight that they become unfeasible, both technically and financially. In order to reduce this hydrodynamic loading Andersen et al. came up with a perforation concept, as shown in Figure 1.1. They proved that for high Keulegan-Carpenter-number designs over 20% reduction of the hydrodynamic loads could be obtained. Following this idea the TU Delft Offshore Engineering Group continued with conceptual work on the design by van der Ploeg (2021) and modelling options by Star (2022). The latter work forms the basis on which the expansion of the upcoming thesis is built. The goal of the perforations is to reduce the hydrodynamic loading more than what one loses in structural rigidity from removing material and introducing stress concentrations around the perforations. The end question is: “Can we move to deeper waters with less material?”, since this is the main cost driver and indicates the general design limitations.

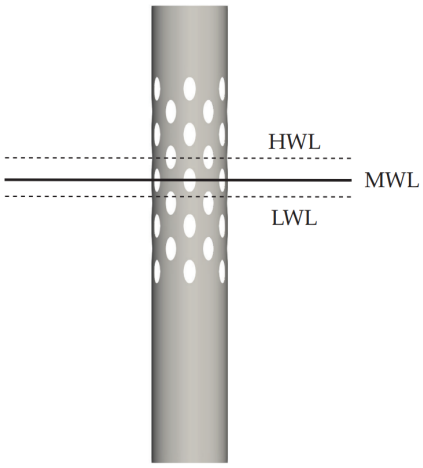


Figure 1.1: Perforated monopile concept (Andersen et al., 2020)

In this thesis the first goal is to get a better understanding of the fluid mechanics surrounding these perforated monopiles. To prevent the computationally expensive modelling that comes with the CFD (Computaitonal Fluid Dynamics) simulations having to be redone in the future, the second goal is to provide a surrogate model that can still grasp the most important aspects of the fluid simulations, at a fraction of the CPU-cost.
The research method for this thesis will be purely theoretical and computational, disregarding any form of physical experiments. The concrete results should answer the research questions, with a road map on how to apply the results in real world practice. The main question to be answered in this thesis is:

How can we apply surrogate models to reduce the CFD calculation cost of deep-water perforated monopiles in offshore wind turbine design?

This question can be solved through the use of three subquestions:

1. Which design aspects and hydrodynamic effects are important in the determination of hydrodynamic loads on perforated monopiles?

2. How can we apply Gridap.jl to accurately capture the fluid flows and pressure-force relations surrounding perforated monopiles?

3. How can we apply machine learning as a surrogate model for improving the calculation cost over Gridap.jl based CFD calculations?

From the initial introduction here in Chapter 1 the outline of the required topics to be studied already comes forward. Firstly the hydrodynamic optimization should be correctly placed within the full wind turbine design process and all of its related aspects. This is done in Chapter 2. The same chapter also gives a deeper understanding of the hydrodynamics, and mainly turbulence effects which are at play. After that the second subquestion is answered in Chapter 3. A deeper dive is made into the application of Gridap.jl for the design issues at hand, together with an overview of results which can be directly applied in engineering practice. To eventually speed up the lengthy simulations different surrogate models could be used. The options and their practical application are looked into in Chapter 4. A recommendation is made on which model to apply for further work, and how to apply it. The work is concluded in Chapter 5. Additionally a short discussion will be added on missing literature which falls out of the scope of this thesis report, together with recommendations for future research.
All these concepts essentially come back in Figure 1.2, where the conceptual links, general procedural outlines and implementation questions are highlighted.

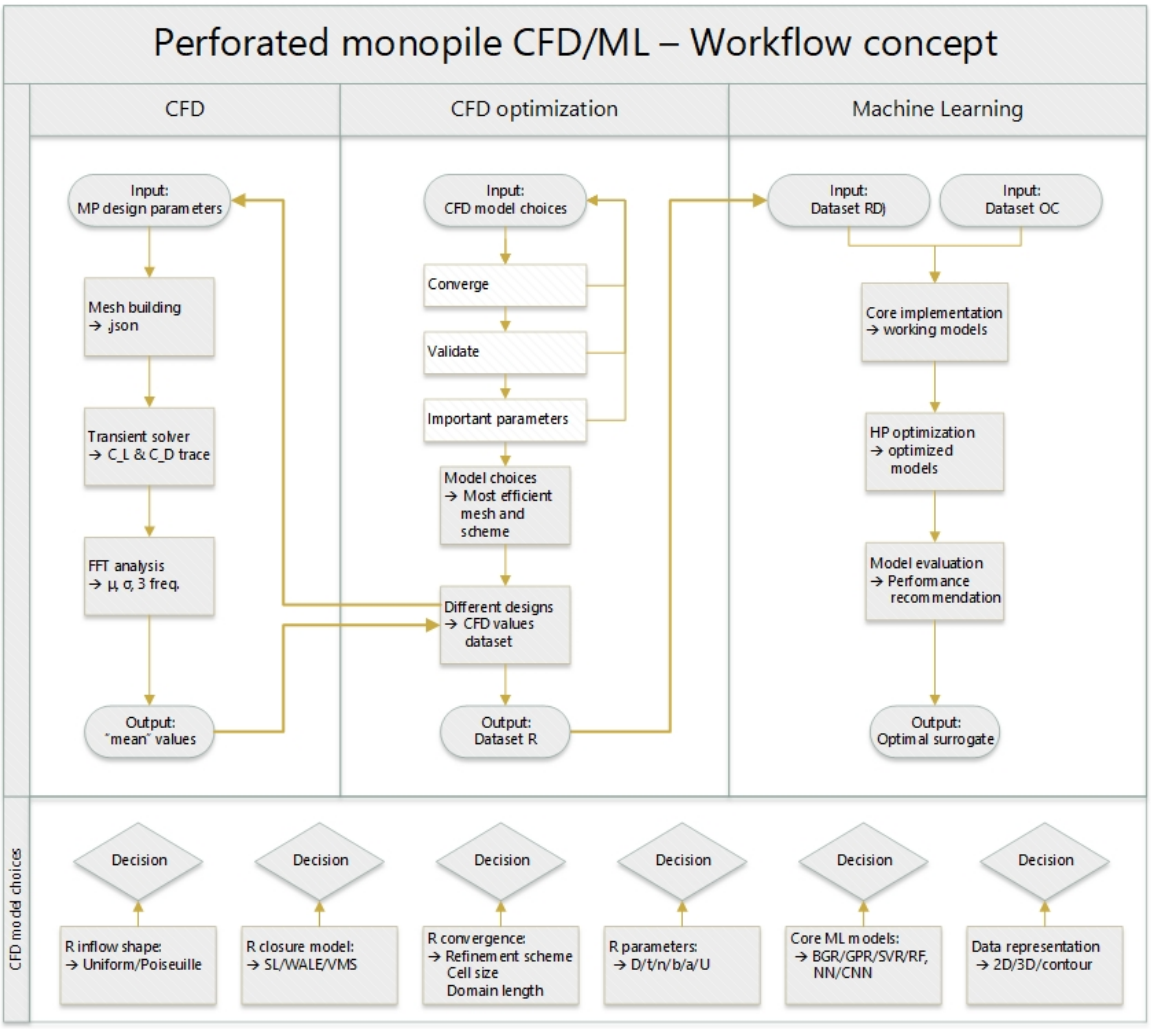


Figure 1.2: Calculation and workflow outline

2

Calculation, Design and Fluid schemes

In this chapter the general outline of the offshore wind turbine design process is given (Section 2.1). The focus lies on the place of hydrodynamic load optimization and the factors contributing to that. Additionally some different options for fluid modelling are discussed in Section 2.2. Both sections show which elements are, at this stage, relevant and which are not. This includes a deeper mathematical dive into the application of the here best-chosen model.

2.1. Offshore wind support structure (OWSS) design

Offshore wind turbines are built up of three main parts: The rotor, the nacelle, and the support structure. The last of these three can be split in the tower and the foundation or substructure, the latter often defined as starting from the waterline down to the seabed, and even containing the structure's foundations below the seabed. To connect the tower to the substructure a transition piece (TP) is often used. This TP allows boat landings and facilitates connection and force transfer, mainly overturning moment and weight of the rotor, nacelle and tower, to the foundation, which runs through the water and into the seabed. The hydrodynamic loads act on the region of the foundation from the water line down to the seabed.
This is visualized in Figure 2.1.

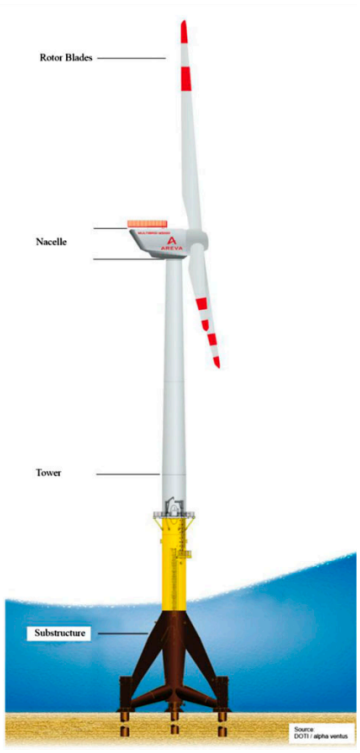


Figure 2.1: Wind turbine component overview (Jaax, 2016)

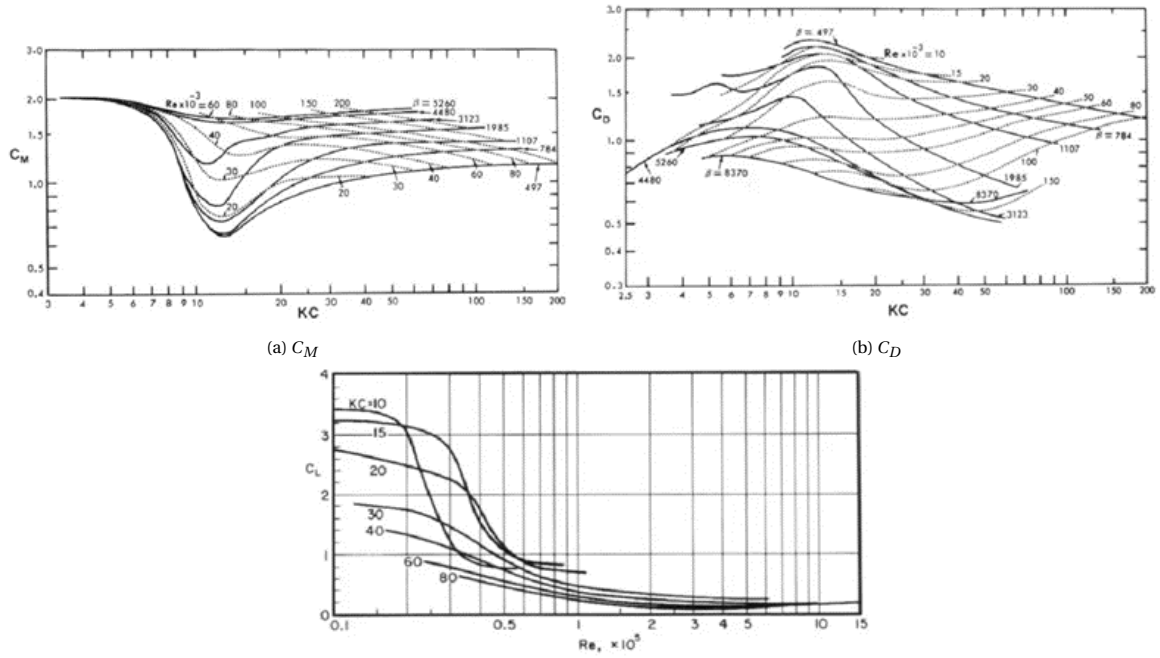
2.1.1. Primary OWSS design considerations

Morison and force determination

The hydrodynamic loads on monopiles are a consequence of constant velocity current and variable velocity wave flow (Journée and Massie, 2008). Simplifying the circular wave motions to a horizontal flow the hydrodynamic forcing can be calculated in a number of different ways. Other than doing physical or numerical experiments, theory-based approaches based on diffraction theory are also possible. More often however design follows the vision of "close enough is good enough", and so design standards are based on the Morison Equation, which has an empirical basis through the use of drag and inertia coefficients (Sumer and Fredsoe, 2006; Chakrabarti, 2005).
The Morison equation reads:

$$F = C_M \rho \frac{\pi}{4} D^2 \dot{u} + C_D \frac{1}{2} \rho D u |u| \tag{2.1}$$

with ρ, D and u being the fluid density, pile diameter and flow velocity respectively. A similar approach can be used for the lift determination. The Keulegan-Carpenter (KC) based inertia (C_M), drag (C_D) and lift (perpendicular to flow, C_L) coefficients can be seen in Figure 2.2. Some flow regimes do allow one of the coefficients to be neglected, but general design procedures provide box rules for their applications. When deleting some constant terms, the main value that we are looking for in the, for monopiles at high Reynolds numbers, drag-coefficient is 0.2.



(a) C_M

(b) C_D

(c) C_L

Figure 2.2: Morison coefficients for a smooth cylinder (Chakrabarti, 2005)

The combination of waves and currents is mainly important in the wave-affected region. As depth increases the influence of wave motions approaches zero. The North Sea is one of the most used water areas for offshore wind (IEA, 2019). While being a current- and tidal-sensitive water body, she only experiences a maximum of around 2.4 kts or 1.3 m/s of current (Waterkaart.net, 2023). Wave motions on the North Sea, assuming Airy waves and thus using orbital velocities of $u_{max} = \frac{2\pi a}{T}$, with a the wave amplitude and T the period in seconds, can give flows of up to 5.36 m/s (O'Boyle et al., 2015). This indicates that any optimization of hydrodynamic forces can best be performed around the waterline or splash zone, since the waves provide the highest loads, especially considering the quadratic velocity relations in Equation 2.1. If optimizations are to be done in the splash zone, it could be possible to include optimization-techniques not in the monopile itself, but rather in the transition piece. The main loading in this splash zone additionally indicates that the structural weakening from perforations or similar hydrodynamic force reduction solutions may be acceptable since the critical overturning moments will only come up below the heavily wave loaded area.

Structural considerations

Once the loads have been determined the structural analysis can begin. However, when including fluid-structure interaction, this becomes an expensive, iterative process since the Morison equation relies on relative flow velocities. In most design standards the structure can be assumed rigid. Simplified structural models are most often based on

Rayleigh Stepped Towers (Hermans and Peeringa, 2016). For deep water applications this simplification is in good accordance (<4% error) with elaborated FEM models (van der Ploeg, 2021). For the remainder of the thesis the structural modelling of perforated towers lies out of the scope. However structural considerations will be taken into account when further determining relevant perforated designs in order to smartly allocate computational resources.

Expected results from previous work

The focus here lies on the perforated monopile concept as introduced by Andersen et al. (2020), shown in Figure 1.1. Alternative concepts for exploration of deeper waters for wind energy, such as the variety of floating concepts or tri-piles, are omitted. The Andersen concept has experimentally shown a reduction of the hydrodynamic load compared to a closed cylinder is possible of up to 20%, with even more reductions foreseen for higher KC numbers. The experiments were carried out in irregular wave basins and compared to the experimental and theoretical (Morison) closed-cylinder results. The last two were within a few percents of each other, well within the measurement uncertainty.

The concept was more practically looked into by van der Ploeg (2021) who, using 0-120 m water depths, determined that fatigue limits would be critical for both traditional and perforated monopiles. Using CFD modelling and structural analysis he showed a possible increase of monopile foundation depths to 87 m under DNV design rules. At this point the foundation would contribute up to 25% of the turbine costs. It was shown that a 27% weight and 17% cost reduction could be obtained compared to the non-perforated variants at the same depth. One of the other key conclusions of the hydrodynamic design was to avoid resonance and to adapt the structure so that the first natural frequency would lie around the 0.2 Hz mark. This frequency is one to pay attention to in further hydrodynamic and turbulence analysis. Structurally he showed the stress concentration factor and dynamic amplification factor to contribute to stress increases by a factor of 3, assuming a damping of 5%.

The main line of work on which this thesis is built is that of Star (2022). Instead of using OpenFOAM or other prebuilt solvers the focus was to develop a FEM CFD model in Julia using the Gridap.jl package. The applied flow theory used was Stokes creeping flow, which disregards inertia and turbulence. The advantage is the linear nature, which speeds up calculations. Accurate Stokes flow only applies to low Reynolds number situations. However, this is not the case for XXL monopiles. It is mentioned that the use of this model as an initial condition for more elaborate solvers could reduce the number of iterations needed, thereby reducing the computation time. This will be implemented in the full transient solution later. The CFD database he created, based on parametric design of the perforated monopiles (diameter D, number of perforations n, porosity β) was used to train a CNN surrogate model. An additional design parameter was the angle of attack of the first perforation α. For $n \geq 4$ this was found to not be of influence. Since the indicated design by Andersen et al. said that low numbers of perforations would be too structurally inefficient, this constraint will most likely not pose any problems. However, this should be verified in more complex, transient flow simulations.
The linear calculations showed hydrodynamic reductions of up to 40% for large values of β and $n \geq 3$. The surrogate model sped up the calculations by a factor of almost 400 (386), which is expected to increase even further for turbulence-inclusive models.

2.1.2. Secondary OWSS design considerations

The secondary design considerations are design aspects which, following a variety of literature, are of influence in hydrodynamic load optimization, but which are not key-determinants. For full scale design they should be taken into account. Most design aspects are even legally required to be considered to obtain official design approval and certification. However, in this proof of concept phase the following considerations can be excluded in force determination projects. Some of these aspects are currently being studied in parallel theses at TU Delft.
For now we can disregard:

- Manufacturing: size limitations, wall thickness optimization, lifting capacity, hole-cutting solutions
- Installation and noise mitigation: hammering techniques, buckling prevention, up-ending schemes, dbA limitations
- Scour: horseshoe vortices with perforations, DNV updates for XXL piles, temporal evolution modelling using RANS or LES
- Corrosion and acidification: sustainability concerns, longevity
- Marine growth: wall roughness and turbulence, detrimental steel effects, hole-reduction in oxygen rich splash zones

2.2. Fluid modelling for monopiles

The perforated monopiles will have large amounts of water passing through them. It is expected that this flow will be nonlinear and nonlaminar. A complex interaction of eddies and vortices similar to the flow through an orifice meter or a double-sided submerged jump is likely. To solve these vortices, not only a flow equation will be needed, but turbulence modelling will make up the bulk of the required work. To model this the Navier-Stokes equation, only requiring assumption of a Newtonian and non-compressible fluid, is most common (Elger et al., 2016). To this day, no closed solution in 3D has been found. This makes that even numerical solvers rely on varying degrees of simplification.

2.2.1. The correct simplification fluid model

There are five main ways of modelling fluids, with increasing amounts of computational cost. This is shown in Figure 2.3a.

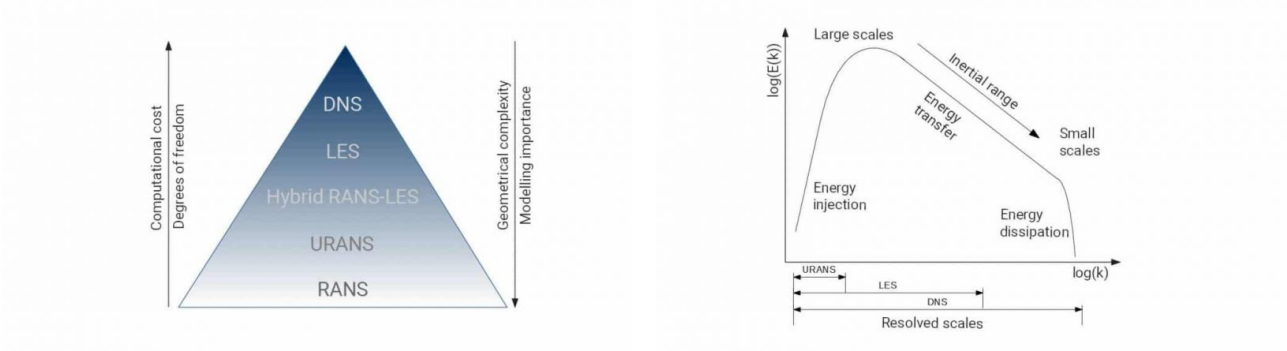
1. RANS: Reynolds Averaged Navier-Stokes
   By averaging small scale pressure and velocity fluctuations the technique gets rid of some nonlinear effects (Reynolds stress). Lower order turbulence models (closure models) use kinetic energy to solve the unresolved terms ($k-\omega$ or $k-\epsilon$). The technique assumes that the unresolved scales only contribute marginally to the overall flow, following the Kolmogorov energy cascade. This is visualized in Figure 2.3b.
   As a drawback, RANS solvers are known to be mathematically inaccurate for reversing flows (Launder and Spalding, 1974). In engineering this excludes them from detailed modelling of flows with Reynolds numbers over 1 million (Duan et al., 2019). For a 1 m/s water flow this thus excludes any and all cylinders with a diameter exceeding 1 m: $Re = U \cdot D/\nu$ with ν = 1e-6 being the dynamic viscosity of water .

2. URANS: Unsteady RANS
   A RANS adaptation better suited for periodic effects or moving structures.

3. DES: Detached Eddy Simulations
   A RANS solver with LES-solved cells near the boundary layers.

4. LES: Large Eddy Simulations
   The flow field is split into a resolved and unresolved part using a certain kernel filter (Deardorff, 1970). A lower order sub-scale-model approximates the flows within the cells. LES can solve actual turbulence, but it comes at a computational cost over RANS. Closure models depend on empirical relations or, more recently, numerical artefacts.

5. DNS: Direct Numerical Simulations
   This is the most accurate but also most expensive simulation type. All time and spatial scales are resolved. No turbulence model is used (Orszag, 1970). However DNS modelling is not often used in full scale engineering applications due to the cubed scaling of computational costs for increasing domain sized or mesh refinements.



(a) Computational cost comparison for different fluid models

(b) Energy level in turbulent fluid flows

Figure 2.3: Fluid modeling technique comparison (Ideal Simulations, 2020)

Which one is the right solver depends entirely on the application of the model to be made. Ideally results should be compared using multiple solvers. For engineering RANS is generally sufficient. However, in case of questions concerning actual turbulence the eddy-viscosity closure models from RANS are unreliable. LES is then needed for its added accuracy and relative cheapness compared to DNS. While combination techniques such as DES or newer developments such as, for now less accurate, RSM could be applied as well, LES is the safest choice for this study.

2.2.2. LES details and application

The LES simplification is based on filtering certain smaller effects and simplifying the solutions of the filtered, non-resolved, sub-scales. Filtering can be done implicitly or explicitly. Explicit filters are built into the Navier-Stokes equation directly using a specific kernel function G. The most known is the box filter, filtering eddies smaller than a given size. The filter cutoff is immediate through the use of a Heaviside function. Alternatively a normalized Gaussian filter can be applied. While the continuous bell-shape can give better stability in the domain the computational cost is also considerably larger (Greenshields and Weller, 2022). More often however implicit filtering takes place through the selection of the grid size. Grid-size filtering is in essence the implicit variant of box filtering, cutting off any and all effects that take place within a single cell (ϕ'). The non-filtered flows $\bar{\phi}$ are than the only resolved ones. In formulas this becomes:

$$\phi(\mathbf{x},t) = \overline{\phi(\mathbf{x},t)} + \phi'(\mathbf{x},t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \phi(\mathbf{r},\tau)G(\mathbf{x}-\mathbf{r},t-\tau)\,d\tau\,d\mathbf{r} + \phi'(\mathbf{x},t) \tag{2.2}$$

with ϕ the full spatial and temporal flow field, $\mathbf{x}$ the spatial coordinate and t the temporal one. G is the kernel functio:n

$$G(\zeta) = \frac{1}{\Delta} H\left(\frac{1}{2}\Delta - |\zeta|\right) \tag{2.3}$$

with Δ the filter size, implicitly equal to the grid cell size, and H being the Heaviside function. In practice G can be expanded to include the time with cutoff time scale τ_c as well.

Once the small effects (ϕ') have been filtered out the Navier-Stokes equation will have difficulty closing, having no stable solution. For this a closure model is needed. These are mostly based on empirical relations. Originally Boussinesq proposed the use of eddy-viscosity models (Launder and Spalding, 1974). These were adapted by Prandtl to convey the concept of "turbulence added viscosity". Physically this adds a mixing length and boundary layer for wall-bounded flows (Prandtl, 1928). With the rise of numerical simulations these were expanded even further. The most commonly used closure models are:

- SL (Smagorinsky-Lilly)
  Determines effective viscosity based on the Smagorinsky constant C_s.
- DSGS (Dynamic Subgrid Germano Scale)
  A modification of SL to dynamically vary the Smagorinsky constant C_s based on a grid and test filter function. In 1992 Lilly again adapted the method using a least-squares approach in order to improve accuracy (Lilly, 1992).
- WALE (Wall-adapting local eddy viscosity)
  Similar to SL this is a constant-based approach using the strain tensor values.

WALE theoretically combines accuracy with additional physical basis (Lilly, 1992), and is, from literature therefore regarded as one of the best choices in this case. This should however be verified by numerical tests.
Important to note is that, due to local refinement in CFD programs, the near-wall cells are often very small. This reduces the closure efficiency, especially for SL models. The solution for this is to adapt a filter length scale using either Van Driest damping or a Zonal Layer approach. This way the turbulence induced viscosity remains non-asymptotic.

Newer applications of closure models use numerical artefacts. The most known is the EMAC form, which uses energy and momentum conservation properties (Varela et al., 2022; Charnyi et al., 2019). This work got picked up to improve energy dissipation in finite element calculations specifically (Lehmkuhl et al., 2019). Most numerical stabilization schemes use the Variational Multiscale Method. However the upwind numerical stabilization introduces nonphysical damping. To solve this, new energy-conserving numerical scheme had to be made (Charnyi et al., 2019). This makes use of estimated eigenvalues in order to calculate the optimal time step for the numerical integration. The scheme thus uses a fractional time step. This method focuses on the pressure equations, since velocity accuracy is not affected.
A similar approach has been developed by Colomés et al. (2015), focusing on ILES (implicit LES). They found that, for sometimes questionable convergence, the computational cost is comparable to simpler LES closure models but the accuracy and numerical stability is increased. For turbulent channel flow, most representing the expected CFD simulations this thesis uses, the most efficient and stable method was found to be "orthogonal subscales" (Dyn-Nl-OSS), generating results almost identical to DNS. Follow-up research into the stabilization showed that the stabilization parameters c_1, c_2 and c_c are best taken as 12, 8 and 32 respectively (Colomés et al., 2016).

A selection of the models described above will be implemented and tested for further research in the coming sections.

2.2.3. Secondary hydrodynamic flow effects

As should be clear by now, numerical fluid modelling is a constant trade-off between computational cost and accuracy. In light of the research question, some ways to cut costs to increase accuracy are to exclude certain effects and design choices, of which a list is provided.

1. 3D modelling
   For cylindrical structures 2D sections have been proven to be very representative in dimensional scale-ups (Nguyen et al., 2021; Alagan Chella et al., 2019; Bianchini et al., 2017; University of Strathclyde, 2015). Conversion rules for 2D to 3D Morison coefficients are available in the mentioned literature, or can be generated by comparing coarse to fine 2D (cheap) results, and projecting them onto 3D coarse results in order to predict the 3D finer mesh outputs (expensive). Either way for conceptual design of novel concepts a 2D simulation should prove sufficient and useful for further large-scale design.

2. FSI, VIV and resonance
   Different sources state different levels of relevance for the inclusion of hydro-elastic calculations for XXL monopiles (Gupta and Basu, 2020). Most sources advise one-way coupling from relative inflow velocity, without changing the flow around the moving structure (Taylor, 2007). For detailed design other, more recent, sources say to always include FSI, especially in later design stages or for ice loading (Y. Liu et al., 2022). This is because some monopiles simulations and physical experiments have shown dynamic amplification factors of 19-34% (Tödter et al., 2021; J. Zhang et al., 2010). The former does state that the DAF (Dynamic Amplification Factor) for surface-piercing cylinders can be considerably lower than the just indicated range. This is explained through multi-phase effects of the free-surface and damping because of that. Still the here proposed 2D simulation would not be influenced by these free-surface effects, making DAF of up to 34% a number to keep in mind if FSI were to be applied in the future.
   When FSI is excluded, this closes the door to accurate research into Vortex Induced Vibrations (VIV) and resonance. As a prediction the induced turbulence from the perforations themselves will most likely reduce the VIV amplitude, like helical strakes would.

3. Fatigue effects
   This is omitted in this fluid focused research, especially since fatigue is closely related to FSI, VIV, and vibrations in general. Possible issues include residual stresses from the cutting processes of the perforations, influence of marine life, inadequate fatigue or marine standards for the used weld or cut types, ... At this moment most research of fatigue on perforations only looks into small (mm order) perforations. Their relevance for monopile (m order) perforations remains questionable, and could provide interesting research in the future.

3

Numerical modelling of fluid flow

In this chapter the implementation of the previously discussed fluid solvers is covered in more detail. Section 3.1 explains the software that is used and how the domain is set up. It forms the theoretical basis for Section 3.2 where the domain is optimized and the results are validated. Lastly Section 3.3 discusses the engineering relevance and trends of the performed simulations as a whole.

3.1. Fluid modelling implementation: Gridap.jl FEM and LES-VMS

3.1.1. Gridap.jl FEM solver

With knowledge of the effects to model from Chapter 2, the next step is to define the numerical scheme with which the solution can be calculated. In this thesis the original assumption or approach was to use Gridap.jl, a Julia based Finite Element Modelling (FEM) package. While other solver schemes for the Navier-Stokes equation exist, FEM proves to be the most versatile and widely applicable for perforated monopiles from the ones considered. Below is a short expansion on this finding.

- VOF and FVM
  Volume of Fluid (VOF) models fluid motions based on the Navier-Stokes equation as a function of the liquid to air ratio in each grid cell, and the related pressure and velocity field (Wellens, 2012). The donor-acceptor scheme is mostly focused on free surfaces, and is known to be unstable for larger time steps, so close monitoring of CFL limits is required. Additionally VOF is, due to its embedded structure, less suitable for inclusion of FSI effects (Méchartes, 2021), although recent developments at Deltares in the ComFLOW solver have shown promising results (Deltares, 2018).
  Similar conclusions can be drawn for FVM (Finite Volume Methods).
  Both VOF and FVM are unable to include higher order elements, limiting their numerical accuracy for complex shapes.

- SPH
  Solid Particle Hydrodynamics tries to model fluid as a set of solid, elastic particles (Hu and Han, 2017). However, SPH has become slightly outdated due to the limited ability to include multi-phase flow, which would be required when updating the perforated simulations to 3D free surface cases.

- FD
  While being generally beneficial for understanding numerical phenomena, the simple Finite Difference method is less often applied in CFD calculations due to its limitation to include complex shapes (Guan and Liu, 2018).

- DG
  The Discontinuous Galerkin method combines elements of FD and FEM, with many successful applications in fluid mechanics (Arnold et al., 2001). The method has been on the rise for LES solvers in the last decade due to its accuracy and energy-accurate predictions (Fernández et al., 2018). However, the method application is often computationally expensive. While implementations for GPU usage exist, for most CPU based calculations there is little added benefit over FEM itself. Because of the velocity-pressure coupling in incompressible Navier-Stokes calculations there is a need for iterative solutions to the system matrix, increasing the computation times. When increasing the order of the formulations the number of degrees of freedom explodes. Additionally there is no full solution found for accurate implementation of diffusion terms, meaning that the DG method is not yet optimal

for transient problems. A solution to the expense of implicit matrix inversion for these large matrices could be to use a more explicit scheme, but often the CFL (time step) limits that this imposes makes DG methods for transient problems inefficient (Manzanero et al., 2018).

- FEM
  Finite Element Modelling is the most complicated and elaborate option described. However, with its complexity comes flexibility as well (Guan and Liu, 2018). For CFD an FEM approach provides the most all-round options, with relatively easy inclusion of FSI, multi-phase flows, and 2D to 3D scaling (Cadence, 2022). Additionally it is the best option for complex shapes, which is one of the main reasons why it is suited for research into perforated monopiles.

Like mentioned before the FEM solver of choice in this case, as a starting assumption or prerequisite of the thesis project, is Gridap.jl. The Julia based solver and its open source nature allow one to adapt the source code if needed for stability or accuracy purposes (Verdugo and Badia, 2022; Badia and Verdugo, 2020). For a new development such as perforated monopiles this feature is of key importance since chances are that no direct implementation in Gridap.jl is possible. The choice for this specific Julia based Gridap.jl solver can be additionally justified because of three reasons:

1. "Gridap.jl provides a set of tools for the grid-based approximation of partial differential equations (PDEs) written in the Julia programming language (Badia and Verdugo, 2020)." This means that while the focus here will lie on CFD, the tools are intrinsically capable of incorporating structural effects as well. This model scaleability is the main reason to use Gridap.jl. This limits the need for crossover-software, as is often the case for other models or software packages (OWF Alpha, 2022; Mendoza et al., 2022; Murphy et al., 2018; Hermans and Peeringa, 2016). Especially for post-processing or surrogate modelling this can become useful.

2. The Julia language itself provides a large potential of becoming a widely used scientific programming language due to its Python-like user-friendliness and its FORTRAN and C-languages speed (Soleimani et al., 2022). Earlier tests showed execution times 1.2-1.5 times greater than C, with Python lagging behind at 9-10 times the computational cost of a compiled C-language (Suryad, 2022; Christianson, 2020).

3. In order to build on previous work at TU Delft, done in Gridap.jl, the continuation using the same packages is logical. Although TU Delft often uses Plaxis3d for FEM simulations (Murphy et al., 2018), the added work to a young open source package has additional benefits for the scientific community.

3.1.2. Internal definitions for the calculation scheme

Domains and boundaries

The domain is visualized in Figure 3.1. This domain forms the basis of all the simulations made and used in this thesis.

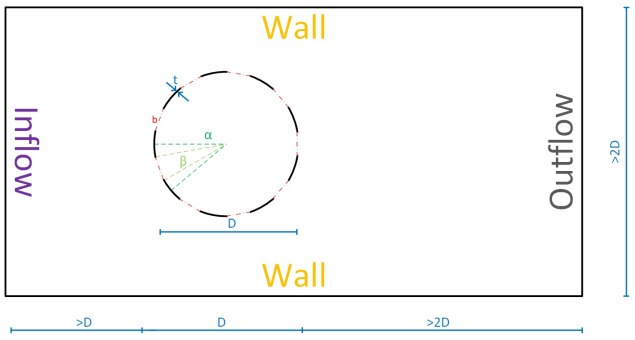


Figure 3.1: Updated domain visualization in 2D

Domain shape

The general domain shape used in CFD simulations is rectangular (Ideal Simulations, 2020; Duan et al., 2019; Bianchini et al., 2017; Walters et al., 2013; Wellens, 2012). This will be used here as well because of the easy definition and meshing references, as well as the better links to alternative research for validation purposes.

Inflow
Flow around cylinders requires a fully developed flow at the location of the intersection. While often parabolic (Poiseuille) inflow conditions are used, here a constant uniform inflow makes more sense when looking at the open-ocean area where monopiles are usually found. The flow velocity definition makes this a Dirichlet boundary, mathematically defined as:

$$\vec{V}_{in} = \begin{bmatrix} V_{in} \\ 0 \end{bmatrix} \tag{3.1}$$

where V_{in} is a constant scalar. To ensure the uniformity of the inflow the length from the inflow boundary to the monopile must be large enough. Since this comes at a computational cost, this inflow length will be optimized in the validation phase from an initial 1 diameter distance.
Later steps both in 2D and 3D would require the introduction of time-dependent inflow and outflow control with the idea of simulating periodic wave motions in deep waters.

Outflow
The outflow boundary is defined as a Neumann boundary which imposes a zero pressure gradient at the domain end (Flow3D, 2022; Kim et al., 2014). This boundary condition is defined through the last term of the weak form, as shown later in Equation 3.13. For non-periodic inflow conditions no damping schemes are required.

Domain Walls
With the uniform inflow the wall conditions are predefined as Dirichlet bounds which set the fluid velocity at the boundary to the inflow velocity in x-direction, while still keeping the cross-current y-velocity set to 0 (Haiderali and Madabhushi, 2012):

$$\vec{V}_{wall} = \begin{bmatrix} V_{in} \\ 0 \end{bmatrix} \tag{3.2}$$

Depending on the applied LES closure model, some simulations perform better with the wall-outflow corners considered at the Dirichlet wall, or in the Neumann outflow bound. Verification of the origin of instability in simulations which crash or give faulty results is required to check that the instability does not originate from a numerical fault in these two corner points.
As a side note, it must be said that in a real life application this boundary condition is not completely accurate. In open seas the water would be able to cross this virtual bound in y-direction, so a velocity-inducing Dirichlet bound in x-direction with a free traction Neumann bound in y-direction would be physically more correct. However, as to not over-complicate the boundary models, which would increase the computational cost even further, it makes sense to assume a full Dirichlet bound in both directions, and to assure that the domain is wide enough so that the y-velocity constraint is not a key factor in determining the overall fluid flow. In the future it could be interesting to re-run some simulations with these adapted boundary conditions, and to see whether or not a computational gain can be made from a narrower domain with these Dirichlet-Neumann combined walls.

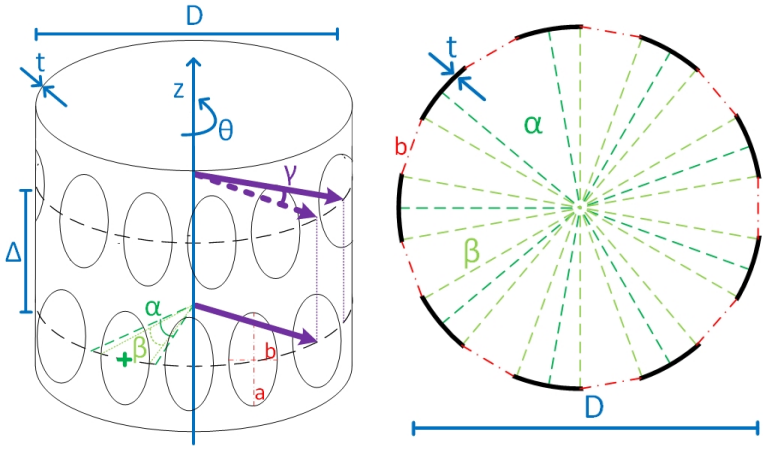
Structural Walls
Like mostly used for domain boundaries, the structural boundaries from the monopile and its perforations will be of the no-slip, no-penetration Dirichlet type:

$$\vec{V}_{mp,local} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.3}$$

For the rounded monopile surfaces it is important that the directions of these two conditions depend on the location and orientation of the element surface on the structure. This means that in general no immediate relation can be made between the in- and cross-flow direction, as can be done for the side walls (Greenshields and Weller, 2022; Elger et al., 2016). The monopile is placed in the centre of the domain width.

Monopile definition

The original parametric cylinder was based on the number of perforations, angle of attack, and porosity (Star, 2022). This idea will be continued here with some minor changes. In the original work the angle of attack was found to be of no influence. However, this conclusion will be verified in this thesis.
The parametric design has now been set up with model expansion in mind. This means that the initial parameters allow full 3D definition and scaling of the cylinder. Cylindrical coordinates are needed rather than radial ones. Almost all parameters are made to scale with the diameter, and rely on internal relative factors otherwise. This ensures easy and automatic scaleability in the future. A visual overview can be seen in Figure 3.1 and Figure 3.2, with the numerical explanations and design space boundaries in Table 3.1. In these figures and table the 2D relative scaling is explained as well.



(a) 3D (b) 2D

Figure 3.2: Parametric cylinder sketch

Table 3.1: Parametric cylinder definition

2D				
Parameter	**Symbol**	**Determination**	**(Factor) Design values**	**# samples**
Diameter	D	-	$D = 8, 9, ..., 12$ [m]	5
Wall thickness	t	$t = \frac{D}{R_t}$	$R_t = 80, 90, ..., 120$[−]	5
Sector angle	α	$\alpha = \frac{2\pi}{n}$	$n = 3, 4, ..., 20$ [-]	17
Horizontal opening	b	$b = D \cdot \beta/2$; $\beta = R_\beta \alpha$	$R_\beta = 0.1, 0.2, ..., 0.9$ [-]	9
				Total: 3825
3D				
Parameter	**Symbol**	**Determination**	**(Factor) Design values**	
Vertical opening	a	$a = R_a b$	$R_a = 1.1, 1.2, 1.3, 1.5, ..., 6.5$ [-]	
Row rotation	γ	$\gamma = R_\gamma \alpha$	$R_\gamma = 0.1, 0.2, ..., 0.5$	
Row separation	Δ	$\Delta = R_\Delta a$	$R_\Delta = 1.1, 1.2, 1.3, 1.5, ..., 6.50$ [-]	

A few notes must be made on these definitions:

- The unique variables at the basis of the parametric design are thus $D, R_t, n, R_\beta, R_a, R_\gamma, R_\Delta$. The angles are given in radians. The angle of attack then requires a separate definition, but it does not really change the cylinder design itself.
- The horizontal opening b is defined along the original circular curvature. It is therefore not the straight line distance (chord length) of the opening. This is done to facilitate porosity calculations. For large values of n the differences will be negligible. The formulation follows from:

$$n \cdot b = \text{Circumference} \cdot \text{Total opening angle ratio} = \pi D \cdot \frac{\beta \cdot n}{2\pi} = n \cdot \frac{D \cdot \beta}{2} \tag{3.4}$$

- The porosity, often called by the symbol β, would be R_β in case of the 2D designs. For the design the change in notation was done to keep uniformity in the Greek indicated angles and the R indicated ratios. In later plotshowever simply β will be used. In 3D this would become, using an along-the-curve opening-area approach from the middle of one horizontal perforation row to the middle of the next, substituting a, b, Δ to obtain the basis variables:

$$R_{\beta,3D} = \frac{A_{open}}{A_{total}} = n \cdot \frac{\pi a b}{D\pi} \cdot \frac{1}{\Delta} = \frac{\pi^3 R_a D^2 R_\beta^2 / n}{\pi^2 R_\Delta R_a D^2 R_\beta^2 / n} = \frac{\pi R_\beta}{R_\Delta} \tag{3.5}$$

- The variables R_a and R_Δ are not evenly spaced. Assuming that the changes in a closely knit perforation pattern are more significant than when cut-outs are already very long (for large R_a) or very far apart (for R_Δ), their step size is increased in a decimal Fibonacci manner: 1+1/10=1.1, 1+2/10=1.2, 1+3/10=1.3, 1+5/10=1.5, 1+8/10-1.8, 1+13/10=2.3, ... This is a matter of experimental preference and curiosity.

- In theory the value R_Δ can be varied smaller than 1 for certain values of γ. In that case two consecutive rows would partly slide into each other. The exact limitations and structural boundaries associated with this however lead us to not explore these options here.

In total this results in 3825 samples. As mentioned in Subsection 2.1.1 the maximum, realistically, expected water velocity is 5.36 m/s from the wave motion, with up to 1.3 m/s currents. It is thus interesting to analyse these samples at different, evenly spaced, incoming water velocities up to 6.5 m/s, here proposed as $V = 0.5, 1.5, ..., 6.5$ amounting to in total 26775 design cases. By smart deductions and relation finding we will reduce this number to allow a reasonable allocation of computation resources.

Meshing scheme

For the numerical calculations the domain is split into an array of elements over which the Navier-Stokes equation, with all related LES adaptations, can be solved. In Gridap.jl meshing takes place, usually, via GMSH.jl (GMSH, 2022). The package makes a conformal mesh using (as default) unstructured grids through the Delaunay algorithm. Conformal indicates that nodes are always matched on either side of the boundary lines (Cadence, 2022). For 2D cases the meshing method makes use of triangular cells. Similarly expansion to 3D would generate tetrahedral cells (Wollblad, 2018). These triangles are easier than squares or rectangles for the meshing stages itself, but are more prone to giving skewed results. As indicated earlier there will not be the initial requirement for FSI inclusion, so a fixed mesh rather than an adaptive one can be used, which limits the risk of getting skewed results.

As said, implementation of GMSH.jl in Gridap.jl is quite standard. An important part of the meshing will be to choose the right resolution or mesh density. The right resolution is the one that is as coarse as possible while accurately capturing all the intended physics. Generally the grid is made more dense using steps of a factor 2 or 3, in 2D requiring 4 or 9 times as much computational power, with possible added computational increases from CFL limits. The increase must be done until further increases do not yield different or more accurate results. The exact definition for refinement is a disputed fact in literature (Nguyen et al., 2021; Haddoukessouni, 2021; Ideal Simulations, 2020; Duan et al., 2019; Bianchini et al., 2017; Walters et al., 2013). For engineering applications one possible answer is to consider the mesh converged when the quantity of interest experiences changes lower than 1% when halving the cell size (Kuron, 2015). This “convergence process” will be performed later on.
To optimize this, local refinement is possible. That way one uses the bulk of the computational power in the high-interest areas. In the case of perforated monopiles the refinement should thus be focused around the steel structure. Additionally it is argued that the wake-region will be important as well. This makes for the need of a refinement scheme that (1) captures the intended physics, (2) has a refinement function linked to the distance to the structure, and (3) is more refined in the cylinder wake.

LES-VMS calculation scheme

For nonlinear effects such as turbulence the use of a transient fluid flow solver is needed (Elger et al., 2016, Wellens, 2012). This means that the Navier-Stokes equations must be solved at each time instance, after which a numerical integration scheme is applied to find the velocity and pressure values in the next time step. The general way Gridap.jl does this is as follows (Verdugo and Badia, 2022; Badia and Verdugo, 2020):

1. Set up the problem
   (a) Define the model parameters and input values
   (b) Define the geometry
   (c) Define the boundary conditions
2. Set up the numerical scheme
   (a) Set up the test spaces with the right conformity, for both velocity (H1 conformity) and pressure (C0)
   (b) Build trial spaces for the velocity and pressure fields
   (c) Combine them in a (transient) multifield
   (d) Initialize the domain
   (e) Define the weak form
   (f) Set up the time integration scheme
3. Apply the nonlinear solver and time integration
4. Do post-processing

For the engineering research into perforated monopiles the post processing should be built around the following data:

- The velocity and pressure fields (usually in .vtk format). This will be useful in the early stages of the design to look into the exact flow effects, eddies, ...
- Drag and lift time-traces, either in the time or frequency domain, for further design analysis and trend-seeking.

A tutorial on the implementation of this scheme in Gridap.jl is added in Appendix A.

Setup of the problem
The model gets 3 main inputs: the mesh, the inflow velocity, and the time step. At this stage Gridap.jl does not allow the useof variable time-stepping. Some auxiliary inputs define the output storage, but are computationally irrelevant.
The mesh is used as the central input. It is triangulated and split into a fluid and a solid domain. Additionally the boundaries are read from predefined tags from the mesh-file. The bounds of all these sub-domains are "Measured" to be used in integration schemes later, and additionally the normal directions of the structure and the outflow boundary are calculated.
The boundary conditions are as described above: a uniform (Dirichlet) inflow of speed $(V_{in},0)$, prescribed-velocity $(V_{in},0)$ walls and a no-slip-no-penetration structure $(0,0)$. The outflow boundary is included in the weak form directly.

Making the multifield
With the domain fully defined the next step is to define the internal structure of each cell through reference elements in their respective transient test spaces. With one for velocity (in 2 dimensions), and one for pressure (1 dimension), a multifield test space and trial space is created.
The velocity reference elements in the test spaces are Lagrangian second order elements. For numerical stability, this means that the pressure elements should be Lagrangian elements as well, but of order 1 (Star, 2022). These 2 test spaces have H1 and C0 conformity respectively. H1 conformity indicates that the test space functions are continuous, although their gradients can contain jumps. C0 conformity allows the test space to contain jumps already, which is required for the pressure field.

Domain initialization
To facilitate the calculation start-up, it is better to not start from a no-flow situation, where all velocities are 0. To have an initial state, a Stokes solver is ran once. The strong form of the steady state Stokes equation is:

$$\begin{cases} -\mu\Delta\mathbf{u}+\nabla p=\mathbf{f}, & in\Omega \\ -\nabla\mathbf{u}=0, & in\Omega \end{cases} \tag{3.6}$$

where $\mathbf{u}$ denotes the velocity field, 2 dimensional, and p denotes the pressure field. μ is the incompressible fluid viscosity. Multiplying the test function $v\in\mathbf{H}_0^1(\Omega)$ to the former momentum equation on the top, and doing the same for the continuity equation on the bottom with $q\in L^2(\Omega)$, we can find the weak form problem through integration by parts. The problem then reads:

Find $\mathbf{u}\in\mathbf{H}_0^1(\Omega)$ and $p\in L^2(\Omega)$ such that:

$$\begin{cases} (\mu\nabla\mathbf{u},\nabla\mathbf{v}-(p,\nabla v)=\langle\mathbf{f},\mathbf{v}\rangle, & \forall v\in\mathbf{H}_0^1(\Omega) \\ -(\nabla\mathbf{u},q)=0, & \forall q\in L^2(\Omega) \end{cases} \tag{3.7}$$

Finally this can be combined into the bilinear form:

$$\begin{cases} a(\mathbf{u},\mathbf{v})=\mu\int_\Omega\nabla\mathbf{u}:\nabla\mathbf{v}d\mathbf{x} \\ b(\mathbf{v},\mathbf{q})=-\int_\Omega(\nabla\mathbf{v})\,q\,d\mathbf{x} \end{cases} \tag{3.8}$$

Rewriting this such that the formulation includes the outflow boundary condition and the non-forced fluid flow inside the domain, and then converting the inputs so that the form is equal to the one implemented in Gridap.jl, the problem to be solved becomes:

Find $\mathbf{u}\in\mathbf{H}_0^1(\Omega)$ and $p\in L^2(\Omega)$ such that

$$a\left((u,p),(v,q)\right)=\int\left(\epsilon(v)\odot\left(\sigma_{dev_f}\cdot\epsilon(u)\right)-(\nabla\cdot v)\cdot p+q\cdot(\nabla\cdot u)\right)d\Omega_f \tag{3.9}$$

with $\sigma_{dev_f}(\epsilon) = 2 \cdot \nu_f \cdot \epsilon$ and ϵ the symmetric gradient of the field; and such that

$$l\big((v,q)\big) = \int (0)\, d\Omega_f \tag{3.10}$$

in which the 0 indicates a free fluid flow inside the domain.

Weak form
In this thesis a total of three LES schemes were analysed: Smagorinsky-Lilly, WALE, and OSS-ISS (VMS-ILES). Each of these schemes has their respective weak form, given below. Very soon it was clear that the first two options were a lot less stable than the VMS variant. Most likely they require much smaller time steps to overcome this deficit. Therefore further tests only use the VMS option.

The problem formulation starts with the Navier-Stokes equation:

$$\begin{cases} \partial_t \mathbf{u} - \nu \Delta \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = \mathbf{f}, & \text{in } \Omega \times (0,T) \\ -\nabla \mathbf{u} = 0, & \text{in } \Omega \end{cases} \tag{3.11}$$

with ν the kinematic viscosity. Using the same space definitions as for the Stokes initialization, the weak form then reads:

Find $\mathbf{u} \in \mathbf{H}_0^1(\Omega)$ and $p \in L^2(\Omega)$ such that:

$$(\partial_t \mathbf{u}, \mathbf{v}) + B(\mathbf{u}; [\mathbf{u}, p], [\mathbf{v}, q]) = \langle \mathbf{f}, \mathbf{v} \rangle \qquad \forall v \in \mathbf{H}_0^1(\Omega) \cap \forall q \in L^2(\Omega) \tag{3.12}$$

where $B(\mathbf{a}; [\mathbf{u}, p], [\mathbf{v}, q]) = \nu(\nabla \mathbf{u}, \nabla \mathbf{v}) + c(\mathbf{a}, \mathbf{u}, \mathbf{v}) - (p, \nabla \cdot \mathbf{v}) + (q, \nabla \cdot \mathbf{u})$. The trilinear weak form $c(\mathbf{a}, \mathbf{u}, \mathbf{v})$ has multiple definitions, which may depend on the closure model used, as is highlighted below.

The final step is to add a stabilizing closure model, for which three options have been applied.

1. Smagorinsky-Lilly

$$res\big(t, (\mathbf{u}, p), (\mathbf{v}, q)\big) = \int \left(\frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{v} + c(\mathbf{u}, \mathbf{u}, \mathbf{v}) + \nu_f \cdot (\nabla(\mathbf{u}) \odot \nabla(\mathbf{v})) - p \cdot (\nabla \cdot \mathbf{v}) + (\nabla \cdot \mathbf{u}) \cdot q \right) d\Omega_f$$
$$+ \int \big(\nu_{sgs}(\mathbf{u}, C_s, \Delta_{SL}) \cdot (\nabla(\mathbf{u}) \odot \nabla(\mathbf{v})) \big)\, d\Omega_f + \int \big(0.5 \cdot (\mathbf{u} \cdot \mathbf{v}) \cdot (\mathbf{u} \cdot n_{\Gamma_{out}}) \big)\, d\Gamma_{out} \tag{3.13}$$

   where

   - $c(\mathbf{a}, \mathbf{u}, \mathbf{v}) = 0.5 \cdot ((\nabla(\mathbf{u}) \cdot \mathbf{a}) \cdot \mathbf{v} - \mathbf{u} \cdot (\nabla(\mathbf{v}) \cdot \mathbf{a}))$
   - $C_s = 0.11$
   - $S_{ij}(\mathbf{u}) = 0.5 \cdot (\nabla(\mathbf{u}) + \nabla(\mathbf{u}))$
   - $\nu_{sgs}(\mathbf{u}, C_s, \Delta_{SL}) = \big(C_s^2 \cdot \Delta_{SL}\big) \cdot \big(2 \cdot \big(S_{ij}(\mathbf{u}) \odot S_{ij}(\mathbf{u})\big)\big)^{0.5}$
   - $\mathbf{u}, \mathbf{v}, p, q$ are the velocity and its test value, and the pressure and its test value

   From this residual form the solution for the velocities and pressures over the full domain is determined. Here one sees that the last term does indeed cover the boundary definition in the outflow region.

2. WALE
   For the WALE scheme the only change is the definition of Δ_{SL}, since a correction factor is applied:

$$\Delta_{WALE} = \Delta_{SL} \cdot \left(1 - \exp\left(-\left(\frac{y^+}{25} \right)^3 \right) \right) \tag{3.14}$$

   where $y^+ = d \cdot u_\tau / \nu_f$ with $u_\tau = \sqrt{\tau_w / \rho}$, $\tau_w = 0.5 \rho C_f |\mathbf{u}|^2$, $C_f = 0.026 / Re^{1/7}$ and finally $Re = |\mathbf{u} \cdot D / \nu_f$.

3. OSS-ISS
   The weak form reads:

$$res\big(t, (\mathbf{u}, p), (\mathbf{v}, q)\big) = \int \left(\frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{v} + c(\mathbf{u}, \mathbf{u}, \mathbf{v}) + \nu_f \cdot (\nabla(\mathbf{u}) \odot \nabla(\mathbf{v})) - p \cdot (\nabla \cdot \mathbf{v}) + (\nabla \cdot \mathbf{u}) \cdot q \right) d\Omega_f$$
$$+ \int \big(\tau_m \cdot \big((\nabla(\mathbf{u}) \cdot \mathbf{u} - \eta_{nh}) \cdot (\nabla(\mathbf{v}) \cdot \mathbf{u}) \big) + \tau_c \cdot ((\nabla \cdot \mathbf{u}) \cdot (\nabla \cdot \mathbf{v})) \big)\, d\Omega_f + \int \big(0.5 \cdot (\mathbf{u} \cdot \mathbf{v}) \cdot (\mathbf{u} \cdot n_{\Gamma_{out}}) \big)\, d\Gamma_{out} \tag{3.15}$$

   For the exact details and implementation of all new factors a reference is made to Colomés et al. (2016). Note that the definition of $c(\mathbf{u}, \mathbf{u}, \mathbf{v})$ changes slightly compared to the SL and WALE scheme.

Integration scheme
These results should then be integrated to the next time step. This is done through the implicit "generalized alpha" scheme with $\rho_\infty = 0.5$. This numerical time integration method, like Forward Euler or RK 45 exist as well, adds stability through its implicit nature. Additionally the α parameter allows the internal damping of the method to be controlled. This adds accuracy to the integration. It works best for problems which are of second order in time, although adaptations for Navier-Stokes equations (of first order in time) have been widely applied since the early 2000s (Jansen et al., 2000).

Post-processing
The processes above are compiled as a single function. While calculating each time step, the pressure and velocity are stored. For the VMS scheme the intermediate numerical artefacts u_{nh} and η_{nh} which are required for the orthogonal projection scheme are stored too. The pressure and velocity fields are used to determine the drag and lift force on the perforated cylinder through:

$$\vec{F} = -\left(\sum \int \left(\left(n_\Gamma \cdot \sigma_{dev_f}(\epsilon(\mathbf{u}))\right) - p \cdot n_\Gamma\right) \cdot d\Gamma_s\right) \cdot \rho \tag{3.16}$$

The force vector contains then the total drag and lift components of the cylinder. The minus sign is explained through the general use of the fluid domain in this calculation scheme. The integration of pressures and viscosity terms then calculates the force as experienced by the fluid. A minus sign converts this to get the forces as experienced by the structure itself.

3.1.3. FEM execution and cluster parallelization

The amount of computational power required is most likely very high. Therefore efficient coding might not be enough to find useful results within a reasonable time frame. Options for parallelization, are provided via DelftBlue, the TU Delft super computer (DHPC, 2022), and Snellius, the Dutch National super computer (SURF, 2022).
To apply this parallelization there are two main routes. The first one is to parallelize different design cases. This is done on both clusters by uploading different batch jobs for each design case. Secondly, internal parallelization over different domain areas is possible (Colomés, 2023). Initial testing of the named package indicated a 12 core optimum for 3-perforation designs in 2D. However, for the meshes developed for this thesis some designs came back with segmentation faults. The origin of this fault was a discrepancy between the 64-bit based GMSH.jl software, and a 32-bit compilation of Metis in PETSc and Mumps, the parallelization solver. A first option would be to recompile one of the two packages into another bit format. This is technically possible and application procedures exist for OpenFOAM, which could be adapted for the Gridap.jl case. The alternative route chosen here was to run Mumps with the "icntl-7=0" flag, which forces Mumps to use another (correct bit format) solver than Metis.
Concretely all testing, convergence and optimization was done on DelftBlue. The full design runs were performed on Snellius.

3.2. Validation and optimization

In order to optimally use all computational resources, initial testing is required to determine the answers to the following questions:

1. Which is the coarsest mesh that captures the drag and lift on the cylinder "exactly"?
2. Which is the largest time step that captures the drag and lift on the cylinder "exactly"? Does this capture all intended physics?
3. Which is the smallest domain size (wake length, inflow length, simulation time) that captures the drag and lift on the cylinder "exactly"?
4. Which of the 6 initial 2D design parameters (D, n, R_β (using β later on), R_t, α (angle of attack), V_{in}) are irrelevant, or can we model very accurately?

When we are certain that the results are "fixed" we can compare them against other research and practical experiments.

3.2.1. Mesh convergence

GMSH.jl uses the MathEval function to indicate local refinement with defined sizes. The outline of the function to be applied is:

$$\Delta = \begin{cases} \Delta_1 = (1 + r' \cdot f) \cdot \Delta_{min} & \text{, if } x < 0 \text{ and } r > r_0 \\ \Delta_2 = (1 + r' \cdot \frac{f}{R_{domain}}) \cdot \Delta_{min} & \text{, if } x \geq 0 \text{ and } r > r_0 \\ \Delta_3 = (1 - r' \cdot f) \cdot \Delta_{min} & \text{, if } r \leq r_0 \end{cases} \tag{3.17}$$

with:

- Δ_i the local refinement in sector i, as shown in Figure 3.3.
- $r' = \frac{r-r_0}{r_0}$ with $r = \sqrt{(x-x_0)^2 + (y-y_0)^2}$ the distance to the centre of the cylinder with centre point (in Cartesian coordinates) $[x_0, y_0]$. r_0 is the cylinder radius $D/2$. r' is then the relative, normalized distance to the cylinder wall. It takes negative values in region 3, explaining the minus sign in Equation 3.17c, thus still increasing the cell size towards the cylinder centre.
- Δ_{min} the minimum cell size, found at the cylinder walls, with here an absolute maximum of the wall thickness t.
- $f = \frac{\Delta_{max}}{\Delta_{min}}$ the ratio between the maximum and minimum cell size, with the maximum cell size found at the inflow and outflow boundary. With an inflow domain of length L_{in} and a wake length of $R_{domain} \cdot L_{in}$, for the domain highlighted in Subsection 3.2.2, the factor $\frac{f}{R_{domain}}$ makes the boundary cells of equal size. Originally L_{in} will be equal to the cylinder diameter D, but as said it can be increased to values higher than the cylinder diameter if needed for convergence.

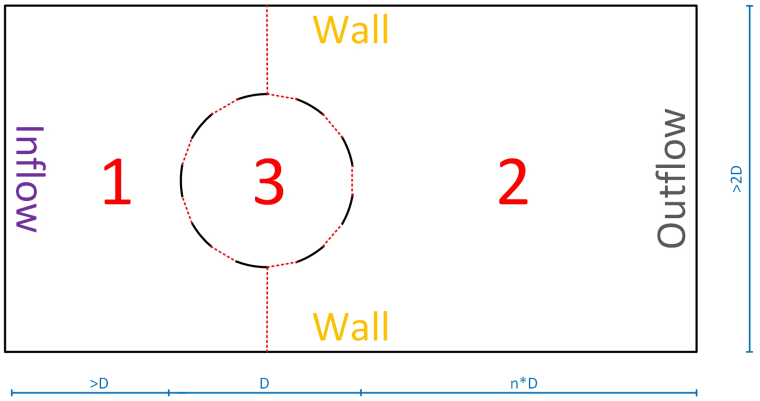


Figure 3.3: Updated domain visualization with indicated meshing regions in 2D

During testing some some minor tweaks were applied, amongst others extra refinement in the centre of the wake region, and a more aggressive cell size increase in the centre of the cylinder. The final applied scheme is modified such that the small, minimum cell size, is kept constant within the cylinder. Additionally there is a refined axis in the centre of the wake, which coarsens as the vertical distance increases:

$$\Delta_2 = \left(1 + r' \cdot \frac{f}{R_{domain}} \cdot \left(\frac{y-y_0}{y'}\right)^2\right) \cdot \Delta_{min} \tag{3.18}$$

with $y' = (y-y_0)/r_0$ being the normalized distance to the wake centre. An example with 3 perforations at $\beta = 0.15$ can be seen in Figure 3.4

Before this mesh was finalized a series of tests was performed with different mesh densities. The mesh above is defined as "Convergence level 4". A mesh of what is here called "level i" has a maximum cell size of 2^{2-i} meter at the inflow and outflow boundaries, with the smallest cell then approximately 5 times smaller at the cylinder wall or in the wake centre. For level 4 this comes to 0.25 m cells at the outer bounds and cells of around 5 cm in the most refined regions. This also means that a mesh of level 3 would have cell sizes double of what level 4 has. Similarly level 5 is twice as dense, also meaning that the computational cost in 2D goes times 4. In Figure 3.5 one can see that the drag trace of level 4 is only partially visible. That is because it lies below the level 5 trace. This is a visual proof of convergence. Although the 5 cm cell size is enough to capture the large scale drag and lift, this will not be adequate to capture flow effects around the perforation edges and along the wall thickness since $t = D/R_t$ is between 6.7 (8/120) and 15 (12/80) cm, then only being covered by 1 to 3 cells.

The previous convergence was ran at time steps of 0.05 seconds. Additional testing to optimize the time step were performed, both to increase it for faster calculations as to decrease it to see of the original 0.05 s assumption was justified. The plot in Figure 3.6 visualizes this. The 0.2 second time step trace (higher ones are ran but not shown) is the first one which does not accurately follow the finer traces. Therefore an 0.1 second time step is seen as the maximum

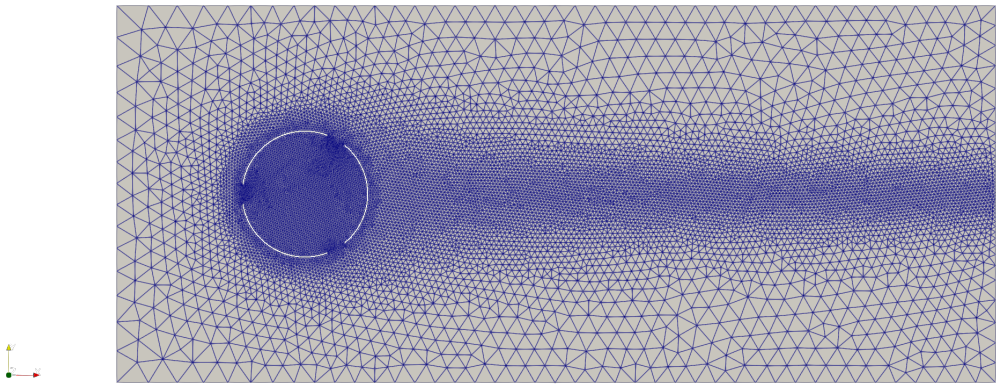


Figure 3.4: Domain mesh visualization in 2D

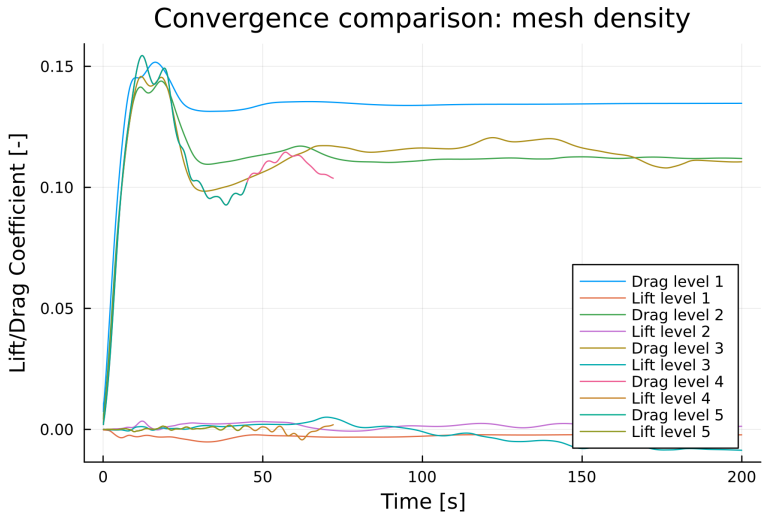
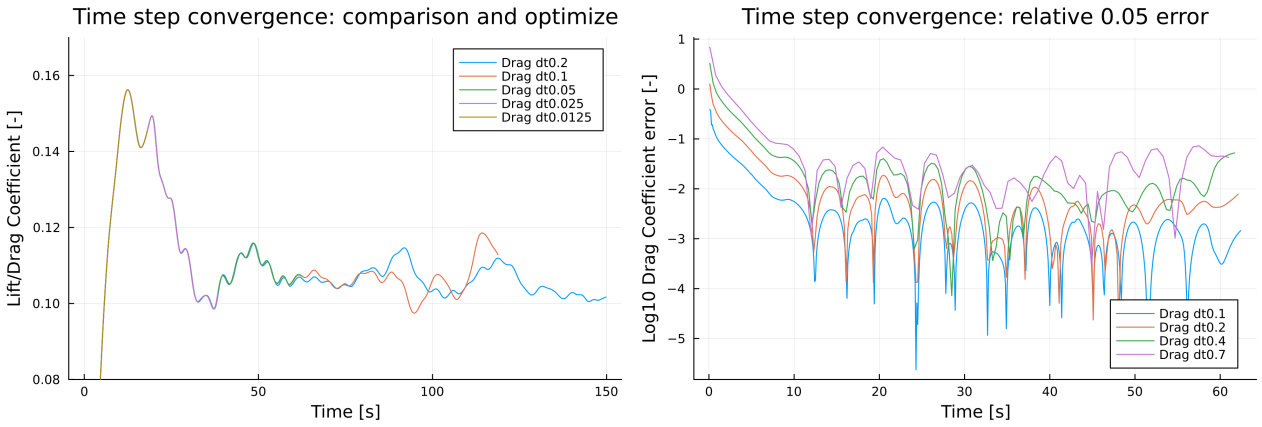


Figure 3.5: Mesh convergence for levels 1-5

allowable step. For larger time steps the relative error exceeds the 1% (10e-2) allowable error defined earlier. This will still allow analysis up to 10 Hz, which is generally more than sufficient to capture all engineering-relevant effects for structural design.



(a) Absolute value

(b) Relative error to smallest step

Figure 3.6: Time step convergence for different time steps

A second time related question is how long the simulations should last. While for higher n the full development of the flow is a bit faster, the most critical simulations are the ones with low number of perforations and low porosity. Their wake field takes upwards of 200 seconds to fully develop. While the first part of their drag-trace follows an expected overshoot-dip-plateau shape (seen in the first 180 seconds of Figure 3.7), after a certain extended period of time the wake vortices do seem to experience a lock-in effect, changing the drag regime. Since this means that, to be safe, all traces should be analysed after 200 seconds, it is chosen to let the simulations run up to 600 seconds. This way 400 high quality seconds can be analysed, allowing conclusions with an accuracy of up to 0.0025 Hz.

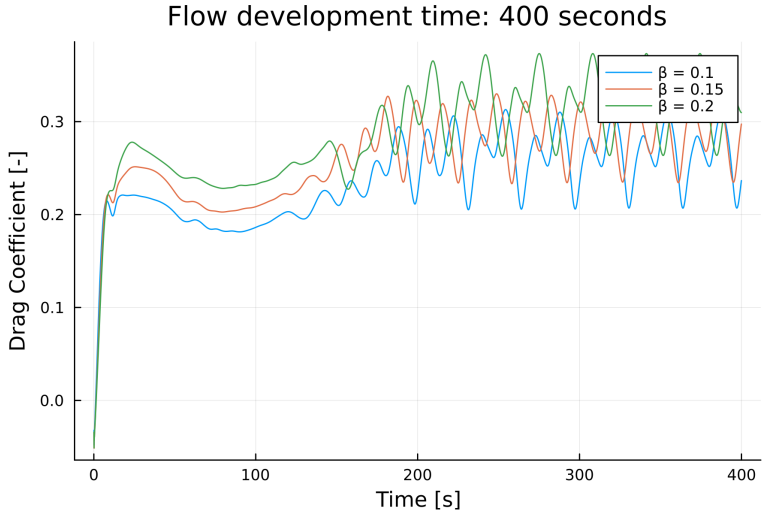


Figure 3.7: Low n and low β flow development time

The found optimal spatial and temporal resolution allow two quick checks on the actual captured energy scales as defined by Kolmogorov, and the ones for which we rely on the sub scale model.
To apply DNS the original theory requires a cell size lower than $\eta = (\nu^3/\epsilon)^{0.25} = 56$ μm for ν = 1e-6 as the dynamic viscosity and $\epsilon = u'^3/D$. In this u' is taken at 2 m/s, as preliminary simulations indicated this to be the maximum expected flow velocity for 1 m/s inflows, which are used later on. This 1 to 2 m/s increase around a cylinder is in exact accordance with the expectations from potential flow theory (Elger et al., 2016). Taking a critical CFL limit of 1 the temporal resolution then becomes 56 µs. For comparison this would make the current model set-up 1.5 billion times more expensive. This indicates that the use of LES was a smart move over DNS.
Secondly the Kolmogorov energy cascade can be approximated by:

$$E(k) = K_0 \epsilon^{\frac{2}{3}} k^{\frac{-5}{3}} \exp\left(-\frac{3K_0}{2}\left(\frac{\nu^3 k^4}{\epsilon}\right)^{\frac{1}{3}}\right) \tag{3.19}$$

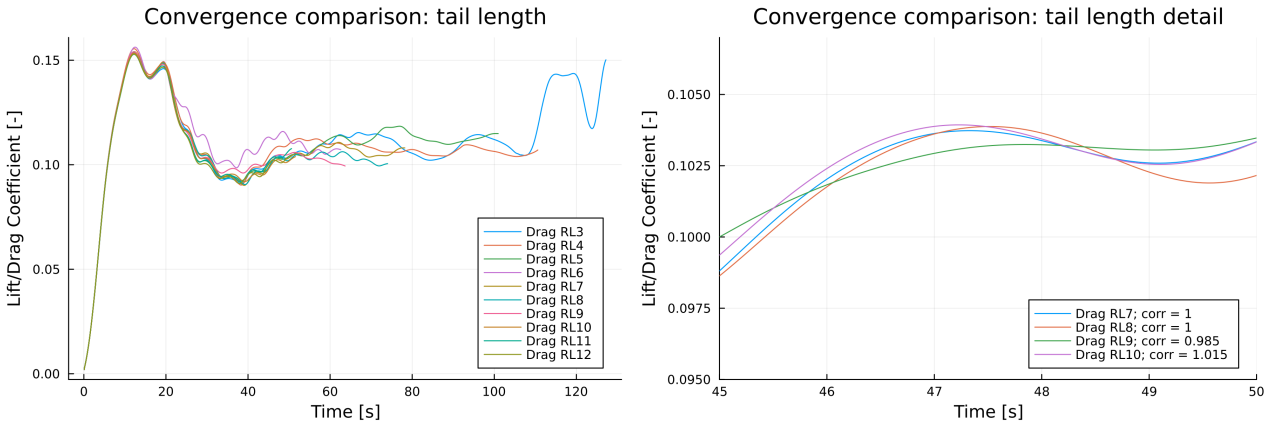
with $K_0 = 1.5$, $k = 2\pi/\Delta = 125$ with Δ the captured eddy size, equal to the cell size of 0.05 m. By integration of the full energy and comparing that to the, for us, cut-off at $k = 125$ it can be shown that only 0.8% of all energy originates from the sub scale VMS model. The mesh choice should therefore be adequate. Even when not taking into account the cells near the boundary itself, where Kolmogorov is technically not applicable, and only looking at cells outside of the boundary layer, one finds most cells with sizes of around 10 cm. Here the boundary layer is defined as $\delta_D = 0.37\frac{x}{Re_x^{1/5}} = 0.15$ m (Schlichting, 1955). Even for this cell size the subscale model only needs to account for 2.1% of the total energy.

3.2.2. Domain optimization and key parameter selection

Wake length

The wake length is one of the main points of discussion in drag flow simulations (Méchartes, 2021; Colagrossi et al., 2019; Guan and Liu, 2018; Deltares, 2018; Hu and Han, 2017; Wellens, 2012). These sources are more unanimous on the optimal width of the domain, which should be at least 3-4D. 4D is taken here to be on the safe side.
In order to keep the computational cost low, the wake should be as short as possible while still capturing an adequate amount of shedded vortices. Numerically this means that the energy dissipation must be large enough so that the outflow condition can be correctly imposed. In Figure 3.8 one can see that, although there are still minor differences, the characteristics of the drag history do not change for domain lengths of 7 diameters or more. With in this case

the cylinder centre placed at 1.5D (D being the cylinder diameter) from the inflow side, this makes the optimal wake length equal to 5D. The right sub-figure shows that the changes after the domain length exceeds 7D are within 2% of each other with no clear trend, indicating that they mainly originate from minor, propagated numerical artefacts.



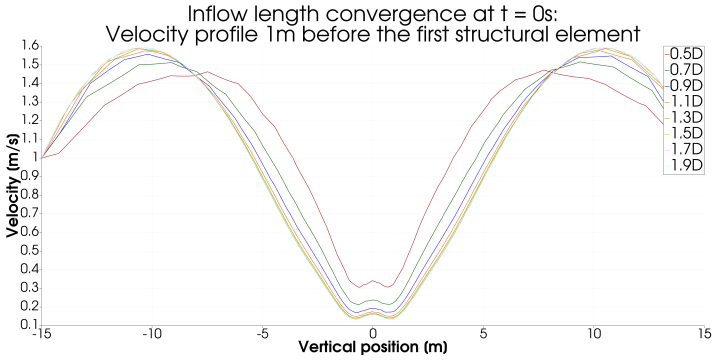
(a) All runs

(b) Detailed converged cases with minor corrections

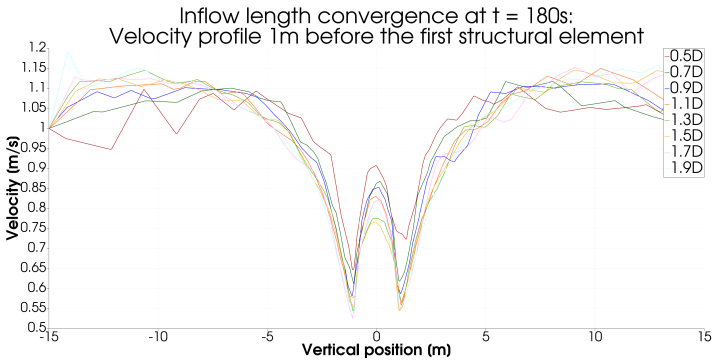
Figure 3.8: Wake length convergence visualization

Inflow length

A similar consideration can be made for the inflow length. However, the verification approach here is slightly different. While the drag traces were very similar, the flow regime of the simulations was not. This means that to verify the optimal inflow region length we should not look at a force output, but rather to the velocity profiles at the inflow boundary. In Figure 3.9 one sees the cross-flow velocity profile a few cm before the cylinder. One sees that the velocity profile only changes marginally when the inflow length, that is the distance from the inflow boundary to the first encountered part of the cylinder, exceeds 1.3D, and definitely after 1.5D. Given that the original wake-length tests were performed on a 1D inflow length, the total required domain length now becomes 7.5D



(a) 0 s



(b) 180 s

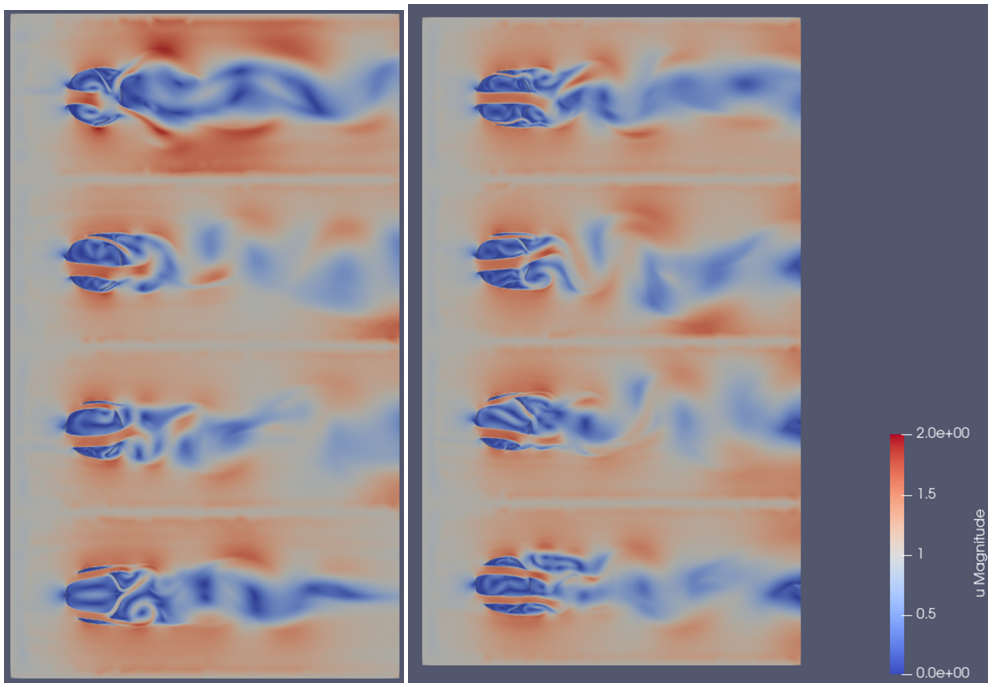
Figure 3.9: Inflow length convergence visualization

Design parameter relevance and prediction

The 6 design parameters and their influence for small changes are researched in form of “box runs”. Rather than analysing all combinations, each time 1 parameter is changed from a central case. The average case has D = 10m, R_t = 100, n = 12, R_β = 0.5, α = 0 and V_{in} = 1. For full reference the results for drag and lift can be seen in Figure D.2. The left column shows the full force traces. The right column shows the (Morison) coefficients for drag and lift.
Here the focus lies on the drag and lift coefficients themselves, disregarding inertia effects. From the Morison coefficient data it can be seen that for high Reynolds simulations drag dominates the force on a cylinder in a steady flow. However, for periodic and wave loading, the flow becomes more inertia dominated. Implementation of periodic flows for wave-simulation may require a focus shift to the inertia constant since the drag-dominated flows would then disappear to make way for a more inertia dominated scheme with different U-D scaling laws. The further validity of Morison coefficient usage should then be rechecked.
From the data the following conclusions can be drawn:

- Drag
  - Diameter: Scales good using Morrison and larger diameters do have a lower influence of high-frequencies (smoother traces). From Figure 2.2 this was theoretically expected. Since the D = 10 monopiles have Reynolds values well above the drag crisis, this drag coefficient should be constant for small changes in D, and at that also for V_{in} .
  - Wall thickness: No clear trend and limited importance, as one would expect given the relatively low number of cells over the steel thickness of the plating. This was predicted earlier as when looking at the mesh already.
  - Perforations: Lots of frequency changes and clear value influence. The flow regime gets more turbulent for higher n, but the variation amplitude reduces due to an averaging-out effect of the smaller vortices over the full structure area.
  - Porosity: Clear influence on the average value, but the exact relation is hard to define.
  - Angle of attack: No trend is visible after “flow development” is complete. The main changes are the flow pattern.
  - Velocity: We need to compare relative time traces, so the faster traces are now "stretched out horizontally". This is because in a 3 m/s flow the flow development after 30 seconds would be similar to one at 90 s for a 1 m/s inflow. We see that Morison works very well for drag force scaling in this case.
- Lift
  - General: Very periodic traces so there can be a general VIV risk. The values themselves are multiple orders of magnitude below the drag force (factor 12). This leads to believe that lift is not our main concern when excluding FSI.
  - Diameter: Large D’s show lower frequencies. This is expected from Strouhal relations. The Morison scaling is similar to the drag case.
  - Wall thickness: No immediate influence from traces. However thinner walls show less variation. Maybe the influence on the "ends of steel" is quite large and one gets locally high velocities. The mesh in these regions however does not allow exact verification, as was stated earlier.
  - Perforations: High n’s give more averaged lift. This is a consequence of the added turbulence. Lift-peak values increase for smaller n due to lower or less random turbulence.
  - Porosity: Higher porosity gives more random traces due to more "space" for eddies to pass through the perforations. Low porosity increases the relevance of lift.
  - Angle of attack: The traces may not be fully developed for the performed simulations. However, it is already clear that non-symmetric samples do increase mean lift values. Rather than the angle it looks to be more related to the different “paths” the fluid follows, especially at low n. A visualization is added in Figure 3.10. The main drag peaks follow from a fluid flow crossing a piece of steel when trying to follow the wake vortices behind the cylinder. Non-zero average lift is generated when a flow path is “permanently forced” to take a non-straight route. These findings can fuel upcoming research.
  - Velocity: Morison scaling still looks adequate when comparing to drag orders of magnitude.

To summarize it appears that the number of perforations n and the porosity β are most important for further drag research. While the diameter D and the flow velocity V_{in} contribute greatly as well, their effects are better captured in existing analytical relations. Lift in general can be disregarded. The angle of attack itself is of low importance. Rather the existence of a logical and symmetric path for the fluid through the structure determines the size of drag and lift variability for angled simulations. Again, the full force traces are added in Appendix D



(a) $n = 5$ (b) $n = 6$

Figure 3.10: Visualization of flow paths through a cylinder for different angles of attack (0, 1/6, 2/6 and 3/6 $\cdot \alpha$)

3.2.3. Validation against literature and design standards

Not only must the results be converged in order to prove a CFD method, but the results must be physically sound as well. A large part of numerical modelling is comparing its results against other forms of experimentation or theory. The complex shapes of perforated monopiles will limit the options for full theoretical validation. However, the initial solver accuracy can be tested on simplified geometries (full cylinders here), while using experimental validation for the full geometry. For the full geometry, validation against other, existing, numerical models would be possible as well. In this case, validation has been done to three different sources. The validation can be visually verified in Figure 3.11.

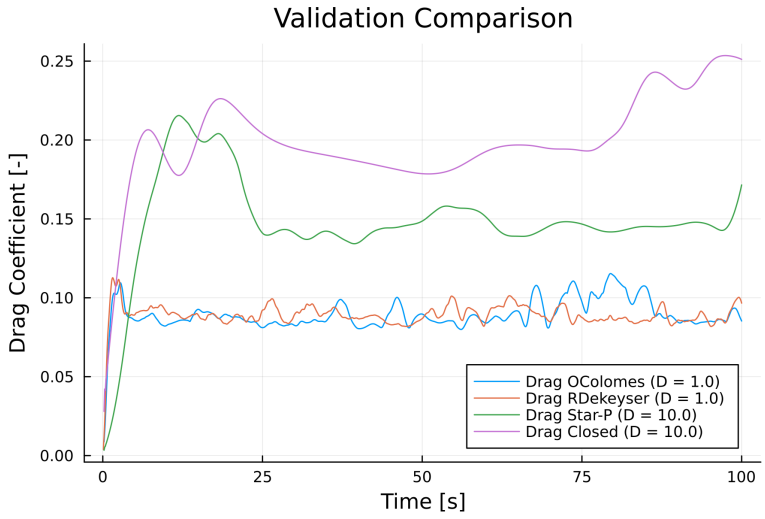


Figure 3.11: Validation of the LES-VMS method against 3 literature sources

Full cylinder Morison

Following the design graphs of Chakrabarti (2005), one would expect the closed cylinder to have a drag coefficient of around 0.2. When looking at the mean of the purple trace, this is indeed the case. Numerically it can be verified that the mean of this drag trace lies at 0.2099 (absolute value from 50-100 seconds).

OC dataset
The numerical example model developed internally at TU Delft by dr. O. Colomés was used as a second verification. The mean drag from his simulations compared to ours, which have a similar calculation scheme but a slightly changed mesh and domain, are 0.092 and 0.090 respectively. Their standard deviations are 0.009 and 0.004. So while the drag mean itself has not changed that much, the fluctuation has gone down considerably. This makes that the traces look very unalike, while for engineering practice they are rather similar. On the value itself one could comment that this reduction of the drag of almost 55% is more than what was proposed in earlier work. There a reduction factor of 20% (M. C. Anderson, 2017) to 40% (van der Ploeg, 2021, Star, 2022) was predicted. However, the latter did mention that added benefits could be found when looking at transient situations.

Qualitative comparison to QStar
Finally we can come up with a numerical comparison to earlier Stokes creeping flow work by Star (2022). While this work cannot provide transient data, his simulations predicted a reduction factor of 17% for the shown case ($n = 12$, $\beta = 0.5$), translating into a drag factor of 0.17. However, this was done through the use of a parabolic inflow profile. By using a parabolic inflow profile ("Star-P" with "P" for parabolic) the LES method applied here showed a mean drag of 0.15. And while the results are technically off by a little over 10%, the rudimentary nature of the creeping Stokes method, strengthened by the prediction of Star that transient solvers may show even better reduction factor results, gives us a third indication that the applied transient scheme here can be considered as validated.

3.2.4. Domain and simulation build-up summary
From the previous paragraphs we can conclude that the correct situation to run many simulations on is:

- A domain of length 7.5D, width 4D, and the cylinder centre at 2D with D the diameter here.
- A time step of 0.1 seconds to amount to a total simulation time of 600 seconds.
- A focus on number of perforations and porosity, since wall thickness and angle of attack appears to be of no influence and diameter and inflow velocity scale very well using known analytical (Morison) relations.
- A separate set of simulations for low-perforation angle-of-attack research to obtain insight in the flow-paths and the influence on the drag, lift, and their variability. This separate aspect is left as a future recommendation.

By applying these parameters we ensure high quality simulations from which practical engineering conclusions for design can be drawn. Additionally the dataset then forms a reliable basis for finding a surrogate model. The surrogate model's relevance is highlighted here using a numerical comparison. Two datasets are mentioned (OC and RD), both consisting of around 500 simulations. Each of these simulations required a (maximum of) 48h of computation time on 12 parallel cores. To circumvent the Snellius restriction of always having to "pay for" 32 cores on each occupied node, it was tried to run the simulations on 10 cores so that 3 simulations could occupy 30 of the 32 allocated cores in a requested node. This however did not work, meaning that some CPU hours were "lost" because of the Snellius 32-core minimum allocation cost. In total the two datasets required over 450.000 CPU hours to run. To ensure that this work must not be redone, surrogate models come into play. This is further explored in Chapter 4.

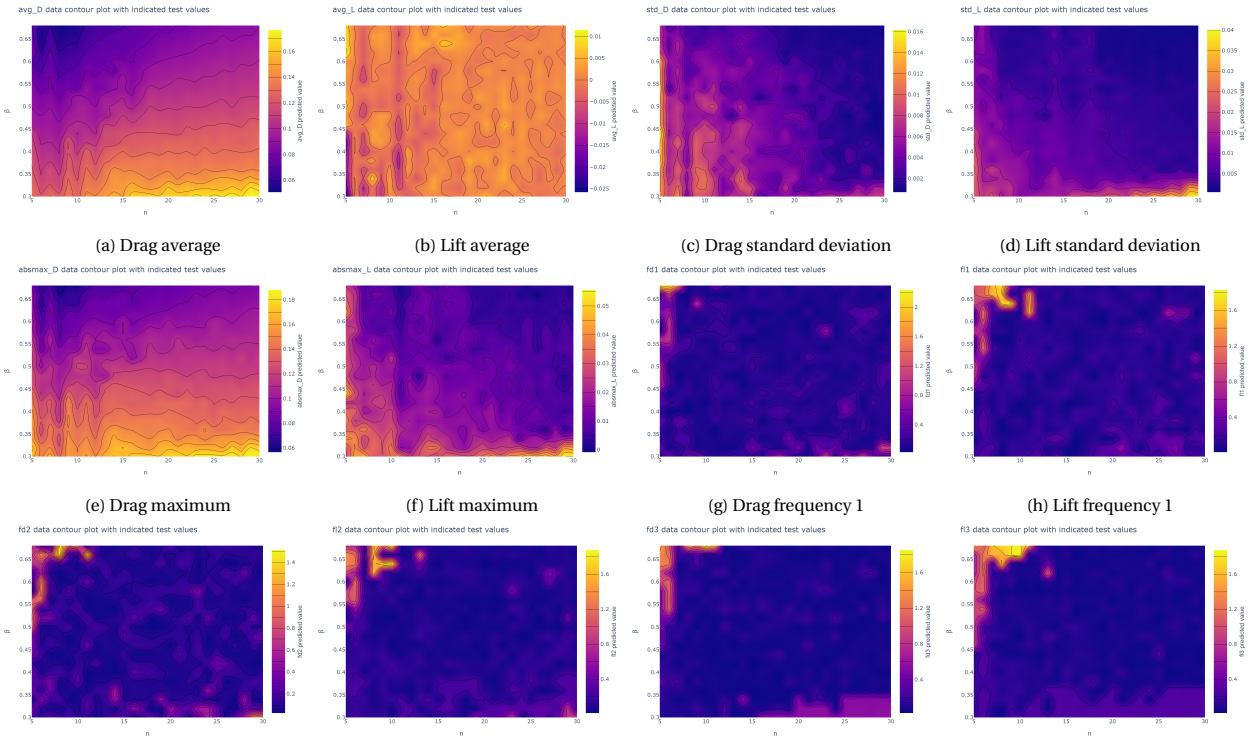
3.3. CFD analysis and dataset comparison
The CFD analysis is based on two different datasets. One was created in January 2023 by dr. Oriol Colomés, the other one was created specifically for this thesis. Firstly the two datasets will be considered separately, after which their combined conclusions and/or differences are discussed.

3.3.1. OC data: square cutouts - diameter normalized
This analysis is a summary of a short update-report written for the 2023 SIAM convention in The Netherlands. The full report can be seen in Appendix B.

The generated data consists of 500 samples of varying numbers of perforations n (5 to 30) and porosity values β (0.3 to 0.7). The output files contain time traces for the pressure integrated drag and lift force on the cylinder. 12 simulations failed to converge, meaning that there were no outputs. 1 simulation can be considered an extreme outlier (order of 6 magnitude difference) and is therefore discarded. This results in 487 time traces.
A Fast Fourier Transform (FFT) on the time traces, excluding start-up and shutdown effects (using 80-180 of the 200 provided seconds) provides the most interesting basis for data analysis. The average value, standard deviation, statistical absolute maximum (99% double sided quantile) and the three most dominant frequencies are extracted for both the drag and the lift. These 12 outputs are to be analysed. The results are shown in Figure 3.12.
Initial data analysis has revealed the following key findings:

- For drag the average value and statistical maximum follow expectations within a narrow noise band. Lower porosity results in higher loads. Higher numbers of perforations give a more random nature to the flow, resulting in slightly lower drag numbers but also much lower variation amplitudes (standard deviations). This is because the small turbulence eddies are averaged out over the whole cylinder, where small numbers of perforations or low porosity values give larger eddies which cannot be averaged out.
- Lift for a symmetrical profile should average out at 0. This is the case, especially for high n. Lower numbers of perforations generate larger eddies, giving more variation in the lift over time. This is analogous to the averaging-out property in the drag discussion.
- Both drag and lift frequencies follow similar trends. Low n and high β result in some higher frequency forcing. The relatively large eddies in these cases carry enough energy to influence the full monopile. As n increases the lift and drag traces become more and more constant. This gives a large decrease of the associated forcing frequencies. Generally the frequency data has a much higher variance than the averaged forces.
- The energy spectra quickly die after 2-3 Hz.

It is promising to see the (absolute) drag coefficient values between 0.06 and 0.17, since full cylinders would theoretically give values of around 0.2. So to conclude the data shows expected values and trends. The trends can generally be explained through flow field patterns and turbulence of the water itself. An FFT extracts the most important parameters. The main focus lies on the average and maximum drag.



(a) Drag average (b) Lift average (c) Drag standard deviation (d) Lift standard deviation

(e) Drag maximum (f) Lift maximum (g) Drag frequency 1 (h) Lift frequency 1

(i) Drag frequency 2 (j) Lift frequency 2 (k) Drag frequency 3 (l) Lift frequency 3

Figure 3.12: Drag and lift engineering mean values: contour plot visualization

3.3.2. RD data: triangular cutouts - full size diameters

In total 448 simulations were planned with a simulation time of up to 600 seconds. These ranged from perforations 3 to 30 and porosity 0.1 to 0.85 with 0.05 intervals. Initial visual verification of $\beta = 0.9$ showed that these meshes were unstable and, with that, structurally not relevant. Therefore these 0.9-porosity meshes were omitted. The Stokes preconditioner failed to converge for almost all meshes with n of 16 and above. From the performed simulations not all ran for the full 600 seconds. The most critical one was cut off at 441 seconds, still allowing a 0.004 Hz resolution from the assumed 200 second steady state region boundary. The completed simulations are shown in Figure 3.13. The analysis and machine learning will be trimmed down to $n \leq 15$. And although the 0.1 s sampling rate would allow analysis of up to 10 Hz, the dataset is limited to 1 Hz for the actual analysis as even the highest-frequency simulation only has 0.1% of its energy contained in higher frequencies. It was originally hypothesized that these large monopiles

will not be too sensitive to large frequencies, to which this conclusion can serve as validation. Additionally the OC, smaller diameter, dataset required analysis of up to 3 Hz. So again the fact that for larger diameters a 1 Hz limit is sufficient is in accordance with Strouhal's theory.

Figure 3.13: RD dataset completed runs

While the OC dataset, ran on 1 m diameter cylinders, showed improvement of the mean drag values for all designs, this is not the case for the RD 10 m diameter simulations, as shown in Figure 3.14. The other mean quantities are added in Appendix D. There is a clear trend where increased porosities give smaller drag values. This makes sense, since in general there is just less material to generate this drag. Especially for higher numbers of perforations ($n \geq 8$) this relationship looks to be not very dependent on n itself. One also sees that for $\beta \leq 0.3$ the mean drag is generally higher than 0.2, which one would expect for a non perforated cylinder. It seems that the added turbulence from the perforations increases the drag rather than reducing it due to the reduction of material in the water flow path. On the other side very high porosities have almost 0 drag means. From a structural point of view however these don't seem too realistic for further application.

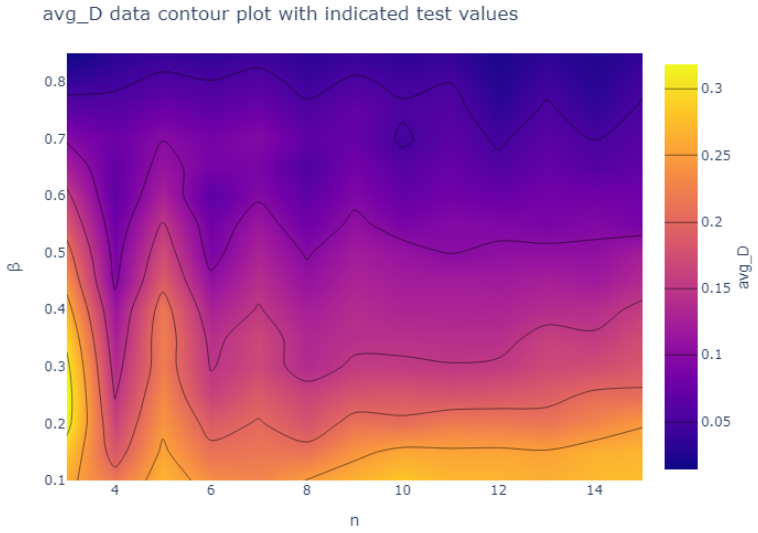
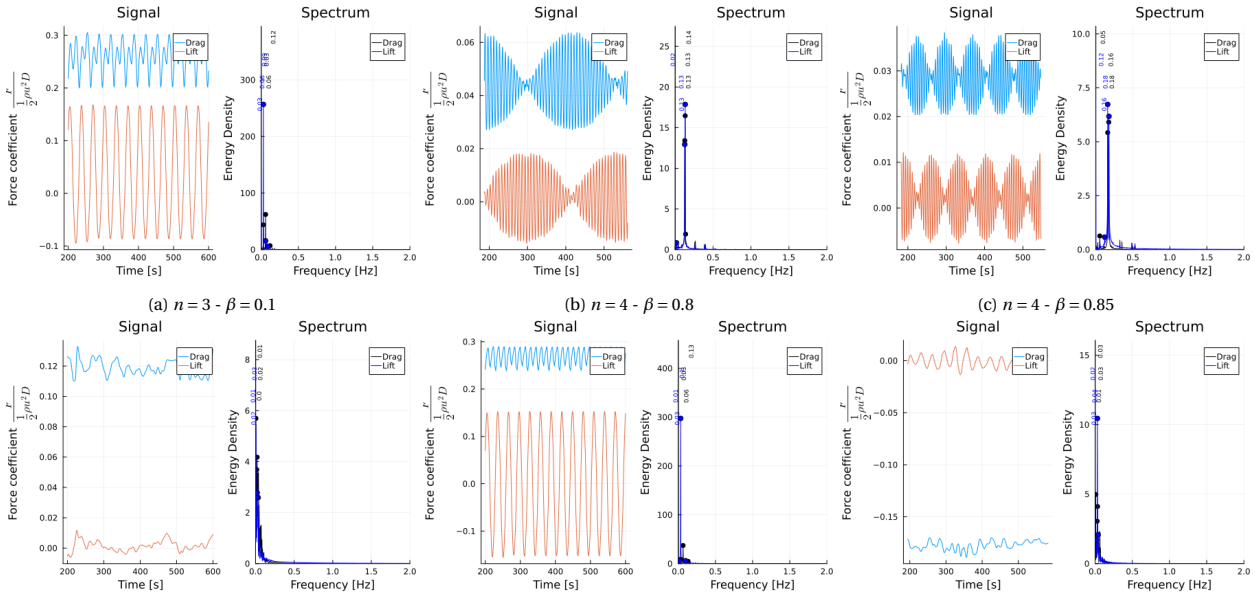


Figure 3.14: RD average drag values contour plot

When looking at the frequencies present in the simulations, the results are much more steady and relatable than the ones found from the small diameter OC dataset. Some key designs are shown in Figure 3.15. The first point to note is that, for low n, all low β simulations are double periodic. This makes sense since their flow patterns would be most closely linked to a closed cylinder simulation, with an additional "simple" flow path through the cylinder generating just one or two additional frequency effects.
A second point to note is that these same low-n designs do get more random traces in mid-β regions, but become periodic again for very high β. An interesting play of frequencies arises for $\beta \geq 0.7$, both for even and odd numbers of perforations, with a beat-phenomenon result. While the exact physical origin of this pattern is unknown, it could prove very important in future determination of statistical maxima or for the determination of resonance effects. On the other hand, the low steel area and following structural weakness in these designs leads to believe that there is only a slim change that they will be actually applied offshore.
Additional results from data analysis give the same conclusions as for the OC data. Higher n increases the amount of variability in the signal but reduces the actual variational amplitude due to the reduction of vortex sizes, then averaging out pressure fluctuations over the whole monopile. These results are now more easily distinguished because of the higher diameter used, making the averaging-out effect more pronounced. The same reasoning also explains why the low-β samples have much higher standard deviations as well, making their prediction using DDM at later stages, most likely, more difficult.
Lastly, due to the smoother signals, it was easier to distinguish some lift relations as well, showing that, as predicted, low n designs were prone to heaving non-zero mean lift values (Star, 2022). Additionally, from these new simulations it becomes clear that the non-zero lift is stronger for low-β designs than for the larger porosities. This can be explained using the same more-or-less-random water flow explanation as given earlier, averaging out the lift signal. Additionally, most designs do show large lift amplitudes for low porosities, regardless of the number of perforations.



(a) $n = 3$ - $\beta = 0.1$ (b) $n = 4$ - $\beta = 0.8$ (c) $n = 4$ - $\beta = 0.85$

(d) $n = 12$ - $\beta = 0.45$ (e) $n = 15$ - $\beta = 0.15$ (f) $n = 23$ - $\beta = 0.35$

Figure 3.15: Drag and lift traces and FFT conversion for some core RD simulations

3.3.3. OC-RD data comparison and design take-aways

While there are some interesting effects at the extremes of the n-β design space, the centre cases of the simulations show general reductions of the mean drag coefficient of around 50%. While additional gains are possible by increasing the porosity, the structural validity of these designs reduces. In case that the stiffness can be assured, one must beware of beat-frequency effects, especially at low-n designs. These effects only become clear for very high porosities and at real life (non scaled) diameters. Low porosities may increase the mean drag, but do reduce the periodic nature of the forcing by adding some new frequencies into the mix. Earlier this effect was predicted by making a comparison to the function of helical strakes. For the design-relevant center-cases generally higher porosities give a lower mean drag, and higher numbers of perforations make the signal more constant due to averaging-out of the vortex effects.

4

Surrogate modelling of fluid flow

In this chapter the data generated by the simulations is applied into different sets of machine learning models. These models, given their training on the dataset, will then be able to predict intermediate relations without having to re-run the expensive simulations. Section 4.1 further explores why one would want to have a regression model for design purposes. Secondly the implementation of different models is discussed in Section 4.2 with the results presented in Section 4.3 for the two datasets.

4.1. Surrogate models and their application

As indicated in Chapter 3 the calculation times for transient, non-linear models can ramp up very quickly. Surrogate modelling in essence tries not to speed up the computations of a numerical model, but rather looks for alternative, easier models to obtain the same results. For CFD models there are two main ways of finding simulation results without using high amounts of computational power. Although the first may not be seen as a full surrogate model, it is still added for completeness.

4.1.1. Physical experiments

One of the main ways CFD calculations are omitted is through the use of scaled tests and physical experiments. Even for perforated monopiles the initial concept was physically tested by M. C. Anderson (2017) before being picked up by CFD researchers.

Physical experiments have three main drawbacks (Wellens, 2012). The first one is the physical cost of modelling, especially at full scale. Building different scale models for different designs can become expensive and time consuming. A second drawback is the limited amount of results that can be obtained from an experiment. Due to the physical size of sensors, measurement errors and inaccuracies. Thirdly and lastly a major drawback of physical testing is the uncertainty in scaling laws. This creates very special, sometimes impossible, validation demands.

4.1.2. Data driven models and Machine Learning

As an alternative to physical experimentation, the results of earlier models, whether physical or numerical, can be used to find patterns. This is called a data driven model (DDM), and is often linked to the more known concept of "machine learning" (ML). As the name suggests DDM's require large amounts of input data. This input data needs to be tailored to the ML-model specifically and vice versa. There are two main model and thus input types: numerical values or visual values. In the first case a limited set of input values generates a set of output values. On the other hand when inputting an image-like array the 2D data is used to predict the output values. This is most often done through the use of a CNN (Convolutional Neural Network). In essence the convolutions just change the image array into a comprehensible set of input parameters.

Training of a regression model is mostly done on an 80% subset of the full data, with the leftover "unseen" 20% left to test the model quality (Morozova et al., 2022). This is called the train-test split, and the 80-20 ratio is used in this thesis as well. Additionally, data often needs to be normalized to limit the effect of outliers. More specifically for CFD problems surrogate models need to be able to generate multiple outputs. This makes a lot of regression algorithms unsuitable.

The most known form of this is linear regression, where one tries to find the best “average” line in a 2D data cloud. While there is, at least for least-squares errors in a linear model, an analytical best-fit, this is often not the case for non-linear or multi-dimensional data. For this different regression algorithms have been made, which all generally rely on iteration and smart parameter-optimization to converge towards a statistical best-fit. Since the iterations update the parameter, it is as if the computer learns the best parameters, giving name to the concept of “Machine Learning”. In this case, the following algorithms are looked into:

- Gradient Boosting (GBR)
  GBR relies on tabular datasets as its input. It uses averages and patterns within the residuals in order to update an average-fit-line (Morozova et al., 2022). It has been stated as the best model for CFD by Morozova et al., due to its tabular-to-output architecture. For multiple outputs however, the model must be trained for each output separately, increasing calculation times and eliminating output relations. Additionally the algorithm becomes very slow when inputs are correlated (Kukacka, 2019). Therefore no implementation is done.

- Response surface models (RSM)
  While often used in engineering and design, RSM is mainly intended towards design optimization. This makes it less suitable for regression. Therefore no implementation is done here, although it may prove useful in later, more detailed, design stages for perforated monopiles.

- Space mapping (SMp)
  Space mapping is mainly a technique to get fine results from coarse data. Only very limited possibilities for extrapolation exist. This may be one of the reasons why there are no CFD precedents to be found, at this time, in literature. Therefore no implementation is done. A possible implementation for the earlier mentioned 2D-to-3D scaling between course and fine simulations could be worth looking into.

- Support vector regression (SVR)
  SVR uses a technique of hyper-plane fitting, being in essence the non-linear and multi-dimensional brother of linear regression. It relies heavily on the correct kernel choice, making it hard to implement for unknown relations. Therefore no implementation is done.

- Nearest Neigbours (KNN)
  KNN pools the most-similar results into an interpolated answer. While being powerful, simple, and not requiring any actual training, the prediction itself is quite slow and very poor on high-dimensional data sets (Ronaghan, 2018). Therefore no implementation is done.

- Random forests (RF)
  RF regression is an ensemble learning technique based on the use of multiple randomly combined decision trees (Amit and Geman, 1997). Its optimization and input parameters require some knowledge of the problem at hand, but when found the scheme can find highly nonlinear relations very efficiently. The scheme is implemented on the data.

- Gaussian Processes (GPR)
  GPR is a non-parametric ML model tailored towards space and/or time series (Powell, 2021). This ensures a wide application on different problems. While the technique relies on the choice of a fitting kernel for its stochastic calculations, it is not extremely sensitive to the choice, especially when using RBF (Radial Basis Function) kernels. The wide application comes at a cost for both accuracy and speed, especially in very high-dimensional problems. GPR has been successfully applied to CFD problems using RANS (Shen et al., 2022), where it proved to have a non-transient speedup by a factor 10. The scheme is implemented on the data.

- Neural networks (NN)
  NN are generally very flexible and have, because of this, gotten a lot of attention and fast development in the past few years. Convolutions, as used by Star (2022) are omitted here since the convolutions focus on finding relations or parameters in the input data from an image-like array. However, in this case the monopile is parametrically defined, so the convolutions would only try to predict these already known parameters. The focus can thus lie on the Neural Network itself. The NN consists of fully or partially connected neuron layers and uses a backwards scheme to update the model regression weights. These weights determine the strength of the signal through an activation function, most often ReLU, that reaches the neurons. The neurons then combine the input signals for the different input values into a single or multi dimensional output. Learning takes place in a back-propagation phase which updates the weights iteratively to arrive at the best fitting output (Bento, 2021).

In offshore applications, machine learning has been applied on multiple occasions in order to speed up calculations and prediction times, both for research and in industry practices. Earlier GPR work has provided RANS design simulation speedups of factor 10 (Shen et al., 2022). Similarly design optimization of wind turbine blades has been done using SVR (Kaya, 2019). However, no incompressible projects have been found for SVR application. The application of MLP's in CFD is limited due to the, while still nonlinear, very rudimentary predictions that follow from a single neuron layer. Neural networks are more popular, for example predicting engineering results like airfoil lift coefficients (Y. Zhang et al., 2018), stationary flow patterns (Eichinger et al., 2021; Jiang et al., 2019) and other offshore applications such as sailboat performance prediction (Byrne et al., 2022). Alternatively NN's have been successfully applied for LES result prediction Beck et al., 2019), Spallart-Alamas closure (B. Liu et al., 2020), or even more fundamentally focusing on more internal concepts such as Reynolds stress tensor prediction (Morozova et al., 2022; B. W. Anderson and Domaradzki, 2012) or LES shape filter optimisation for Smagorinsky-Lilly models (De Stefano and Vasilyev, 2004).
The wide array of previous applications hints that for perforated monopiles too, machine learning models can contribute to the prediction quality, either in terms of accuracy or speed.

4.2. Implementation of ML in Julia: OC data

For this research two different paths are explored for the application of machine learning on the design of perforated monopiles. Firstly a trend-and-averages scheme is developed, from now on called the "Engineering-mean models". Here the focus lies on mean values, maxima percentiles, and predicting a few key frequencies. These models are best suited for preliminary design and trend analysis. Secondly some models are explored which look into the detailed frequency-domain prediction. From their outputs it is not a relative value, but rather the expected energy at each frequency that can be predicted. These models find their application in more detailed design phases where the exact energy-distribution of a single design is required for, for example, resonance avoidance.

4.2.1. OC Engineering mean models

Data preparation

From the force-time traces (both lift and drag) the average value, standard deviation, statistical absolute maximum (99% double sided quantile) and the three most dominant frequencies are extracted. Only the developed values between 80 and 180 seconds of the 200 second simulation time are used. For the frequency extraction a fast Fourier Transform (FFT) is performed, only looking at the positive frequency domain. Each of these outputs, 12 in total, can now be predicted based on two input parameters, namely the number of perforations n and the porosity β.

Model definition and optimization

Prediction is done based on three different models. The hyper-parameters were determined using a random 80-20 train-test split. No random-batch sampling was used. Both input and output data were normalized between 0 and 1 in their own dimension. The three model architectures are:

- A Decision Tree Regressor (ScikitLearn.jl): optimal depth of 9 branches. Originally a Random Forest was used, but hyper-parameter optimization showed that a 1-tree forest performed optimally, therefore the predictor was slightly changed, even though this makes predictions more prone to overfitting. At this point in time no specific reason has been found as to why the 1-tree solution worked best.
- A Gaussian Process Regressor (GaussianProcesses.jl): internal optimization using Optim.jl, using an initial zero mean and a kernel sum and double kernel with a Matern-5/2 and Squared Exponential kernel (all length scales and standard deviations initialized at 0). The optimizer uses a LogNoise correction of -2.0.
- A Neural Network (Flux.jl): learning rate of 0.025, 100 epochs and 60 hidden nodes in a network using a single, dense, ReLU activated hidden layer and using a MSE loss function. The architecture is based on a Flux.jl tutorial by Komarniczky (2022). Afterwards a grid search was used to find the optimal parameters. Here the parameters are varied between 0.0001 and 0.1 for the learning rate, 10 to 1000 in increasing steps for the maximum number of epochs, and 10 to 200 in steps of 10 for the number of neurons.

Model results

The results of this optimization are shown in Table 4.1. The RMSE is reported as an absolute value based on the normalized data. It is determined by the test set only. The time step Δt is given in milliseconds. The actual tests were done on 1000 predictions in view of increasing the timing accuracy, but the reported times are now scaled back to a single prediction. Lastly the badness is defined here as the product of the prediction time and the RMSE. Higher values of either input make the model worse, and hence increase the badness. While different weights could be given to the two badness elements using different exponents, a simple linear relation was chosen here.

	RF			GPR			NN		
	RMSE	Δt	Bad.	RMSE	Δt	Bad.	RMSE	Δt	Bad.
avg_D	0.008827	19673	174	0.004414	2617	12	0.014345	9136	131
avg_L	0.0684	20434	1398	0.063757	2582	164	0.123585	25899	3201
std_D	0.009934	19807	197	0.014637	10926	160	0.025229	11262	284
std_L	0.026339	17905	472	0.018457	1583	29	0.020262	15667	317
Max_D	0.009546	18970	181	0.010089	3206	32	0.013232	9927	131
Max_L	0.037633	14245	536	0.044118	7282	321	0.04846	17835	864
f1_D	0.480	14809	1708	0.411	2085	857	0.421	13290	5595
f1_L	0.143	13623	1948	0.639	1749	1118	0.580	7357	4267
f2_D	0.405	14961	6059	0.341	2366	806	0.346	8798	3044
f2_L	0.190	19097	3628	0.447	1240	554	0.497	9032	4489
f3_D	0.529	16525	8597	0.216	2461	532	0.384	17769	6823
f3_L	0.457	13065	5971	0.289	1330	384	0.477	6897	3290

Table 4.1: OC dataset: Engineering model results

While mainly optimized for the drag mean and statistical maximum predictions, the ideal machine learning model should be able to capture all 12 outputs as accurately as possible. From the data in Table 4.1 the following interesting conclusions can be extracted:

- The badness between the 12 different output variables is hard to compare because the RMSE is scaled due to the different original values. The large RMSE for the frequency data is a consequence of the huge randomness and small correlation within the data itself.
- The Neural network seems to perform on par with the RF and GPR models in terms of RMSE for all frequency related outputs. The force focused output accuracy differs by a factor 2. This is most likely a consequence of the use or disuse of normalized data to train the models. The non-normalized RF and GPR inputs for the frequency matters all lie between 0 and around 1.7, so normalization is less useful than for the force related quantities between 0.05 and 0.2 approximately. In general these two model types are less sensitive to normalization than NN's, regardless of the input data order of magnitude.
- Visually all fits look equally good. This explains the worse RMSE for almost all frequency based output types, as a relatively random point cloud RMSE is hard to predict from visual inspection. We see that the less correlated original outputs have RMSE values which are much closer to each other still. This is because in this case the fit quality through the uncorrelated data cloud is not very dependent on the model used. All trend seeking models perform equally bad (by principle).

The most important conclusion to be drawn however is that GPR is outperforming its rivals on all force-value fronts. This is opposite of what was earlier said to be expected from the GPR technique (Powell, 2021). Its badness is on average a factor 10 lower than the second best model for that specific output prediction. While having RMSE values similar to RF and NN, the training and prediction times are much better. While this may partially be a consequence of the more elaborate, built in optimization of the hyper-parameters, rather than the manual grid-search selection done for the RF and NN model, the main reason is most likely the straightforward and simple nature of the model. Its robustness has been shown in literature, both in general as for CFD applications (Shen et al., 2022).

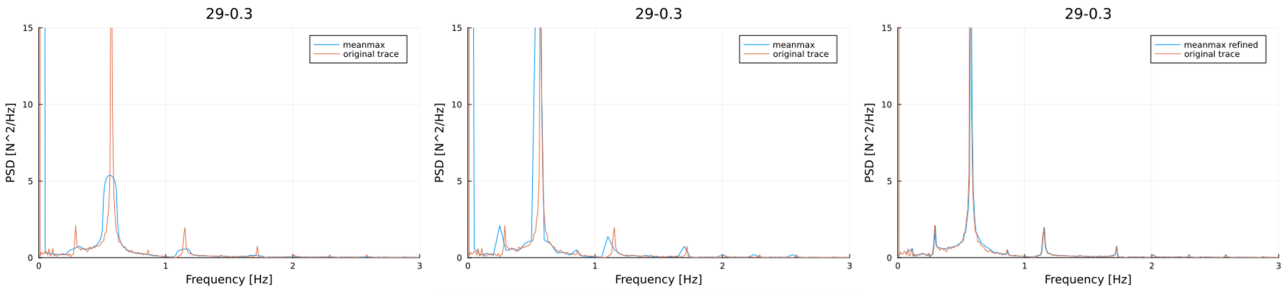
4.2.2. OC Frequency domain prediction models

Data preparation

For the prediction of the full frequency-energy data the data preparation needs a bit more work. The small diameter used in the OC dataset results in very fast flow oscillations, making the data look more random and non-smooth. The same 487 datasets that were used for the engineering-mean models are converted using a FFT. In order to allow a machine learning model to remain simple enough and capture all the trends, while not filtering out the important peaks, two extra steps of data conversion are required.

Firstly we cap the data at 3 Hz. Visual analysis indicated that no interesting or relevant effects take place at higher frequencies. Additionally, their relevance for engineering practice is small, due to the ever lower getting natural frequencies of the large monopile structures, especially if their stiffness were to be reduced even further by perforating them. Also checking the energy which is lost from the FFT by capping it at 3 Hz was below 1%, indicating that no important data is lost through the cut-off.

Secondly, in order to give the models the right focus, three different routes for the data simplification were taken. The three results are shown in Figure 4.1. The goal of the preparation is to simplify the data, either by smoothing it, or by reducing the number of values that the model must learn.
In the first "rolled" scheme, a 10 point rolling average of the data is used. The smooth results would allow for rather simple models to be fit. However, the rolling mean reduces the height of the important energy peaks. This makes its application for resonance questions impossible. Additionally, there is no mathematical way of proving or optimizing the number of frequencies over which the moving mean should be taken. 10 is just an educated guess.
Alternatively the data can be poured into bins, in this case 60 bins of 0.05 Hz. To not over-estimate the total energy, the mean energy in each bin is reported, except for 5 bins where the maximum is reported. The full energy-line is then shifted downward slightly as to ensure the total energy over all frequencies in the original data as well as in the reworked sample is the same. This way the dataset size gets greatly reduced, while still capturing the peaks. As a drawback this does mean that one cannot determine the exact frequency of a peak, with the "shifted peaks" in of Figure 4.1b as a consequence. Also, like for the 10 point rolling mean, there is no way of proving, rather than some manual inspection of the data, that taking 5 peaks into account is the best option.
The third and preferred option is to use a bin-approach but to refine the bins in areas where maxima occur. In this case 30 bins of size 0.1 Hz get refined 10 times to 0.01 Hz (the maximum sample rate) when the maximum energy peak in the bin exceeds 2.5 times the mean energy in the bin, and the peak is higher than 5% of the total energy mean. The latter condition is added to disregard relatively small variation peaks in the almost zero energy tail region of the spectra. As can be seen from the right plot of Figure 4.1, this generally coarse way of depicting the data still provides a closely detailed trace in the areas where it really matters.



(a) Rolled (b) Mean-Max (c) Mean-Max Refined

Figure 4.1: Three different data preparation techniques: qualitative comparison for a representative case

Although the report continues by only discussing the Mean-Max-Refined scheme results, it is worth mentioning that the same model architectures will work, and have been tested to do so, on the other data types, with some minor tweaks to the hyper-parameters in order to find optimal convergence.

Model definition and optimization

As indicated above, there is an engineering discrepancy between redirecting the frequency trace for a single design case, or predicting a trend over multiple designs. In order to incorporate this a total of three model concepts is worked out together with their optimized hyper-parameters for the same three prediction schemes (Random Forest RF, Gaussian Process GPR, Neural Network NN) as before. In Figure 4.2 more details are given on their exact application.

Frequency-Discrete models

Each ML model is trained for each frequency specifically. The inputs are the perforations and porosity. It returns a 2D array with energy values for all these $n-\beta$ combinations at the model's frequency. This way of predicting allows engineers to quickly, in one prediction, see the trends around a frequency that they are interested in, for example which design regions to avoid given a structural eigenfrequency. Some output examples are added in Figure 4.3. To obtain the frequency-energy trace for a specific design one needs to call each respective frequency model, extract the wanted $n-\beta$ combination results, and then concatenate this result over all frequencies. The final ML models were:

- RF: The random forest optimization resulted in 1 tree being optimal, making the model idea interchangeable with a single decision tree. A tree depth higher than 8 did not increase the accuracy using the earlier mentioned 80-20 train-test split. However, since the training and prediction times go up for deeper trees, this single tree depth of 8 was chosen as the optimal architecture. Training times are in the order of 10 seconds for each frequency. The model is based on the code of the ScikitLearn.jl project.

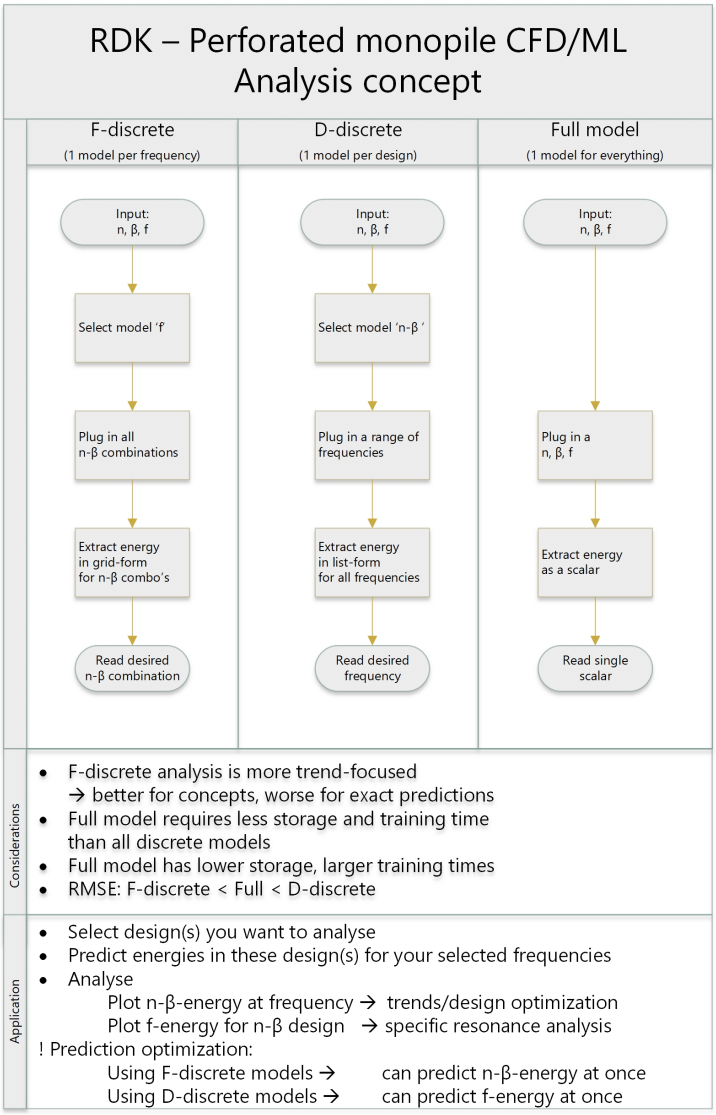
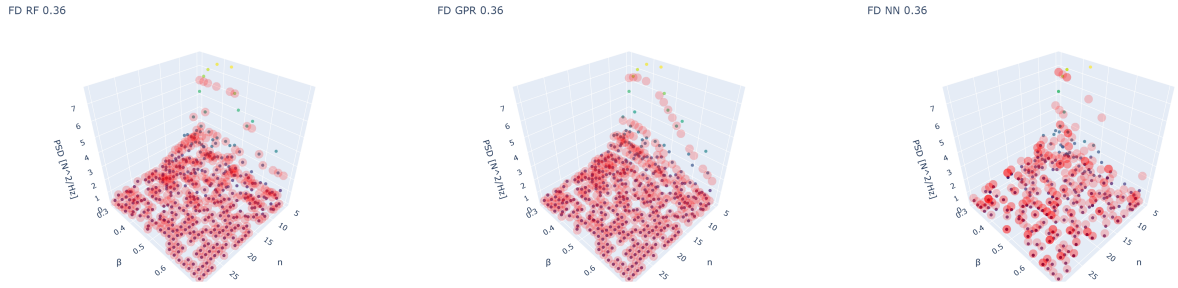


Figure 4.2: Application flow-chart for different ML model types

- GPR: Internal optimization using the Optim.jl package was performed for the hyper parameters themselves. The GPR algorithm was set-up using an initial zero mean and a kernel sum. The first kernel is of the type Matern-5/2 with length scale 0 (for both input dimensions) and standard deviation 0 as well. The second Squared Exponential kernel has length scale and standard deviation 0.1 and 0.0 which showed optimal results. The optimizer uses a LogNoise correction of -1.0. The model is based on the GausianProcesses.jl source code.
- NN: The neural network used here is non normalized and has a fixed depth with 2 dense, hidden, ReLU activated layers. A grid-search for the optimal hidden layer width, batch size, learning rate and number of epochs before cut-off gave 32, 16, 0.0005 and 100 respectively. While a 128 neuron width gave slightly more accurate results (1% relative error reduction), the added expense for predictions showed that this was not necessarily worth the trouble. Not through a grid search but through design the most efficient loss function was a relative (to the dataset mean value) mean-squared error function together with an Adam weight optimizer, using an epoch-based learning rate reducer of factor 1e-2 per epoch. The model is based on the Flux.jl source code. In the final implementation the use of (Xavier-Glorot) weight initialization was removed because, while helping to reduce the number of epochs required to converge, the final model accuracy was not affected.



(a) RF (b) GPR (c) NN

Figure 4.3: Three different Frequency-Discrete prediction models ($n-\beta$ prediction at 0.36 Hz)

Design-Discrete models
For 1 design ($n-\beta$ combination) it predicts all energy levels between 0 and 3 Hz. This model allows engineers to deeply analyse the details of a specific design's drag energy spectrum without the need of model concatenation.

- RF: Here too a single tree depth of 8 was found optimal for the Decision Tree Regressor. When not applying a train-test split, this model especially was prone to overfitting for depths exceeding 12. The model is based on the code of the ScikitLearn.jl project.
- GPR: The best-fit for the design discrete GPR model was found using the same parameters as for the frequency discrete variant, but with the initial squared exponential length scale set to 0. The model is based on the GausianProcesses.jl source code.
- NN: Using the same Adam and weight-decay optimizer, and using the same MSE/mean loss function, it was found that a 3 hidden layer architecture, also ReLU activated, gave the best results. The optimal hyperparameters found were a hidden layer width of 64-256-1028 for the three layers respectively, a batch size of 64, and a learning rate equal to 1e-5. The MSE loss stabilized after 12.000 epochs, where the model optimization was then cut off. The high width and added complexity follows from the fact that in this case all, over 300, binned frequencies are predicted at once. The model is based on the Flux.jl source code.

Full models
By inputting n, β and f the model returns a single energy value. By predicting different values either for different designs or different frequencies, any of the 2 standard results from the previous concepts can be generated. As a drawback the increased dataset size, since now everything must be analysed at once, requires more complicated architectures and therefore also longer training and prediction times, and less accurate results.

- RF: for the full model a tree depth of 32 was found to give optimal results. The model is based on the code of the ScikitLearn.jl project.
- GPR: The previous Matern-Exponential kernel resulted in infinite estimator weights. Therefore an alternative implementation was built with an RBF kernel with length scale 1. The model is based on the code of the ScikitLearn.jl project.
- NN: Testing to find the optimal architecture parameters showed that the same model set-up as for the design-discrete NN predictor was already quite optimized, assuming the there used base-architecture as a starting point. Further investigations showed no immediate improvement. The model is based on the Flux.jl source code.

Model results

The previous paragraph is summarized in the double-purpose Table 4.2. Firstly an overview of the model hyper parameters is given again. They are accompanied by the model performance results.
The hyperparameter optimization has been done through a grid search. For the RF an increasing number of trees was tested (1-10-50-100-200-500) at different depths (1-50). The GPR model had a trial-and error based set-up. The Optim.jl package performs a Riemann optimization internally on the GPR kernel parameters. The Neural network here was tested for 8, 16, ..., 2048 neurons in each layer, a batch size of 2, 4, ..., 256, and a learning rate of 1e-5, 5e-5, 1e-4, 5e-4 or 1e-3.

		FD	DD	Full
RF	Depth	8	8	32
	RMSE	1.858	3.915	6.058
	Δt	5.351	6.278	15.249
GPR	Kernel	Matern-5/2 (l=0.0 - σ=0) + Squared Exponential (l=0.1 - σ=0)	Matern-5/2 (l=0.0 - σ=0) + Squared Exponential (l=0.0 - σ=0)	RBF (l=1.0)
	Mean	zero	zero	zero
	RMSE	2.673	13.887	4.681
	Δt	4.371	19.526	16.284
NN	Dense hidden layers	2	3	3
	Dense layer width	32-32	64-256-1028	64-256-1028
	Layer activation function	ReLU	ReLU	ReLU
	Batch size	16	64	64
	Learning rate	5e-4	1e-5	1e-5
	Weight decay	0.01	0.01	0.01
	Optimizer	Adam	Adam	Adam
	Loss function	MSE/mean	MSE/mean	MSE/mean
	RMSE	6.283	5.338	27.526
	Δt	8.459	1.863	8.862

Table 4.2: OC data prediction model parameter and performance comparison (RMSE in % and prediction time in milliseconds)

From the table a few different conclusions can be drawn.

- The prediction times are all in the order of a few milliseconds. This indicates that the prediction times are an irrelevant parameter in the model quality comparison. Even the most expensive prediction (Full GPR) can be performed over 60 times in just a single second. These figures are negligible in comparison to the 480 CPU hours a single CFD simulations would take in order to acquire the same data. Star (2022) performed non-transient simulations where the surrogate CNN model was 386 times faster, and he predicted that surrogates for transient simulations could be even faster. For the prediction of 0-3 Hz frequency data every 0.01 Hz (the sampling rate in the performed simulations) the slowest model developed here would take 5 seconds, equalling a speedup factor of 350 thousand. Even though some post-processing of the predictions is still required to recreate the force trace itself (through a reverse FFT, not being able to account for the phase shifts of the harmonic signal), the speed-up is still immense. When comparing to the factor 400 speed-up found in creeping Stokes surrogates (Star, 2022), this value makes sense. In essence the transient model requires 10 time steps per second for 600 seconds, showing a speed-up for a single calculation in time of 58.3. This is then 7 times less than the creeping stokes prediction, as one would have expected for the more complicated LES predictions. Although not exactly similar, the order of magnitude for the prediction speed-up is comparable (7 to 1), possibly indicating that the here developed models are relatively well, or at least similarly well, optimised.

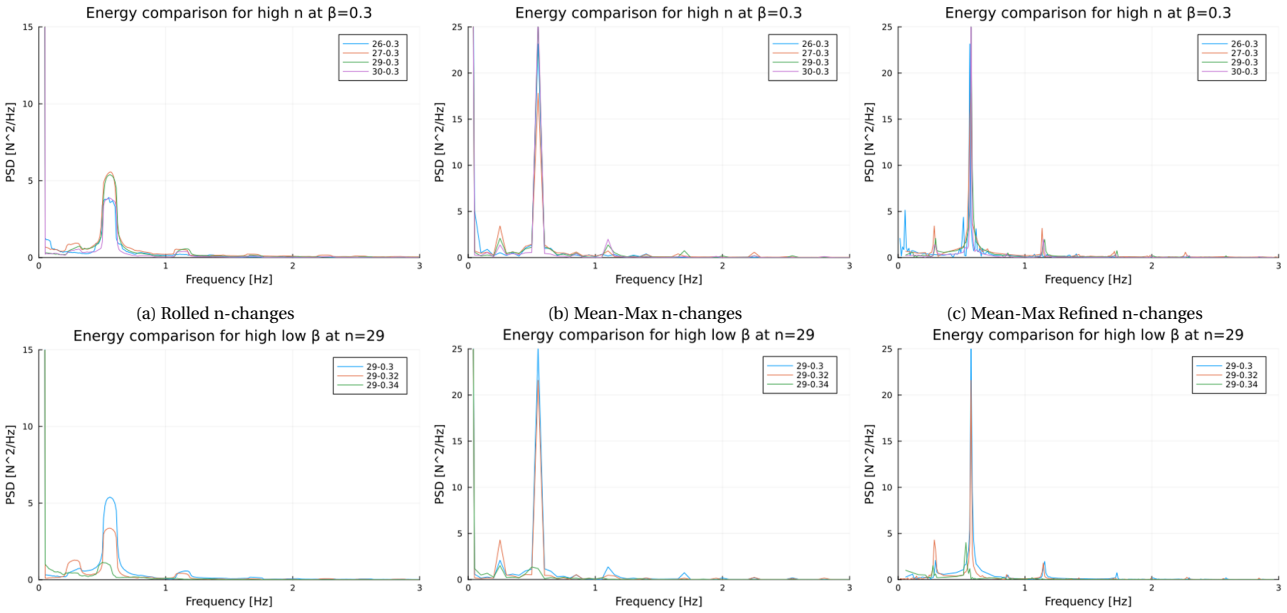
- Most models have a RMSE accuracy of 6% or lower. This can generally be considered acceptable for engineering design, as dynamic safety factors often include a factor two or three as standard. Only the design-discrete GPR and full NN predictions do worse. In both cases this is expected to be caused by sub-optimal kernel and architecture choices. Due to time constraints further improvements were not developed. At the moment the NN options are expected to have the most potential for future improvements, however this is just speculative. The current model with an error of 27%, on average, is unsuited for application in design scenarios.
  As a secondary comparison tool, the original CNN model developed by Star had a similar error rate of 5%.

- While the earlier engineering-mean models showed the best results using GPR predictors, in this case the Random Forest performs slightly better. The Neural Networks are generally worse in both prediction philosophies. This result is attributed to the large randomness which can be found in the provided data. The RF and GPR architectures are more prone to overfitting, making their results on random data look better. In practice however their applicability may not reflect this. As indicated just before, it is expected that the better controlability of neural networks, given enough time and energy to develop them, will allow them to catch up performance-wise and to be the model of choice in the future.

4.2.3. OC Analysis insights and design take-aways

While the raw data trends are already commented on in Chapter 3, the further data handling showed some additional useful insights. Because of the rolling means and/or binning techniques applied, it became possible to show some additional relations in the seemingly random data.

The first conclusion is visualized in Figure 4.4. For a high number of perforations and a low porosity, the energy peak at 0.65 Hz reduces with n, and gets wider with n. The relative effect of this does not show this trend in the rolled data conversion, but it does in a binned mean-max (refined) scheme. Additionally, one sees that all these 4 perforation-samples have energy peaks at the (approximately) same frequency, but that the design with the highest peak is different for each of these frequencies. Their respective relations disappear quickly as the porosity β increases.



(a) Rolled n-changes (b) Mean-Max n-changes (c) Mean-Max Refined n-changes

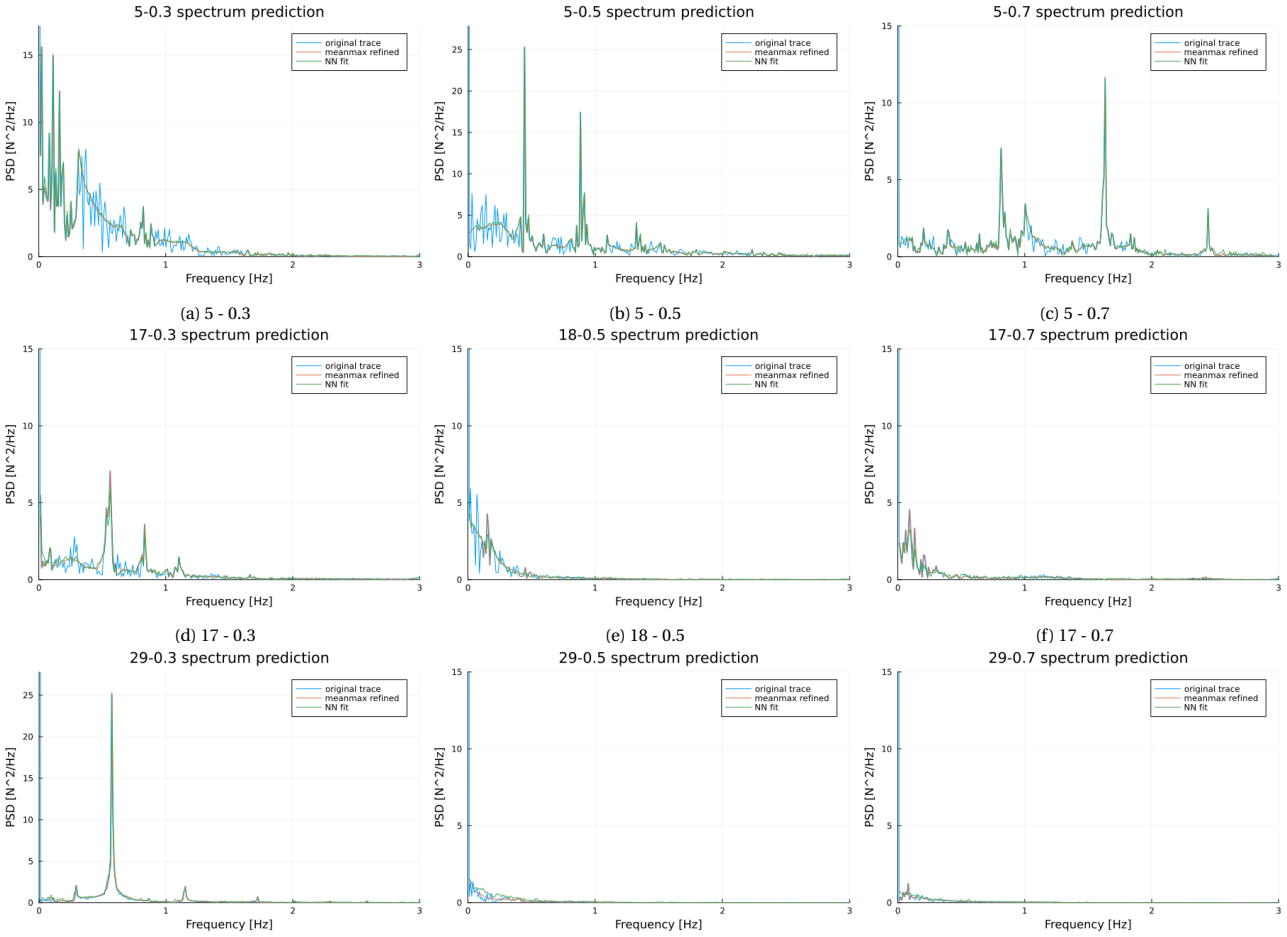
(d) Rolled β-changes (e) Mean-Max β-changes (f) Mean-Max Refined β-changes

Figure 4.4: Small design variation energy patterns for high n - low β designs

An additional point to note is visualized for the RMSE-based least optimal model (Full-NN) in Figure 4.5. More representations can be found in Appendix C. While the RMSE showed a suboptimal relation the visual fit can still be considered quite good. As a side note it is mentioned that simulation case 17-0.5 failed to converge, hence 18-0.5 is chosen as its replacement for imaging purposes.

As the number of perforation increases, the general average energy decreases. This is in accordance with earlier observations in the CFD phase, where the high perforation samples were said to have a much more constant, less varying time-domain drag force trace due to the reduced scale of the created vortices, averaging out over the full cylinder size. Secondly there are some samples which show a clear, exponentially decaying trend, while others have clear resonance peaks. This large discrepancy in data shape can serve as a verification of the reasoning why the much less over-fitted neural network did not give optimal RMSE results, since the characteristics over the data change so drastically. Some samples however do show clear energy peaks. This indicates the importance of looking at the full frequency domain in design phases in the future as to avoid issues with resonance and VIV.

The final important note considers the high energy peaks at the 0 Hz line. In these regions all prediction algorithms failed to give accurate results. This means that the “frequency domain models” are not suited to predict the average drag force on the cylinder. They should be used to predict variations and frequency-related effects. For the constant level the earlier “engineering-mean models” are much better suited. While not explored here, due to the earlier explained lower relevance, this may mean that this problem does not exist for, on average zero-mean, lift predictions.



(a) 5 - 0.3 (b) 5 - 0.5 (c) 5 - 0.7

(d) 17 - 0.3 (e) 18 - 0.5 (f) 17 - 0.7

(g) 29 - 0.3 (h) 29 - 0.5 (i) 29 - 0.7

Figure 4.5: Different selected fitment visualizations for the Full-NN model (OC data)

4.3. Implementation of ML in Julia: RD data

The same paths and models are now applied to the second dataset. Firstly a trend-and-averages scheme is developed. The architecture and parameters are kept as close as possible to the previous models. The same goes for the detailed frequency-domain prediction.

4.3.1. RD Engineering mean models

Data preparation

The same 6 values (mean, standard deviation 99% statistical maximum, and the 3 top frequencies) are gathered from 200 to 600 seconds, or to the end of the simulation, the shortest one being 441 seconds. The same n-β combination idea is used for the prediction of the 12 (drag and lift) values.

Model definition and optimization

Prediction is done based on the same models as before with an 80-20 train-test split as well. No random-batch sampling was used. Both input and output data was normalized between 0 and 1 in their own dimension. For the different hyperparameter options, the same grid-search was performed as the one for the OC dataset. The three final model architectures are:

- A Decision Tree Regressor (ScikitLearn.jl): optimal depth of 9 branches. Here the depth optimization is beatifully clear when looking at its evolution over the branch depth, as shown in Figure 4.6
  This is the same as the OC dataset.
- A Gaussian Process Regressor (GaussianProcesses.jl): internal optimization using Optim.jl, using an initial zero mean and a kernel sum and double kernel with a Matern-5/2 and Squared Exponential kernel (all length scales and standard deviations initialized at 0). The optimizer uses a LogNoise correction of -2.0.
  This is the same as the OC dataset.

- A Neural Network (Flux.jl): learning rate of 0.1, 100 epochs and 160 hidden nodes in a network using a single, dense, ReLU activated hidden layer and using a MSE loss function.
  This is different than the OC dataset.

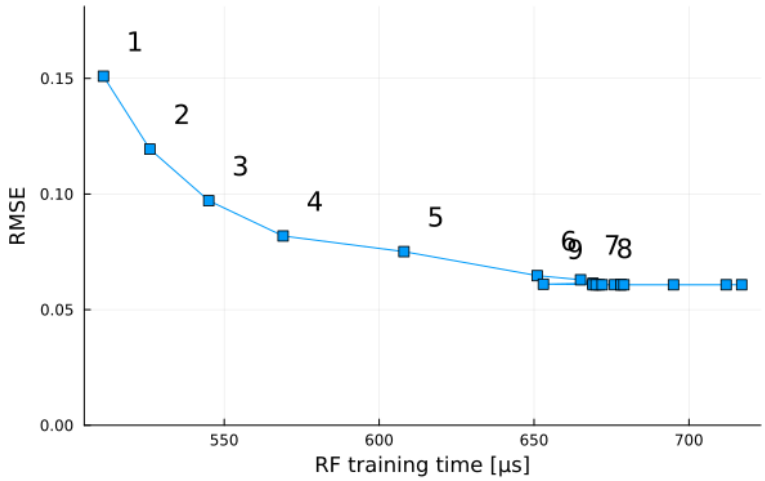


Figure 4.6: RMSE evolution for different RF depths for the RD engineering data

Model results

The results of this optimization are shown in Table 4.3.

	RF			**GPR**			**NN**		
	RMSE	Δt	Bad.	RMSE	Δt	Bad.	RMSE	Δt	Bad.
avg_D	0.0706	2.175	0.1536	0.0266	2577.5	68.6775	0.0764	16.375	1.2505
avg_L	0.3204	2.25	0.7208	0.3184	2530	805.6557	0.2398	15.325	3.6748
std_D	0.0406	2.375	0.0965	0.0247	2527.5	62.4874	0.0421	15.375	0.6469
std_L	0.0363	2.175	0.079	0.0657	2512.5	165.1089	0.0529	15.65	0.8279
Max_D	0.0922	2.325	0.2144	0.033	2570	84.9179	0.096	17.075	1.64
Max_L	0.0529	2.275	0.1204	0.0734	2615	191.9593	0.0638	15.825	1.0098
f1_D	0.1425	2.275	0.3241	0.2341	2545	595.871	0.2881	16.675	4.8033
f1_L	0.2997	2.2	0.6594	0.2056	2657.5	546.4006	0.2174	15.425	3.3532
f2_D	0.1291	2.2	0.284	0.1576	2680	422.4457	0.1727	15.5	2.6764
f2_L	0.1865	2.275	0.4243	0.1577	2532.5	399.3145	0.219	15.725	3.4438
f3_D	0.0389	2.225	0.0865	0.0967	2550	246.4652	0.1001	15.55	1.5558
f3_L	0.1954	2.25	0.4396	0.1233	2697.5	332.5343	0.1945	15.7	3.0541

Table 4.3: RD dataset: Engineering model results

From the data in Table 4.3, with the time given in milliseconds for the prediction of all samples, the following interesting conclusions can be extracted:

- Like for the OC data, the badness, defined as the linear product of the prediction time and RMSE, between the 12 different output variables is hard to compare because the RMSE is scaled due to the different original values. The large RMSE for the frequency data is a consequence of the huge randomness and small correlation within the data, even though the difference is now much smaller than for the OC data.
- GPR is generally the worst model in terms of badness. For the most important features, and RF is the best. Even though execution times are similar, their millisecond order of magnitude will in practice not weigh into the decision making.
- Visually all fits look equally good, just like for the OC data. For most frequency peak data there is a bigger correlation now, making their fit look better over the OC-data random clouds. This is true for the lift values as well. The RF contour data visually shows a much better prediction too, as can be seen in Appendix D.

In contrast to the OC data where GPR was the best predictor, here Random Forests prove to be better predictors for all data, even including the random looking frequencies.

4.3.2. RD Frequency domain prediction models

Data preparation

As the mean-max-refined scheme was shown to be optimal for the OC data, it is here used again without alteration. This was done to make the models as comparative as possible. The cut-off frequency now lies at 1 Hz, giving 400 FFT sample points per simulation. The refinement resulted in a smaller reduction of the amount of samples since the frequency peaks in the RD data were more pronounced, meaning that each simulation had clear peaks and thus refinement areas, whereas the OC data had some no-peak simulations.
Due to time constraints the RD data has only been used to train the ML models for this mean-max-refined data preparation scheme. However it is expected that implementation using rolling averages or simple binning will work without additional adaptations as well.

Model definition and optimization

The same philosophy for prediction, using three different "routes" is worked out for the RD data as well. Their build-up and software usage is identical, and therefore only summarized in Table 4.4. The same grid search as for the OC data has been performed in order to find the hyper parameters.

Model results

The table below summarizes all model hyper-parameters, as well as the model performance on the RD dataset.

		FD	DD	Full
RF	Depth	16	16	256
	RMSE	1.188	1.349	1.632
	Δt	5.167	4.888	9.861
GPR	Kernel	Matern-5/2 (l=0.0 - σ=0) + Squared Exponential (l=0.1 - σ=0)	RBF (l=1.0)	RBF (l=1.0)
	Mean	zero	zero	zero
	RMSE	1.611	2.002	4.554
	Δt	5.494	5.317	10.852
NN	Dense hidden layers	3	3	3
	Dense layer width	256-256-256	64-256-1028	64-256-1028
	Layer activation function	ReLU	ReLU	ReLU
	Batch size	8	64	64
	Learning rate	1e-4	1e-5	1e-5
	Weight decay	0.01	0.01	0.01
	Optimizer	Adam	Adam	Adam
	Loss function	MSE/mean	MSE/mean	MSE/mean
	RMSE	3.964	9.807	5.981
	Δt	2.750	20.142	19.932

Table 4.4: RD data prediction model parameter and performance comparison (RMSE in % and prediction time in milliseconds)

From the table a few different conclusions can be drawn:

- The models are quite identical to the ones used for the OC dataset, but all needed a bit more depth and complexity. This is due to the more pronounced frequency forcing after the FFT process, making less elaborate models view the peaks as outliers. Deeper, more over fitting networks were required to accurately capture these effects. This does make the prediction times similar to the ones found for the OC data.

- Interestingly the Full RF found a converged RMSE optimum at a depth of 16 branches, with a sudden, and at this moment unexplained, increase in accuracy at a depth of around 200. The increase was however minimal, especially considering the already, for engineering, accurate predictions, that the better prediction was not worth the additional computation time, even though it still only took a few 10s of milliseconds.
  The GPR models required no kernel alterations, which was expected due to the general applicability of the used RBF kernel (Duan et al., 2019).
  The NN workded quite well out of the box too, with some added depth and small hyperparameter alterations.

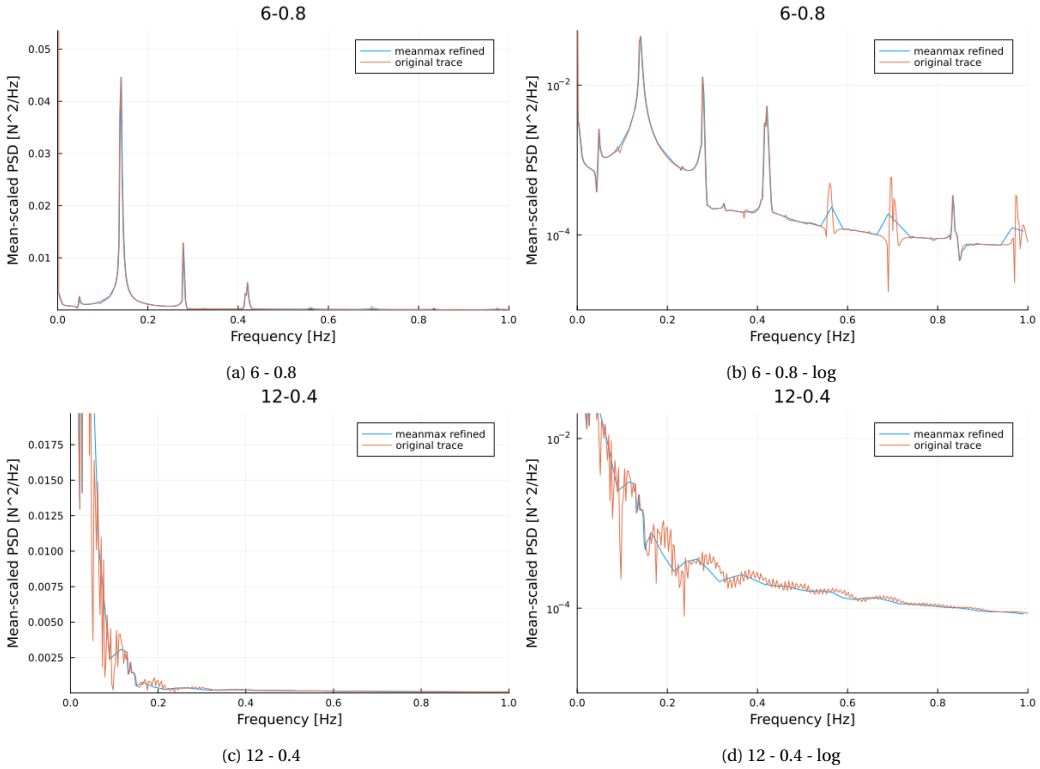
- Like for the OC data, the FD-RF model is the best predictor, with the full RF model not far behind and, from a practical point of view, having the upper hand

- A lot of models have a RMSE accuracy of around 1%, which is exceptionally good. The smoother frequency-domain traces are the main reason for this. Like before the NN's are a bit worse, but the difference is less pronounced than for the OC data. The original recommendation to further optimize this way of prediction therefore still stands. As a drawback, but being of millisecond order not a big one, the predictions do take twice as much time approximately. .

4.3.3. RD Analysis insights and design take-aways

Due to the more constant trends in the RD dataset, no new special features propped up after the ML data preparation. This peak-focused data also allows to omit a full appendix of representations like done earlier for the OC data, since the prediction of the full trace is now less important than the peak predictions. Like for the OC data, still as the number of perforation increases, the general average energy decreases. However the effect is less pronounced since the large cylinder diameter already filters out some of the large vortices from low-n designs.
Like for the OC data, these models for FFT converted predictions are not suited for constant and mean predictions.

As a final take-away, it is recommended to adapt the data preparation scheme in the future to focus on log energy scales, as shown in Figure 4.7. On a log10 scale the energy peaks become much more comprehensible, especially in the case of narrow-peaked spectra. Additionally the effect of the preparation technique and the binning is better visualized. This may indicate that a machine learning training process on these log values would yield more accurate results as well, especially in the close-to-zero tail. Also for more random signals the log scale shows more interesting effects in this higher frequency region.



(a) 6 - 0.8

(b) 6 - 0.8 - log

(c) 12 - 0.4

(d) 12 - 0.4 - log

Figure 4.7: Visualization of the frequency domain on normal and log scales

4.4. OC and RD similarities and discrepancies

Below a list is added of notable conclusions from the comparison of the two datasets, with hopes of facilitating future research using a combination of the strong points of both.

- It is assumed that the small changes in meshing strategy and the way the perforations are cut out do not contribute greatly to the results, other than changing the convergence-properties of the Stokes pre-conditioner. A continuation of the meshing scheme used for the OC data (rectangular rather than triangular cutouts from the cylinder center) is advised since less simulations crashed using the OC meshes. The non-convergence could also be related to the diameter scale itself, but since all values in the mesh are

scaled on the diameter in both meshing schemes this is unlikely. The test for this should however be fast and simple enough so as to not ignore the possibility.

- The effect of the larger cylinder on the smoothness of the data in the frequency domain was very important. This indicates that scaled simulation results are inaccurate, and should be avoided.
- The increased design space size, with more extreme (high and low) n and β gave some additional insights as well. The large variations from the beat-pattern for low-n and high-β samples are the first that spring to mind. However further analysis of these extremes may be futile once more details on the structural side of this perforated monopile concept become available.
- The larger cylinder also showed a much larger time required for development of the wake field, requiring up to 600 seconds of simulation times. While the OC data was only ran for 200 seconds, their traces showed better signs of a developed state than some of the RD traces in the 100-200 second region. But, to be fully correct, at this point it is unknown what would happen if the OC simulations were ran for longer. It does make sense that the 1 m diameter, so in total 7 m long OC domains develop faster for a 1 m/s inflow than the 10 m diameter and thus 70 m length domains do for the same 1 m/s inflow velocity.

5

Conclusions and recommendations

5.1. Conclusions

To increase the water depth at which conventional offshore wind turbine monopiles remain economical, the concept of M. C. Anderson in which the monopile is perforated around the splash zone provides interesting possibilities. To simulate the effects of the perforations on the water flow and in order to predict the consequent drag and lift forces a transient fluid flow model should capture the required flow and turbulence effects at a reasonable computational cost. For preliminary design and concept evaluation it is not required to include manufacturing details, installation and noise mitigation solutions, scour effects, corrosion and acidification, or marine growth, nor any 3D and hydro-elastic effects.

In order to solve the Navier-Stokes equation in this case an LES model provides the accuracy-cost balance needed. From the three closure models tested, Smagorinsky-Lilly and WALE proved to require much smaller time steps, whereas a VMS scheme provided the required stability at a much lower computational cost. Parametric design of the monopiles based on wall thickness and angle of attack showed that the influence of the former was negligible, whereas the latter only relevantly changed the lift patterns. The existence of logical flow-paths through the monopile, mainly at a low number of perforations, is expected to be a better measure for lift and drag effects than the angle itself. The monopile diameter and the inflow velocity were found to impact the drag in relation to the Morison equation, allowing them to be excluded in coefficient-based force reduction determination. The most important influence of the diameter is that larger cross sections average out small pressure fluctuations, resulting in smoother time-drag histories. Varying porosity and number of perforations remains the key of this novel monopile design process, with some combinations allowing for modeled force reductions of up to 55% for small diameter cylinders and 50% for larger ones when excluding “extreme” porosities which are assumed to be structurally invalid. For these conclusions it was required to have simulations exceeding 200 seconds, since a 1 m/s inflow velocity took that time to fully develop and for the vortices to “lock into” the wake region. The RD data also showed that the conclusion of Star (2022), namely that lift is only relevant for $n \leq 4$, could be extended such that lift will only contribute really for low-n designs which are also low in porosity. Additionally for low-n designs, it is important to closely monitor the beat-effect of extremely high-β samples. From the RD data, with some engineering-feel limitations applied, a porosity of around 0.5 will most likely give the best trade-off between strength and mean force reduction. For 10 to 12 perforations the cost of cutting the holes will most likely stay reasonable, and the high number of perforations reduces the total amplitude that the force trace has.

To facilitate the design process two datasets, one pre-existing for small diameters (OC), and one with large diameters (RD), were used to train a variety of Random Forest Regressors, Gaussian Process Regressors, and Neural Networks. By using binned-data which got refined locally at locations of interest, the dataset is reduced to a trainable size. While optimization is still required, Neural Networks are expected to provide the best basis for accurate and trend-based predictions. General values such as mean and expected maximum drag can best be predicted using a GPR model for small diameter, and a RF model for large diameter designs. To predict a full energy-spectrum for future resonance checks a RF model performs best, with errors being less than 6% and prediction times in the order of milliseconds. A speed-up factor over three hundred thousand over the original CFD simulations can be achieved. Based on input of the number of perforations, and the porosity, one can use these models to predict the energy for a specific design at any frequency, as an engineer would require to ensure dynamic stability of future monopile designs. However, this model type is on its merits not suited for mean value predictions due to the use of the frequency domain.

5.2. Recommendations and future work

During the study a number of subjects was found not to be extensively covered in earlier works, or which were excluded from the current scope at a later stage. These are discussed in a T-shaped manner, covering a wide and a deep dimension.

For width the expansion of the code and models to include additional CFD and structural effects is required in order to verify whether or not their added computational expense is justified to get more accurate predictions, or even more rudimentary of the economic implications of perforated monopiles will help advance the industry as intended. The most important concepts are:

- The influence of angle of attack variations and the flow-path shape through the perforations on drag, but especially lift time-traces. The importance of the effect and the development at low numbers of perforation are the key questions. The hypothesis that flow-relations may be more important than angular values may provide an opportunity for CNN models to be applied for surrogate prediction.
- 3D effect inclusion, with wave motions and free surfaces or multi-phase flow, even though the conclusions of the current research indicated that Morison scaling laws are still applicable. With this a further look into numerical stability of the Stokes preconditioner can be worked in as to determine if rectangular cutouts rather than triangular ones are really, also from a manufacturing point of view, more beneficial.
- FSI, VIV, resonance and frequency analysis. Since scaling seems to follow Morison's theory, one-way coupling may already give a relatively cheap but relevant addition to the modelling scheme.
- Fatigue determination and inclusion in a full turbine model.
- Design optimization of the hole lay-out and shape, and practical manufacturing details with possible inclusion of the perforations in transition pieces rather than in the monopiles themselves. The optimization should be based on the relative impact that adding more or bigger holes has in reducing the drag, compared against the structural weakening which comes from the removal of material and introduction of stress concentrations around the holes. Elliptical holes in vertical directions may already reduce these concentrations.

On an even larger scale, subjects that fell out of the scope of this thesis and that provide main research topics for future work are:

- Accurate numerical or physical experiments for the determination of p-y curves for large monopiles.
- Accurate numerical or physical experiments for the determination of SCF of curved perforated plates with large holes.
- Detailed information of costs and scaling rules for XXL monopile foundations.
- The importance and need of FSI inclusion.

Diving deeper and more detailed into recommendations to this thesis specifically, there are four main areas where future work is required:

- A detailed analysis of the structural possibilities and limitations in order to decrease the size of the solution space and amount of simulations needed in future design or research, and in order to guide future CFD research to the prediction values which really matter, like for example frequencies to avoid or the relevance of lift in general.
- Upgrades on the domain and the modeled structure. A few ideas are given here:
  - One could look into Dirichlet-Neumann combined boundary conditions for the side walls, as explained in Chapter 3.
  - Now the cutouts in the OC and RD datasets are based o rectangles or triangles respectively, originating from the cylinder center and then being rotated to mark the different holes. In real world applications, especially in areas with a dominant single-direction current, a cut-out straight through both walls of the cylinder may more sense. This way the flow path of the fluid is more straightforward, which may minimize drag effects.

  - Implementation of periodic flows for wave-simulations. It must be noted that the assumption in this thesis of drag-dominated flows would then disappear to make way for a more inertia dominated scheme with different U-D scaling laws. The further validity of Morison should then be rechecked. For this all domain boundaries require a redefinition, with the top and bottom one still consisting of a Dirichlet or Dirichlet-Neumann combined bound, only with an x-direction velocity which varies over time and along the wall, depending on the wave period and length. Similarly the now left inflow and outflow bound would require a Dirichlet-Neumann combination which simulates periodic or irregular waves in time, with the inflow boundary still forcing a constant current and the outflow boundary having a Neumann element to accommodate a constant-current outflow. It is expected that the distance between the inflow boundary and the cylinder should increase to accommodate the waves themselves. For a base case it is most likely not required to add options to control the wave direction in the domain. Wave crests which are aligned with the y-direction and travel purely in x-direction would work in all cases, since it is easier to just rotate the cylinder. Expansion to irregular waves in both amplitude, period and direction would make this a necessity.

- Further optimization of the LES solver and numerical schemes, especially concerning the now brute-force Navier-Stokes solution (PCLU factorization) and the absence of variable time-stepping in Gridap.jl. The implementation of variable time stepping may help reduce calculation times in the future as well, especially if multiple cores calculating different segmented parts of the domain could use their own optimized time step. Similarly further optimization of, or the upgrade to a different integrator than generalized alpha could better the execution times as well.

- A focused surrogate-model study aiming to verify and improve the machine learning models, especially the Neural Network options. The recommendations for future work cover a wide array of possibilities.

  - The absolute simplest model, further optimization of the hyper-parameters, and the effect of removing or introducing random batch sampling, weight initialization, normalization, ... are all relevant questions. To this also comes the search for an explanation as to why the Random Forest approach here found 1 tree to be optimal.
  - A re-run of the found analysis focusing on a log-valued energy input. At the end of this project the alternative representation added some key insights, so a re-fit of the ML models may do so as well.
  - Amongst many additional options that could be considered, novel work on DeepONets is showing promising results on the prediction of dynamic systems and differential equation results. This would allow exponential model convergence even for small datasets (Lu et al., 2019).
  - An additional DDM expansion could be to look into flow-pattern and vorticity predictions, especially in the inner-cylinder water area or in the near-cylinder wake field. These 2D phenomena would most likely require a conversion back to CNN predictions, both for decoding as encoding phases.
  - In Julia, while Flux.jl was used here, a conversion to ScikitLearn.jl for all models, or similarly PyTorch.jl may reduce the amount of software or dependency complexity required to run all calculations and prediction schemes. The Python link additionally allows for better documentation and, possibly, performance over Flux.jl. Even if performance is not improved, this will most likely not pose any problems due to the now very fast and cheap predictions.
  - A final topic in the surrogate-modelling department could be to check the validity of 2D models for 3D predictions. Based on a few, computationally very expensive, CFD simulations the link between 2D and 3D results could be determined. This allows designers and engineers to only having to run a 2D simulation to get the 3D results.

Bibliography

Alagan Chella, M., Bihs, H., & Myrhaug, D. (2019). Wave impact pressure and kinematics due to breaking wave impingement on a monopile. *Journal of Fluids and Structures*, *86*, 94–123. https://doi.org/https://doi.org/10.1016/j.jfluidstructs.2019.01.016

Amit, Y., & Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural computation*, *9*(7), 1545–1588.

Andersen, J., Abrahamsen, R., Andersen, T. L., Andersen, M. T., Baun, T. L., & Neubauer, J. L. (2020). Wave load mitigation by perforation of monopiles. *Journal of Marine Science and Engineering*, *8*(5). https://doi.org/10.3390/jmse8050352

Anderson, B. W., & Domaradzki, J. A. (2012). A subgrid-scale model for large-eddy simulation based on the physics of interscale energy transfer in turbulence. *Physics of Fluids*, *24*(6), 065104. https://doi.org/10.1063/1.4729618

Anderson, M. C. (2017). *The hybrid monopile: Design of a novel foundation structure for large offshore wind turbines in intermediate water depths* (Master's thesis). Delft University of Technology.

Arnold, D. N., Brezzi, F., Cockburn, B., & Marini, L. D. (2001). Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM J. Numer. Anal.*, *39*, 1749–1779.

Badia, S., & Verdugo, F. (2020). Gridap: An extensible finite element toolbox in julia. *Journal of Open Source Software*, *5*(52), 2520. https://doi.org/10.21105/joss.02520

Beck, A., Flad, D., & Munz, C.-D. (2019). Deep neural networks for data-driven les closure models. *Journal of Computational Physics*, *398*, 108910. https://doi.org/https://doi.org/10.1016/j.jcp.2019.108910

Bento, C. (2021). Stochastic gradient descent explained in real life. https://towardsdatascience.com/stochastic-gradient-descent-explained-in-real-life-predicting-your-pizzas-cooking-time-b7639d5e6a32

Bianchini, A., Balduzzi, F., Bachant, P., Ferrara, G., & Ferrari, L. (2017). Effectiveness of two-dimensional cfd simulations for darrieus vawts: A combined numerical and experimental assessment. *Energy Conversion and Management*, *136*, 318–328. https://doi.org/https://doi.org/10.1016/j.enconman.2017.01.026

Byrne, C., Dickson, T., Lauber, M., Cairoli, C., & Weymouth, G. (2022). Using Machine Learning to Model Yacht Performance. *Journal of Sailing Technology*, *7*(01), 104–119. https://doi.org/10.5957/jst/2022.7.5.104

Cadence. (2022). Cfd simulation types: Discretization, approximation, and algorithms. https://resources.pcb.cadence.com/blog/2020-cfd-simulation-types-discretization-approximation-and-algorithms

Chakrabarti, S. K. (2005). *Handbook of offshore engineering*. Offshore Structure Analysis Inc.

Charnyi, S., Heister, T., Olshanskii, M. A., & Rebholz, L. G. (2019). Efficient discretizations for the emac formulation of the incompressible navier–stokes equations. *Applied Numerical Mathematics*, *141*, 220–233.

Christianson, A. (2020). Why is julia so fast. https://www.quora.com/Why-is-Julia-so-fast?share=1

Colagrossi, A., Nikolov, G., Durante, D., Marrone, S., & Souto-Iglesias, A. (2019). Viscous flow past a cylinder close to a free surface: Benchmarks with steady, periodic and metastable responses, solved by meshfree and mesh-based schemes. *Computers & Fluids*, *181*, 345–363. https://doi.org/https://doi.org/10.1016/j.compfluid.2019.01.007

Colomés, O. (2023). Github repository: Perforetedculinder.jl. https://github.com/oriolcg/PerforatedCylinder.jl

Colomés, O., Badia, S., Codina, R., & Principe, J. (2015). Assessment of variational multiscale models for the large eddy simulation of turbulent incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, *285*, 32–63. https://doi.org/https://doi.org/10.1016/j.cma.2014.10.041

Colomés, O., Badia, S., & Principe, J. (2016). Mixed finite element methods with convection stabilization for the large eddy simulation of incompressible turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, *304*, 294–318. https://doi.org/https://doi.org/10.1016/j.cma.2016.02.026

Davis, K. (2019). How deep can the monopile go in offshore wind? https://www.empireengineering.co.uk/how-deep-can-the-monopile-go-in-offshore-wind/

De Stefano, G., & Vasilyev, O. V. (2004). "perfect" modeling framework for dynamic sgs model testing in large eddy simulation. *Theoretical and Computational Fluid Dynamics*, *18*(1), 27–41. https://doi.org/10.1007/s00162-004-0146-0

Deardorff, J. W. (1970). A numerical study of three-dimensional turbulent channel flow at large reynolds numbers. *Journal of Fluid Mechanics*, *41*(2), 453–480. https://doi.org/10.1017/S0022112070000691

Deltares. (2018). Comflow developers manual. http://poseidon.housing.rug.nl/sphinx/index.html

DHPC, D. H. P. C. C. (2022). DelftBlue Supercomputer (Phase 1).

Duan, Y., Cooling, C., Ahn, J. S., Jackson, C., Flint, A., Eaton, M. D., & Bluck, M. J. (2019). Using a gaussian process regression inspired method to measure agreement between the experiment and cfd simulations. *International Journal of Heat and Fluid Flow*, *80*, 108497. https://doi.org/https://doi.org/10.1016/j.ijheatfluidflow.2019.108497

Eichinger, M., Heinlein, A., & Klawonn, A. (2021). Stationary flow predictions using convolutional neural networks. In F. J. Vermolen & C. Vuik (Eds.), *Numerical mathematics and advanced applications enumath 2019* (pp. 541–549). Springer International Publishing.

Elger, D. F., Lebret, B. A., Crowe, C. T., & Roberson, J. A. (2016). *Engineering fluid mechanics*. Wiley.

European Commission. (2022). Offshore renewable energy. https://energy.ec.europa.eu/topics/renewable-energy/offshore-renewable-energy_en

Fernández, P., Moura, R. C., Mengaldo, G., & Peraire, J. (2018). Non-modal analysis of spectral element methods: Towards accurate and robust large-eddy simulations. *Computer Methods in Applied Mechanics and Engineering*.

Flow3D. (2022). Outflow boundary conditions. https://www.flow3d.com/resources/cfd-101/physical-phenomena/outflow-boundary-conditions/

GMSH. (2022). Gmsh reference manual for gmsh 4.11.1. http://gmsh.info/doc/texinfo/gmsh.html

Greenshields, C., & Weller, H. (2022). *Notes on computational fluid dynamics: General principles*. CFD Direct Ltd.

Guan, J. T., & Liu, H. Y. C. (2018). *Computational fluid dynamics: A practical approach*. Butterworth-Heinemann.

Gupta, B. K., & Basu, D. (2020). Offshore wind turbine monopile foundations: Design perspectives. *Ocean Engineering*, *213*, 107514. https://doi.org/https://doi.org/10.1016/j.oceaneng.2020.107514

Haddoukessouni, B. (2021). Better design decisions for your cfd simulations with surrogate models. https://blogs.sw.siemens.com/simcenter/surrogate-model-cfd-simulation/

Haiderali, A., & Madabhushi, G. (2012). Three-dimensional finite element modelling of monopiles for offshore wind turbines.

Hermans, K., & Peeringa, J. (2016). *Future xl monopile foundation design for a 10 mw wind turbine in deep water*. ECN.

Hu, X., & Han, L. (2017). Sph modeling of fluid-structure interaction.

Ideal Simulations. (2020). Turbulence models in cfd. https://www.idealsimulations.com/resources/turbulence-models-in-cfd/

IEA. (2019). Offshore wind geospatial analysis. https://www.iea.org/data-and-statistics/data-tools/offshore-wind-geospatial-analysis

Jaax, A. (2016). Skill relatedness and economic restructuring: The case of bremerhaven. *Regional Studies, Regional Science*, *3*, 58–66. https://doi.org/10.1080/21681376.2015.1116958

Jansen, K. E., Whiting, C. H., & Hulbert, G. M. (2000). A generalized-α method for integrating the filtered navier–stokes equations with a stabilized finite element method. *Computer Methods in Applied Mechanics and Engineering*, *190*, 305–319.

Jeanne Yacoubou. (2022). Utility and residential wind turbine costs explained. https://greencoast.org/wind-turbine-costs/

Jiang, C. M., Wang, D., Huang, J., Marcus, P., & Niessner, M. (2019). Convolutional neural networks on non-uniform geometrical signals using euclidean spectral transformation. *International Conference on Learning Representations*. https://openreview.net/forum?id=B1G5ViAqFm

Journée, J., & Massie, W. (2008). *Offshore hydromechanics*. Delft University of Technology.

Kaya, M. (2019). A cfd based application of support vector regression to determine the optimum smooth twist for wind turbine blades. *Sustainability*, *11*, 4502. https://doi.org/10.3390/su11164502

Kim, M. W., Koo, W., & Hong, S. Y. (2014). Numerical analysis of various artificial damping schemes in a three-dimensional numerical wave tank. *Ocean Engineering*, *75*, 165–173. https://doi.org/https://doi.org/10.1016/j.oceaneng.2013.10.012

Komarniczky, B. (2022). Build your first neural network with flux.jl in julia. https://towardsdatascience.com/build-your-first-neural-network-with-flux-jl-in-julia-10ebdfcf2fa3

Kukacka, J. (2019). Neural network (or deep learning) or gradient boosting for large data with a few features. https://stats.stackexchange.com/questions/431689/neural-network-or-deep-learning-or-gradient-boosting-for-large-data-with-a-few

Kuron, M. (2015). 3 criteria for assessing cfd convergence. https://www.engineering.com/story/3-criteria-for-assessing-cfd-convergence

Launder, B., & Spalding, D. (1974). The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, *3*(2), 269–289. https://doi.org/https://doi.org/10.1016/0045-7825(74)90029-2

Lehmkuhl, O., Houzeaux, G., Owen, H., Chrysokentis, G., & Rodriguez, I. (2019). A low-dissipation finite element scheme for scale resolving simulations of turbulent flows. *Journal of Computational Physics*, *390*, 51–65. https://doi.org/https://doi.org/10.1016/j.jcp.2019.04.004

Lilly, D. K. (1992). A proposed modification of the germano subgrid-scale closure method. *Physics of Fluids A: Fluid Dynamics*, *4*(3), 633–635. https://doi.org/10.1063/1.858280

Liu, B., Tang, J., Huang, H., & Lu, X.-Y. (2020). Deep learning methods for super-resolution reconstruction of turbulent flows. *Physics of Fluids*, *32*(2), 025105. https://doi.org/10.1063/1.5140772

Liu, Y., Shi, W., Wang, W., Li, X., Qi, S., & Wang, B. (2022). Dynamic analysis of monopile-type offshore wind turbine under sea ice coupling with fluid-structure interaction. *Frontiers in Marine Science*, *9*. https://doi.org/10.3389/fmars.2022.839897

Lozano-Minguez, E., Kolios, A., & Brennan, F. (2011). Multi-criteria assessment of offshore wind turbine support structures. *Renewable Energy*, *36*(11), 2831–2837. https://doi.org/https://doi.org/10.1016/j.renene.2011.04.020

Lu, L., Jin, P., & Karniadakis, G. E. (2019). Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *CoRR*, *abs/1910.03193*. http://arxiv.org/abs/1910.03193

Manzanero, J., Ferrer, E., Rubio, G., & Valero, E. (2018). On the role of numerical dissipation in stabilising under-resolved turbulent simulations using discontinuous galerkin methods.

Méchartes. (2021). Fem vs. fvm: All you need to know. https://www.pages.mechartes.com/resources/2021/12/28/fem-fvm-all-need-know

Mendoza, A. S. E., Griffith, D. T., Qin, C., Loth, E., & Johnson, N. (2022). Rapid approach for structural design of the tower and monopile for a series of 25 mw offshore turbines. *Journal of Physics: Conference Series*, *2265*(3), 032030. https://doi.org/10.1088/1742-6596/2265/3/032030

Merrifield, R. (2018). Offshore wind farms to test business in deep water. https://ec.europa.eu/research-and-innovation/en/horizon-magazine/offshore-wind-farms-test-business-deep-water

Morozova, N., Trias, F. X., Capdevila, R., Schillaci, E., & Oliva, A. (2022). A cfd-based surrogate model for predicting flow parameters in a ventilated room using sensor readings. *Energy and Buildings*, *266*, 112146. https://doi.org/https://doi.org/10.1016/j.enbuild.2022.112146

Murphy, G., Igoe, D., Doherty, P., & Gavin, K. (2018). 3d fem approach for laterally loaded monopile design. *Computers and Geotechnics*, *100*, 76–83. https://doi.org/https://doi.org/10.1016/j.compgeo.2018.03.013

Nguyen, T., Strebinger, C., Bogin, G., & Brune, J. (2021). A 2d cfd model investigation of the impact of obstacles and turbulence model on methane flame propagation. *Process Safety and Environmental Protection*, *146*, 95–107. https://doi.org/https://doi.org/10.1016/j.psep.2020.08.023

O'Boyle, L., Doherty, K., van 't Hoff, J., & Skelton, J. (2015). The value of full scale prototype data – testing oyster 800 at emec, orkney.

Oil & Gas Today. (2020). Deep water wind farms take renewable energy offshore. https://www.oilandgastoday.com.au/deep-water-wind-farms-take-renewable-energy-offshore/

ORE Catapult. (2021). Wind farm costs. https://guidetoanoffshorewindfarm.com/wind-farm-costs

Orszag, S. A. (1970). Analytical theories of turbulence. *Journal of Fluid Mechanics*, *41*(2), 363–386. https://doi.org/10.1017/S0022112070000642

OWF Alpha. (2022). Design of monopile for offshore wind turbine – simple example. https://www.offshoreengineering.com/wind-monopile-design/design-of-monopile-for-offshore-wind-turbine-basic-design-example/

Powell, A. (2021). Gaussian process regression. https://towardsdatascience.com/intro-to-gaussian-process-regression-14f7c647d74d

Prandtl, L. (1928). Bericht uber untersuchungen zur ausgebildeten turbulenz. *Zs. Angew. Math. Mech*, *136*. https://doi.org/https://doi.org/10.1002/zamm.19250050212

Ronaghan, S. (2018). Machine learning: Trying to predict a numerical value. https://srnghn.medium.com/machine-learning-trying-to-predict-a-numerical-value-8aafb9ad4d36

Samira Keivanpour, D. A. K., Amar Ramudhin. (2017). The sustainable worldwide offshore wind energy potential: A systematic review. *Journal of Renewable and Sustainable Energy*, *9*.

Schlichting, H. (1955). In *Boundary layer theory, etc.* London; Verlag G. Braun: Karlsruhe; Karlsruhe printed.

Shen, C., Yu, P., Wang, T., & Chen, N.-Z. (2022). A cfd based kriging model for predicting the impact force on the sphere bottom during the early-water entry. *Ocean Engineering, 243*, 110304. https://doi.org/https://doi.org/10.1016/j.oceaneng.2021.110304

Soleimani, A., Zamani, F., & Haghshenas Gorgani, H. (2022). Buckling analysis of three-dimensional functionally graded euler-bernoulli nanobeams based on the nonlocal strain gradient theory. *Journal of Computational Applied Mechanics*, *53*(1), 24–40. https://doi.org/10.22059/jcamech.2022.338327.689

Star, J. (2022). *Surrogate models for the characterization of hydrodynamic loads on perforated monopiles* (Master's thesis). Delft University of Technology.

Statista. (2022). A 2030 vision for european offshore wind ports. https://www.statista.com/statistics/476327/global-capacity-of-offshore-wind-energy/

Steelwind Nordenham. (2022). Mega monopiles produktionskapazitäten. https://www.steelwind-nordenham.de/steelwind/produkte/megamonopiles/

Sumer, B. M., & Fredsoe, J. (2006). *Hydrodynamics around cylindrical structures*. Technical university of Denmark.

SURF. (2022). Dutch national supercomputer snellius.

Suryad, B. (2022). Top 10 fastest programming languages. https://www.geeksforgeeks.org/top-10-fastest-programming-languages/

Taylor, R. E. (2007). Hydroelastic analysis of plates and some approximations. *Journal of Engineering Mathematics*, *58*, 267–278. https://doi.org/https://doi.org/10.1007/s10665-006-9121-7

Tödter, S., el Sheshtawy, H., Neugebauer, J., el Moctar, O., & Schellin, T. E. (2021). Deformation measurement of a monopile subject to vortex- induced vibration using digital image correlation. *Ocean Engineering*, *221*, 108548. https://doi.org/https://doi.org/10.1016/j.oceaneng.2020.108548

University of Strathclyde. (2015). Xl monopile case study. https://www.esru.strath.ac.uk/EandE/Web_sites/14-15/XL_Monopiles/structural.html

US Department of Energy. (2021). Energy secretary granholm announces ambitious new 30gw offshore wind deployment target by 2030. https://www.energy.gov/articles/energy-secretary-granholm-announces-ambitious-new-30gw-offshore-wind-deployment-target

van der Ploeg, J. (2021). *Perforation of monopiles to reduce hydrodynamic loads and enable use in deep waters* (Master's thesis). Delft University of Technology.

Varela, P., Suárez, P., Alcántara-Ávila, F., Miró, A., Rabault, J., Font, B., García-Cuevas, L. M., Lehmkuhl, O., & Vinuesa, R. (2022). Deep reinforcement learning for flow control exploits different physics for increasing reynolds number regimes. *Actuators*, *11*(12). https://doi.org/10.3390/act11120359

Verdugo, F., & Badia, S. (2022). The software design of gridap: A finite element package based on the julia JIT compiler. *Computer Physics Communications*, *276*, 108341. https://doi.org/10.1016/j.cpc.2022.108341

Walters, D. K., Bhushan, S., Alam, M. F., & Thompson, D. S. (2013). Investigation of a dynamic hybrid rans/les modelling methodology for finite-volume cfd simulations. *Flow Turbulence Combust*, *91*, 643–667. https://doi.org/https://doi.org/10.1007/s10494-013-9481-9

Wang, X., Zeng, X., Li, J., Yang, X., & Wang, H. (2018). A review on recent advancements of substructures for offshore wind turbines. *Energy Conversion and Management*, *158*, 103–119. https://doi.org/https://doi.org/10.1016/j.enconman.2017.12.061

Waterkaart.net. (2023). Stroomsnelheid noordzee in knopen. https://waterkaart.net/gids/stroomatlas-noordzee.php?p=knopen

Wellens, P. (2012). Wave simulation in truncated domains for offshore applications. *Delft university press*.

Wind Europe. (2021). Deep water wind farms take renewable energy offshore. https://windeurope.org/intelligence-platform/product/a-2030-vision-for-european-offshore-wind-ports-trends-and-opportunities/

Wollblad, C. (2018). Your guide to meshing techniques for efficient cfd modeling. https://www.comsol.com/blogs/your-guide-to-meshing-techniques-for-efficient-cfd-modeling/

Zhang, J., Sun, K., Wang, Z.-Q., Zhang, L., & Hao, J. (2010). Static and dynamic analysis of monopile foundation for offshore wind farm. *Proceedings of the International Offshore and Polar Engineering Conference*, *1*.

Zhang, Y., Sung, W., & Mavris, D. (2018). Application of convolutional neural network to predict airfoil lift coefficient. https://doi.org/10.2514/6.2018-1903

A

LES-VMS implementation scheme in Gridap.jl: tutorial

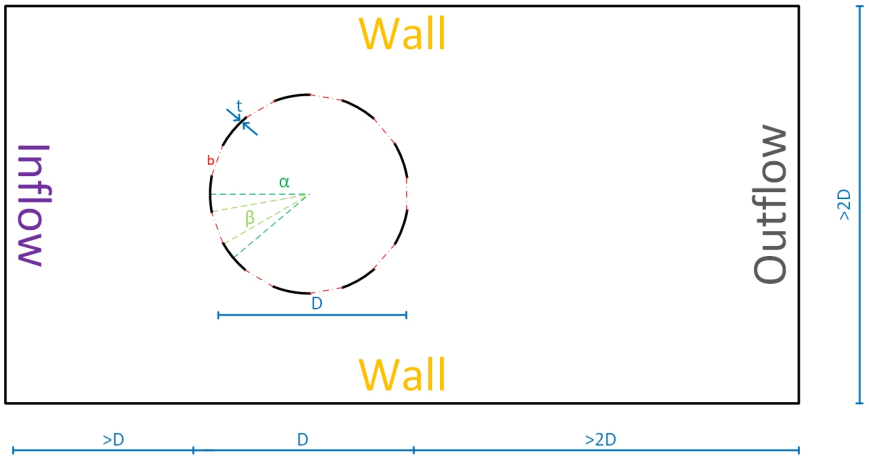
A transient LES-VMS scheme: flow through a perforated cylinder

Author: Ruben Dekeyser and Oriol Colomés
Published: June 2023
Gridap version: Gridap@0.17.17

This tutorial shows the implementation of an external mesh for a perforated cylinder in 2D in Gridap.jl. The goal is to get a time trace of the drag and lift force experienced by the monopile.

A. Problem description

The goal is to model a steady uniform flow through a perforated cylinder. The general domain looks as follows:



To this end, the problem is split into a series of consecutive steps:

1. Set up the problem
    i. Define the model parameters and input values
    ii. Define the geometry
    iii. Define the boundary conditions
2. Set up the numerical scheme
    i. Set up the test spaces
    ii. Build trial spaces
    iii. Combine them in a (transient) multifield
    iv. Initialize the domain
    v. Define the weak form
    vi. Set up the time integration scheme
3. Apply the nonlinear solver and time integration
4. Do post-processing

For engineering research into perforated monopiles the post processing should be built around the following data:

1. The velocity and pressure fields (usually in vtk format). This will be useful in the early stages of the development to look into the exact flow effects, eddies, ...

2. Drag and lift time-traces, either in the time or frequency domain, for further design analysis and trend-seeking.

B. Set up of the module and surrounding parallelization options

We start by importing all required packages in a module. This module allows for easier parallel execution in cluster machines. The 2 files that are imported containt the actual Navier Stokes implementation, and the mesh generator. the latter is not elaborated on here. However, the code can still be publically accessed.

```
using Gridap
using Gridap.FESpaces: zero_free_values, interpolate!
using Gridap.Fields: meas
using GridapGmsh: gmsh, GmshDiscreteModel
using GridapDistributed
using GridapDistributed: DistributedTriangulation, DistributedCellField
using GridapPETSc
using GridapPETSc: PETSC
using PartitionedArrays
using SparseMatricesCSR

using CSV
using DataFrames

include("NavierStokesParallel.jl")
include("mesh_generation.jl")
```

The meshes can be generated by calling a dedicated funtion, taking the different diameters, wall thicknesses, perforations, porosities, angles of attack and domain lengths as inputs. This creates meshes of all their combinations. To get a single mesh, just input 1-element lists. By changing the boolean `do_mesh` value it is possible to easily get a list of all generated files, without having to actually generate them. This is useful if one wants to re-run simulations.

```
function generate_meshes(; nD=[10], nRt=[100], nperf=[12], nβ=[0.5], nα=[0], nR_L=[7], do_mesh=true)
  # Create cases
  cases = []
  # RDK values
  for D in nD
    for Rt in nRt
      for num_perforations in nperf
        for β in nβ
          for α in nα
            for R_L in nR_L
                D2 =  round(D;digits=2)
                Rt2 = round(Rt;digits=0)
                β2 =  round(β;digits=2)
                α2 =  round(α,digits=2)
                filename = "D$D2-Rt$Rt2-n$num_perforations-beta$β2-alfa$α2-RL$R_L.msh"
                push!(cases,replace(filename, r".msh"=>""))

                if do_mesh==true
                  create_mesh(filename=filename, D=D, R_t=Rt, num_perforations=num_perforations, R_β=β,
α=360/num_perforations*α,
                              R_L=R_L, R_Cx=R_L-5.5)
                end
              end
          end
```

```
        end
      end
    end
  end

  return cases

end
```

After that a series of `mumps` flags is defined for the module to work correctly:

```
options_mumps = "-snes_type newtonls \
-snes_linesearch_type basic  \
-snes_linesearch_damping 1.0 \
-snes_rtol 1.0e-8 \
-snes_atol 1.0e-10 \
-ksp_error_if_not_converged true \
-ksp_converged_reason -ksp_type preonly \
-pc_type lu \
-pc_factor_mat_solver_type mumps \
-mat_mumps_icntl_7 0"
```

Lastly the actual parallel solver can be called. The default values of the parameters allow easy execution of a warmup loop, allowing a system image to be stored on a cluster. That way the system image can be used on all calculation cores, without having to recompile the module each time. This saves multiple hours of computation time for high parallel computing.

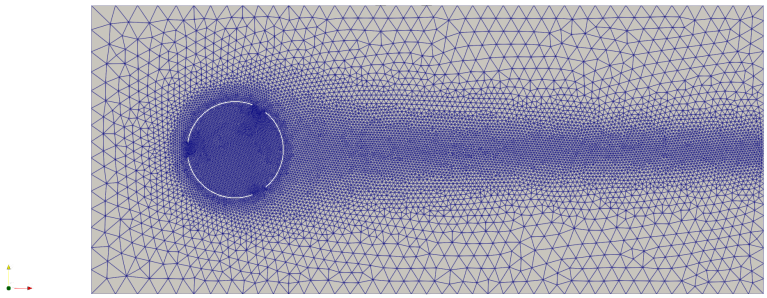
```
function main_parallel(np;
  mesh_file="tmp_mesh_coarse.msh",
  vtk_outpath="tmp_mesh_coarse",
  Vinf=1,
  Δt=0.1,
  tf=1.0,
  Δtout=0.5,
  output_path=joinpath(ENV["PerforatedCylinder_DATA"],"vtk"))

  filename = replace(mesh_file, r".msh"=>"")
  run_name = filename*"-Vinf$Vinf-dt$Δt"

  current_path = pwd()
  cd(output_path)
  with_backend(MPIBackend(),np) do parts
    options = options_mumps
    GridapPETSc.with(args=split(options)) do
      run_test_parallel(parts,run_name,mesh_file,vtk_outpath,Vinf,Δt,tf,Δtout)
    end
  end
  cd(current_path)
end
```

1. Setup of the problem

The model gets 3 main inputs: the mesh, the inflow velocity, and the time step. The mesh is triangulated and split into a fluid and a solid domain. Additionally the boundaries are read from predefined tags from the mesh-file. The bounds of all these sub-domains are `Measured` to be used in integration schemes later, and additionally the normal directions of the structure and the outflow boundary are calculated. The boundary conditions are: a uniform (Dirichlet) inflow of speed V_{in}, prescribed-velocity $(V_{in}, 0)$ walls and a no-slip no-penetration structure. The outflow boundary is included in the weak form directly.



For the code we define a more elaborate structure.

1.1 Define the model parameters and input values

We start by referencing the `part` , or subdomain which we are currently in. This is needed since we are using parallel computing, and this way the differenc processors can communicate with each other by sharing their outputs. Additionally, while this can be omitted, we generate files to log the progress. Also we generate a file to store the force outputs.

```
if i_am_main(parts)

    logs_path=ENV["PerforatedCylinder_LOGS"]
    log_file = run_name*"-output.log"
    full_logs_path = joinpath(logs_path,log_file)
    io = open(full_logs_path, "w")

    forces_path=ENV["PerforatedCylinder_FORCES"]
    force_file=run_name*"-forces.csv"
    full_force_path = joinpath(forces_path,force_file)
    io_force = open(full_force_path, "w")

  end

  function to_logfile(x...)
    if i_am_main(parts)
      write(io,join(x, " ")...)
      write(io,"\n")
      flush(io)
    end
  end

  function to_forcefile(x...)
    if i_am_main(parts)
      write(io_force,join(x, " ")...)
      write(io_force,"\n")
      flush(io_force)
```

```
    end
  end
```

1.2 Define the geometry

After that we call the mesh that we want, and extract all tags and boundaries. Throughout the code some key steps get pushed to the log file so that it is easier to troubleshoot the code if, or rather when some simulations were to fail.

```
starting_timestamp = time()
to_logfile("Start time = $starting_timestamp")
# Geometry
to_logfile("Geometry")
DIRICHLET_tags = ["inlet", "walls", "monopile"]
# FLUID_LABEL = "fluid"
# OUTLET_LABEL = "outlet"
meshes_path=ENV["PerforatedCylinder_MESHES"]
full_mesh_path = joinpath(meshes_path,mesh_file)
to_logfile("Mesh file: ",full_mesh_path)
model =  GmshDiscreteModel(parts,full_mesh_path)
Ω = Triangulation(model)
Ω_f = Triangulation(model, tags = "fluid")
Γ_S = Boundary(model, tags = "monopile")
Γ_out = Boundary(model, tags = "outlet")
```

1.3 Define the boundary conditions

We start by defining the model order and assigning it to the domain and its boundaries.

```
to_logfile("Measures")
order = 2
degree = 2 * order
dΩ_f = Measure(Ω_f, degree)
dΓₛ = Measure(Γ_S, degree)
dΓout = Measure(Γ_out, degree)
n_ΓS = get_normal_vector(Γ_S)
n_Γout = get_normal_vector(Γ_out)
```

In addition, we define some general physical parameters:

```
# Physics parameters
to_logfile("Parameters")
rho = 1.025e3 # kg/m^3
μ_f = 1.0e-3 # rho * Vinf * D / Re #0.01 # Fluid viscosity
v_f = μ_f / rho # kinematic viscosity
```

as well as the boundary conditions for the inflow side, walls and cylinder itself. The outflow boundary condition is immediately incorporated in the weak form. While not used here, this code allows immediate 3D expansion.

```
# Boundary conditions and external loads
dims = num_cell_dims(model)
u0(x,t,::Val{2}) = VectorValue(0.0, 0.0)
u1(x,t,::Val{2}) = VectorValue( Vinf, 0.0 )
u0(x,t,::Val{3}) = VectorValue(0.0, 0.0, 0.0)
```

```
u1(x,t,::Val{3}) = VectorValue( Vinf, 0.0, 0.0 )
u0(x,t::Real) = u0(x,t,Val(dims))
u1(x,t::Real) = u1(x,t,Val(dims))
u0(t::Real) = x -> u0(x,t,Val(dims))
u1(t::Real) = x -> u1(x,t,Val(dims))
println("Dimensions in this problem = $dims")
U0_dirichlet = [u1, u1, u0]
g(x) = 0.0
```

2 Set up the numerical scheme

2.1/2.2/2.3 Set up the test and trial spaces and combining them into a transient multifield

With the domain fully defined the next step is to define the internal structure of each cell through reference elements in their respective transient test spaces. With one for velocity (in 2 dimensions), and one for pressure (1 dimension), a multifield test space and trial space is created.
The velocity reference elements in the test spaces are Lagrangian second order elements. For numerical stability, this means that the pressure elements should be Lagrangian elements as well, but of order 1. These 2 test spaces have H1 and C0 conformity respectively. H1 conformity indicates that the test space functions are continuous, although their gradients can contain jumps. C0 conformity allows the test space to contain jumps already, which is required for the pressure field.
With all of that done, we can generate the actual trial and test spaces for transient use.

```
to_logfile("FE spaces")
# ReferenceFE
reffeᵤ = ReferenceFE(lagrangian, VectorValue{dims,Float64}, order)#,space=:P)
reffeₚ = ReferenceFE(lagrangian, Float64, order - 1)#,space=:P)

# Define test FESpaces
V = TestFESpace(Ω, reffeᵤ,  dirichlet_tags = DIRICHLET_tags, conformity = :H1)
Q = TestFESpace(Ω, reffeₚ,   conformity= :C0)
Y = MultiFieldFESpace([V, Q])

# Define trial FESpaces from Dirichlet values
U = TransientTrialFESpace(V, U0_dirichlet)
P = TrialFESpace(Q)
X = TransientMultiFieldFESpace([U, P])
```

2.4 Initialize the domain

To facilitate the calculation start-up, it is better to not start from a no-flow situation, where all velocities are 0. To have an initial state, a Stokes solver is ran once. The strong form of the steady state Stokes equation is:

$$\begin{cases} -\mu\Delta\mathbf{u} + \nabla p = \mathbf{f}, & \text{in } \Omega, \\ -\nabla\mathbf{u} = 0, & \text{in } \Omega \end{cases}$$

where $\mathbf{u}$ denotes the velocity field, 2 dimensional, and p denotes the pressure field. μ is the in-compressible fluid viscosity. Multiplying the test function $v \in \mathbf{H}_0^1(\Omega)$ to the former momentum equation on the top, and ding the same for the continuity equation on the bottom with $q \in L^2(\Omega)$, we can find the weak form problem through integration by parts. The problem then reads:

Find $\mathbf{u} \in \mathbf{H}_0^1(\Omega)$ and $p \in L^2(\Omega)$ such that

$$\begin{cases} (\mu\nabla\mathbf{u}, \nabla\mathbf{v}) - (p, \nabla v) = \langle \mathbf{f}, \mathbf{v} \rangle, & \forall v \in \mathbf{H}_0^1(\Omega) \\ -(\nabla\mathbf{u}, q) = 0, & \forall q \in L^2(\Omega) \end{cases}$$

Finally this can be combined into the bilinear form:

$$\begin{cases} a(\mathbf{u},\mathbf{v}) = \mu \int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} \mathrm{d}\mathbf{x} \\ b(\mathbf{v},\mathbf{q}) = -\int_\Omega (\nabla\mathbf{v}) q \mathrm{d}\mathbf{x} \end{cases}$$

Rewriting this such that the formulation includes the outflow boundary condition and the non-forced fluid flow inside the domain, and then rewriting the inputs so that the form is equal to the one implemented in Gridap.jl, the problem to be solved becomes:

Find $\mathbf{u} \in \mathbf{H}_0^1(\Omega)$ and $p \in L^2(\Omega)$ such that

$$a\left((u,p),(v,q)\right) = \int \left(\epsilon\left(v\right) \odot \left(\sigma_{dev_f} \cdot \epsilon\left(u\right)\right) - \left(\nabla \cdot v\right) \cdot p + q \cdot \left(\nabla \cdot u\right)\right) d\Omega_f$$

with $\sigma_{dev_f}\left(\epsilon\right) = 2 \cdot \nu_f \cdot \epsilon$ and ϵ the symmetric gradient of the field; and such that

$$l\left((v,q)\right) = \int \left(0\right) d\Omega_f$$

in which the 0 indicates a free fluid flow inside the domain.

This is done through the definition of the weak form:

```
# Stokes for pre-initalize NS
σ_dev_f(ε) = 2 * ν_f * ε #  Cauchy stress tensor for the fluid
a((u, p), (v, q)) = ∫(ε(v) ⊙ (σ_dev_f ∘ ε(u)) - (∇ ⋅ v) * p + q * (∇ ⋅ u))dΩ_f
l((v, q)) = ∫(0.0 * q)dΩ_f
stokes_op = AffineFEOperator(a,l,X(0.0),Y)
```

and the set-up of the parallel solver:

```
# Setup solver via low level PETSC API calls
function mykspsetup(ksp)
    pc       = Ref{GridapPETSc.PETSC.PC}()
    mumpsmat = Ref{GridapPETSc.PETSC.Mat}()
    @check_error_code GridapPETSc.PETSC.KSPSetType(ksp[],GridapPETSc.PETSC.KSPPREONLY)
    @check_error_code GridapPETSc.PETSC.KSPGetPC(ksp[],pc)
    @check_error_code GridapPETSc.PETSC.PCSetType(pc[],GridapPETSc.PETSC.PCLU)
    @check_error_code GridapPETSc.PETSC.PCFactorSetMatSolverType(pc[],GridapPETSc.PETSC.MATSOLVERMUMPS)
    @check_error_code GridapPETSc.PETSC.PCFactorSetUpMatSolverType(pc[])
    @check_error_code GridapPETSc.PETSC.PCFactorGetMatrix(pc[],mumpsmat)
    @check_error_code GridapPETSc.PETSC.MatMumpsSetIcntl(mumpsmat[],  4, 2)
    @check_error_code GridapPETSc.PETSC.MatMumpsSetIcntl(mumpsmat[],  7, 0)
    @check_error_code GridapPETSc.PETSC.MatMumpsSetIcntl(mumpsmat[],  14, 5000)
    @check_error_code GridapPETSc.PETSC.MatMumpsSetIcntl(mumpsmat[],  24, 1)
    @check_error_code GridapPETSc.PETSC.MatMumpsSetCntl(mumpsmat[], 3, 1.0e-10)
    @check_error_code GridapPETSc.PETSC.KSPSetFromOptions(ksp[])
end

# Linear Solver
to_logfile("Stokes solve")
ls₀ = PETScLinearSolver(mykspsetup)
u_ST, p_ST = solve(ls₀,stokes_op)
# u_ST, p_ST = solve(stokes_op)
```

Lastly this solution is interpolated over the different elements in order to be used as initial conditions for the transient calculations.

```
# initial condition NS
to_logfile("Navier-Stokes operator")
xh₀ = interpolate_everywhere([u_ST, p_ST],X(0.0))
vh₀ = interpolate_everywhere((u0(0),0.0),X(0.0))
```

2.5 Define the weak form

The problem formulation starts with the Navier-Stokes equation:

$$\begin{cases} \partial_t \mathbf{u} - \nu \Delta \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = \mathbf{f}, & in \Omega \times (0,T) \\ -\nabla \mathbf{u} = 0, & in \Omega \end{cases}$$

with ν the kinematic viscosity. Using the same space definitions as for the Stokes initialization. the weak form with a residual notation then reads:

Find $\mathbf{u} \in \mathbf{H}_0^1(\Omega)$ and $p \in L^2(\Omega)$ such that

$$(\partial_t \mathbf{u}, \mathbf{v}) + B(\mathbf{u}; [\mathbf{u}, p], [\mathbf{v}, q]) = \langle \mathbf{f}, \mathbf{v} \rangle \forall v \in \mathbf{H}_0^1(\Omega) \cap \forall q \in L^2(\Omega)$$

where $B(\mathbf{a}; [\mathbf{u}, p], [\mathbf{v}, q]) = \nu(\nabla \mathbf{u}, \nabla \mathbf{v}) + c(\mathbf{a}, \mathbf{u}, \mathbf{v}) - (p, \nabla \cdot \mathbf{v}) + (q, \nabla \cdot \mathbf{u})$.

The final step is to add a stabilizing closure model:

$$res\,(t,(u,p),(v,q)) \;=\; \int \left(\frac{\partial u}{\partial t} \cdot v + c\,(u,u,v) + \nu_f \cdot (\nabla\,(u) \odot \nabla\,(v)) - p \cdot (\nabla \cdot v) + (\nabla \cdot u) \cdot q \right) d\Omega_f \;+ \int (\tau_m \cdot ((\nabla\,(u) \cdot$$

This can all be elegantly, in the same formulation, be converted into Gridap code:

```
# Explicit FE functions
global ηₙₕ = interpolate(u0(0),U(0.0))
global uₙₕ = interpolate(u_ST,U(0.0))
global fv_u = zero_free_values(U(0.0))

# Stabilization Parameters
c₁ = 12.0
c₂ = 2.0
cc = 4.0
h2map = map_parts(Ω_f.trians) do trian
CellField(get_cell_measure(trian),trian)
end
h2 = DistributedCellField(h2map)
hmap = map_parts(Ω_f.trians) do trian
CellField(lazy_map(dx->dx^(1/2),get_cell_measure(trian)),trian)
end
h = DistributedCellField(hmap)
τₘ = 1/(c₁*v_f/h2 + c₂*(meas∘uₙₕ)/h)
τc = cc *(h2/(c₁*τₘ))

# Weak form
c(a,u,v) = 0.5*((∇(u)'·a)·v - u·(∇(v)'·a))
res(t,(u,p),(v,q)) = ∫( ∂t(u)·v  + c(u,u,v) + ε(v) ⊙ (σ_dev_f ∘ ε(u)) - p*(∇·v) + (∇·u)*q +
                        τₘ*((∇(u)'·u - ηₙₕ)·(∇(v)'·u)) + τc*((∇·u)*(∇·v)) )dΩ_f +
                    ∫( 0.5*(u·v)*(u·n_Γout) )dΓout
jac(t,(u,p),(du,dp),(v,q)) = ∫( c(du,u,v) + c(u,du,v) + ε(v) ⊙ (σ_dev_f ∘ ε(du)) - dp*(∇·v) + (∇·du)*q +
                                τₘ*((∇(u)'·u - ηₙₕ)·(∇(v)'·du) + (∇(du)'·u + ∇(u)'·du)·(∇(v)'·u)) +
                                τc*((∇·du)*(∇·v)) )dΩ_f +
                            ∫( 0.5*((du·v)*(u·n_Γout)+(u·v)*(du·n_Γout)) )dΓout
jac_t(t,(u,p),(dut,dpt),(v,q)) = ∫( dut·v )dΩ_f
```

```
# Orthogonal projection
aη(η,κ) = ∫( τₘ*(η⋅κ) )dΩ_f
bη(κ) = ∫( τₘ*((∇(uₙₕ)'⋅uₙₕ)⋅κ) )dΩ_f
op_proj = AffineFEOperator(aη,bη,U(0.0),V)
ls_proj = PETScLinearSolver()

# NS operator
op = TransientFEOperator(res, jac, jac_t, X, Y)
```

2.6 Set up the integration scheme

These results, $\mathbf{u}$ and p should then be integrated to the next time step. This is done through the implicit generalized alpha scheme with $\rho_\infty = 0.5$, as defined before.

```
# ODE solver
t₀ = 0.0 # start [s]
ρ∞ = 0.5
ode_solver = GeneralizedAlpha(nls,Δt,ρ∞)
```

3. Apply the nonlinear solver

While being the core of the problem, Gridap allows the solution to be executed in a lazy format with just one line of code.

```
xₜ = solve(ode_solver,op,(xh₀,vh₀),t₀,tf)
```

4. Do post processing

The processes above are compiled as a single function. While calculating each time step, the pressure and velocity are stored. The intermediate numerical artefacts u_{nh} and η_{nh} are stored too. The pressure and velocity fields are used to determine the drag and lift force on the perforated cylinder through:

$$\vec{F} = \left(\sum \int \left(\left(n_\Gamma \cdot \sigma_{dev_f} \left(\epsilon \left(u \right) \right) \right) - p \cdot n_\Gamma \right) \cdot d\Gamma_s \right) \cdot \rho$$

The force vector contains then the total drag and lift components of the cylinder. We use some relay-parameters since the parallel use of global variables gave some discrepancies in the nested loops. Additionally these calculations contain the fluid domain, so integration of pressures determines the forces as experienced by the fluid. To get the forces on the cylinder we add a minus sign.

```
# Postprocess
if i_am_main(parts)
    println("Postprocess")
    to_logfile("Postprocess")
end
global tout = 0
createpvd(parts,run_name) do pvd
    global t_out_relay = tout
    global Δtout_relay = Δtout
    global uₙₕ_relay = uₙₕ
    global ηₙₕ_relay = ηₙₕ
    for ((uh,ph),t) in xₜ
        to_logfile("Time: $t")
```

```
        to_logfile("=======================")
        Fx, Fy = -sum(∫((n_ΓS ⋅ σ_dev_f(ε(uh))) - ph * n_ΓS) * dΓₛ)
        to_forcefile(t,Fx,Fy)
        if t>t_out_relay
        # pvd[t] = createvtk(Ω,joinpath(full_vtk_path,run_name*"_$t"),cellfields=
["u"=>uh,"p"=>ph,"un"=>uₙₕ_relay,"eta_n"=>ηₙₕ_relay])
        # t_out_relay=t+Δtout_relay
        pvd[t] = createvtk(Ω,run_name*"_$t",cellfields=
["u"=>uh,"p"=>ph,"un"=>uₙₕ_relay,"eta_n"=>ηₙₕ_relay])
        t_out_relay=t+Δtout_relay
        end
        uₙₕ_relay = interpolate!(uh,fv_u,U(t))
        ηₙₕ_relay = solve(ls_proj,op_proj)
    end
end
```

Finally the post processing is finished by closing the opened log files and ending the function which was originally called from the main module.

```
if i_am_main(parts)
    close(io)
    close(io_force)
end

ending_timestamp = time()
to_logfile("Start time = $ending_timestamp")
elapsed_time = ending_timestamp - starting_timestamp
to_logfile("elapsed time = $elapsed_time seconds")

return nothing
```

B

SIAM 2023 Intermediate report

Machine-learning based methods to accelerate design optimisation of perforated monopiles

Ruben Dekeyser

Delft University of Technology, Delft, The Netherlands
r.dekeyser@student.tudelft.nl

Keywords: Perforated wind turbine monopiles, Machine learning, Random Forests, Neural Network, Gaussian Processes, Data Analysis

Abstract. To help the energy transition, wind turbine manufacturers are venturing into deeper waters. Reducing the hydrodynamic loads on the foundation monopiles could be achieved by perforating the structures. For this, novel CFD calculation schemes are needed, which can be sped up through Machine Learning. Here a Random Forest, Gaussian Process Regression, and Neural Network are trained to predict the outcomes of 487 samples generated using Gridap.jl. It is found that Gaussian Processes outperform their alternatives by a factor of 10 for the most critical parameters, namely the prediction of average and statistically maximum drag force on the cylinder, based on the number of perforations n and porosity β of the monopile.

1 Introduction

The requirement for an ever-greener energy mix has the offshore wind industry venturing into deeper waters. This comes with great technical challenges. Chief among these are the enormous hydrodynamic loads from currents and waves in the splash zone of monopile foundations. Earlier work by Anderson (2017) and others at TU Delft have shown promising results for reducing the loads using perforated monopiles.
In this intermediate thesis update we give an initial representation of the possibilities that machine learning offers. Three regression models (Random Forests RF, Gaussian Process Regression GPR, and Neural Networks NN) are fitted to a 500-sample dataset created by Dr. O. Colomés at TU Delft. This update document is part of the process of the MSc thesis of R. Dekeyser. The ultimate goal of this thesis, supervised by Dr. Colomés, is to apply the Julia based FEM solver Gridap.jl to solve transient flow around a perforated cylinder using LES. In post-processing, Gridap.jl data generation can be sped up using machine learning (ML). This intermediate update explores the ML options.

2 Data analysis

The generated data consists of 500 samples of varying numbers of perforations n (5 to 30) and porosity values β (0.3 to 0.68). The output files contain time traces for the pressure integrated drag and lift force on the cylinder.
12 simulations failed to converge, meaning that there were no outputs. 1 simulation can be considered an extreme outlier (order of 6 magnitude difference) and is therefore discarded. This results in 487 time traces.
A Fast Fourier Transform (FFT) on the time traces, excluding start-up and shutdown effects (using 80-180 of the 200 provided seconds) provides the most interesting basis for data analysis. The average value, standard deviation, statistical absolute maximum (99% double sided quantile) and the three most dominant frequencies are extracted. These 12 outputs are to be analysed. the results are shown in Figure 1 and Figure 2.
Initial data analysis (all sets are added in section B) has revealed the following key findings:

- For drag the average value and statistical maximum follow expectations within a narrow noise band. Lower porosity results in higher loads (more negative). Higher numbers of perforations give a more random nature to the flow, resulting in slightly lower drag numbers but also much lower standard deviations. This is because the small turbulence eddies are averaged out over the whole cylinder, where small numbers of perforations or low porosity values give larger eddies which cannot be averaged out.
- Lift for a symmetrical profile should average out at 0. This is the case, especially for high n. Lower numbers of perforations generate larger eddies, giving more variation in the lift over time. This in analogous to the averaging-out property in the drag discussion.

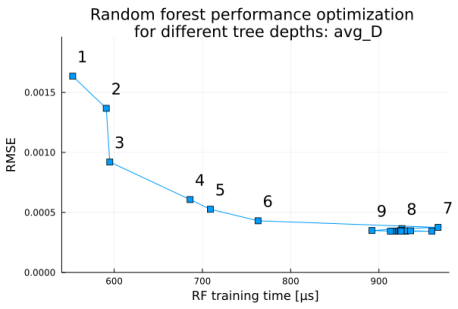
– Both drag and lift frequencies follow similar trends. This is best seen in Figure 6, 7 end 8 rather than in the appended contour plots. Low n and high β result in some higher frequency forcing. The relatively large eddies in these cases carry enough energy to influence the full monopile. As n increases the lift and drag traces become more and more constant. This gives in a large decrease of the associated forcing frequencies. Generally the frequency data has a much higher variance than the averaged forces.

While not the focus of this intermediate machine learning update, it is promising to see the (absolute) drag coefficient values between 0.06 and 0.17, since full cylinders would theoretically give values of around 0.2.

3 Machine learning models

In total three machine learning models have been trained on the 487 samples. While Random Forests can be trained for multi-output regression, this was omitted here in order to get accurate performance comparisons to the, by default single output, Gaussian Processes (Kriging) Regression. However, successful tests with Random Forests have been done on this data. Neural Networks can be trained for multi-outputs as well, but current attempts on this dataset were unsuccessful. The regression results did not converge and they were inaccurate. The current assumption links this to the small amount of samples for training a multi-layer system with 12 outputs. All models use a train-test split ratio of 0.8-0.2.

The RF model has been tested using multiple layer depths. The implementation is based on the Python-originated ScikitLearn.jl package. When weighed linearly against the RMSE accuracy ($Badness = TrainingTime \cdot RMSE$, so lower = better) one finds that 9 levels within the forest are optimal. Further increases in accuracy are not guaranteed and only increase the required training time. This is visualized below.



The GPR model training as laid out by GaussianProcesses.jl has an internal hyperparameter optimization looking into bounds for the mean estimates and kernel functions. The limited time requirement for full optimization indicates further design of the optimization scheme is not needed.

The Neural Network used here has been made in order to optimize results for the average drag and statistic maximal drag, as these outputs are seen as most critical in later wind turbine design schemes. The inputs and outputs of the model are normalized, as this improved result accuracy. For RF and GPR this was not the case. A model with 1 hidden layer using 2 input values (n and β) predicts the single output through:

Chain(Dense(2, n_hidden, relu),
Dense(n_hidden, 1)

In order to allow negative results the end layer is taken linearly. The NN uses an MSE loss function, even though generally MAE is considered better for outlier-prone CFD results (Eichinger et al., 2021). In this case however lower RMSE values were obtained using MSE in the optimization process. The learning rate reduces from the initial value with each epoch through the log10 value of the current epoch.
The hyper-parameter optimization is done for four parameters:

1. the initial learning rate, for which optimally 0.025 is found;
2. the model accuracy cutoff, for which 10e-6 is found;
3. the minimum number of training epochs, for which 100 is found;
4. the number of neurons in the hidden layer, for which 60 is found.

The model accuracy cutoff ensures continuous improvement of the model, meaning that the training is stopped when the loss function change between two consecutive epochs is below this cutoff value, and the number of performed epochs is higher than the minimum number of epochs. There are thus two conditions to accept the trained values.

	RF			**GPR**			**NN**		
	RMSE	Δt	Bad.	RMSE	Δt	Bad.	RMSE	Δt	Bad.
avg_D	0.008827	19673	174	0.004414	2617	12	0.014345	9136	131
avg_L	0.0684	20434	1398	0.063757	2582	164	0.123585	25899	3201
std_D	0.009934	19807	197	0.014637	10926	160	0.025229	11262	284
std_L	0.026339	17905	472	0.018457	1583	29	0.020262	15667	317
Max_D	0.009546	18970	181	0.010089	3206	32	0.013232	9927	131
Max_L	0.037633	14245	536	0.044118	7282	321	0.04846	17835	864
f1_D	0.480	14809	1708	0.411	2085	857	0.421	13290	5595
f1_L	0.143	13623	1948	0.639	1749	1118	0.580	7357	4267
f2_D	0.405	14961	6059	0.341	2366	806	0.346	8798	3044
f2_L	0.190	19097	3628	0.447	1240	554	0.497	9032	4489
f3_D	0.529	16525	8597	0.216	2461	532	0.384	17769	6823
f3_L	0.457	13065	5971	0.289	1330	384	0.477	6897	3290

4 Machine learning evaluation

While mainly optimized for the drag mean and statistical maximum predictions, the ideal machine learning model should be able to capture all 12 outputs as accurately as possible. From the data above, with the time given in milliseconds for the training of the model using train values and the prediction using the test values, the following interesting conclusions can be extracted:

- The badness between the 12 different output variables is hard to compare because the RMSE is scaled due to the different original values. The large RMSE for the frequency data is a consequence of the huge randomness and small correlation within the data.
- The Neural network seems to perform on par with the RF and GPR models in terms of RMSE for all frequency related outputs. The force focused output accuracy differs by a factor 2. However is most likely a consequence of the use or disuse of normalized data to train the models. The non-normalized RF and GPR inputs for the frequency matters all lie within 0 and around 1.7, so normalization is less useful than for the force related quantities between 0.05 and 0.2 approximately.
- Visually all fits look equally good. This explains the worse RMSE for almost all output types, as a relatively random point cloud RMSE is hard to predict from visul inspection. We see that the less correlated original outputs have RMSE values which are much closer to each other still. This is because in this case the fit quality through the uncorrelated data cloud is not very dependent on the model used. All trend seeking models perform equally bad (by principle).

The most important conclusion to be drawn however is that GPR is outperforming its rivals on all fronts. Its badness is on average a factor 10 lower than the second best model for that specific output prediction. While having RMSE values similar to RF and NN, the training and prediction times Δt are much better. While this may partially be a consequence of the more elaborate, built in optimization of the hyperparameters, rather than the manual selection done for the RF and NN model, the main reason is most likely the straightforward and simple nature of the model. Its robustness has been shown in literature, both in general as for CFD applications (Shen et al., 2022).

5 Conclusions and future work

To conclude this brief progress and data analysis overview we highlight three main observations:

1. The data shows expected values and trends. The trends can generally be explained through flow field patterns and turbulence of the water itself. An FFT extracts the most important parameters. the main focus lies on the average and maximum drag.
2. The comparison of three models shows clear effects of their hyperparameter optimization. The parameters optimized for the avg_D data perform well for the other 11 outputs.
3. GPR gives the all-round best results, both in terms of accuracy as well as required training and prediction time. Some sources indicate that this effect can be attributed to the simplicity and robustness of the method, while it can also be a consequence of software package included optimization functions rather than manual schemes.

Going further, the work outline can be split into three parts. Within the machine learning aspect of this thesis further training optimization should be researched, especially on the architecture optimization of the Neural Network. Additional models can be looked into as well. Within the full thesis scope the expansion of the dataset using variations of the CFD and LES calculation scheme with validation and stability testing has priority. Possible added work in 3D and free surface modelling, FSI and fatigue come second. Lastly design optimization of the perforation layout must be performed, and different industry design rules should be validated for perforated monopile use, such as p-y-curves for XXL monopiles, installation and manufacturing issues, and economic aspects.

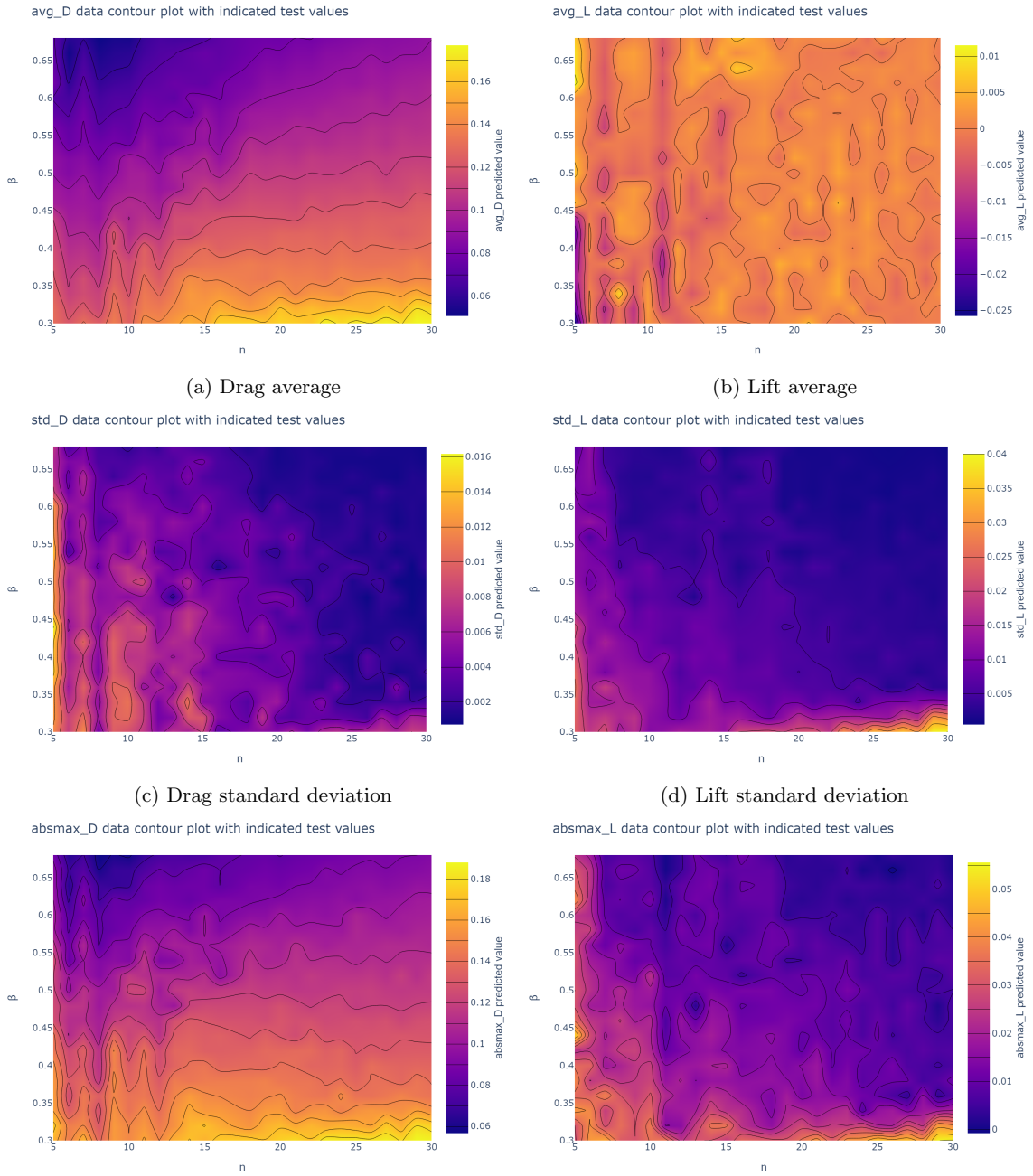
References

Anderson, M. C. (2017). *The hybrid monopile: Design of a novel foundation structure for large offshore wind turbines in intermediate water depths* (Master's thesis). Delft University of Technology.

Eichinger, M., Heinlein, A., & Klawonn, A. (2021). Stationary flow predictions using convolutional neural networks. In F. J. Vermolen & C. Vuik (Eds.), *Numerical mathematics and advanced applications enumath 2019* (pp. 541–549). Springer International Publishing.

Shen, C., Yu, P., Wang, T., & Chen, N.-Z. (2022). A cfd based kriging model for predicting the impact force on the sphere bottom during the early-water entry. *Ocean Engineering*, *243*, 110304. https://doi.org/https://doi.org/10.1016/j.oceaneng.2021.110304

A Raw data contour plots



(a) Drag average

(b) Lift average

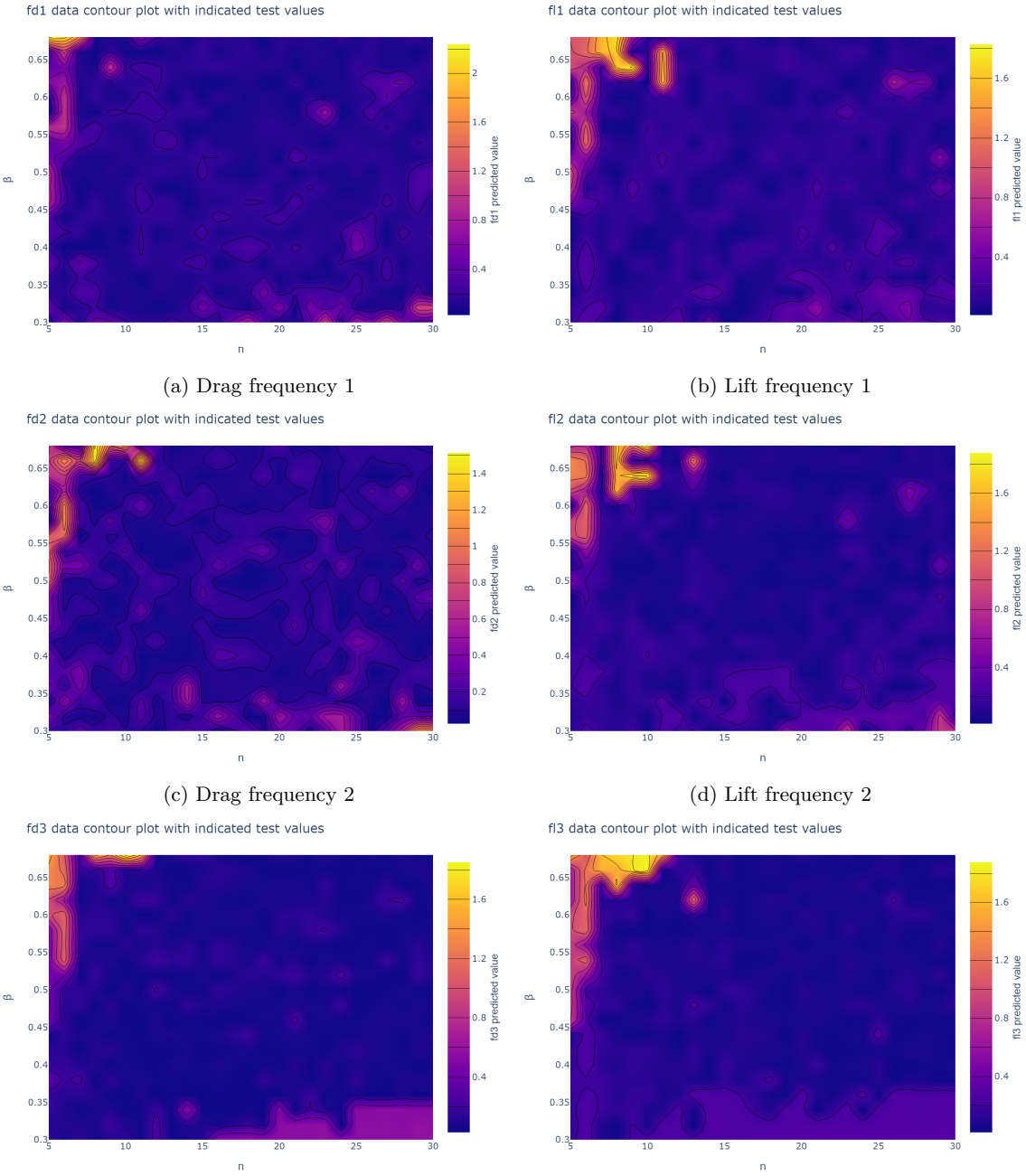
(c) Drag standard deviation

(d) Lift standard deviation

(e) Drag maximum

(f) Lift maximum

Fig. 1: Drag and lift raw data visualization part 1: force values



(a) Drag frequency 1

(b) Lift frequency 1

(c) Drag frequency 2

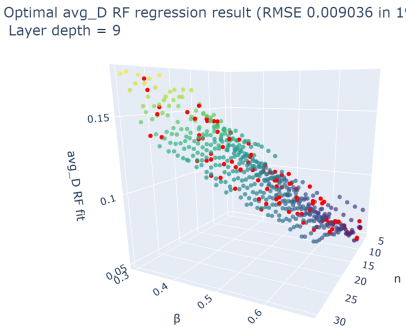
(d) Lift frequency 2

(e) Drag frequency 3

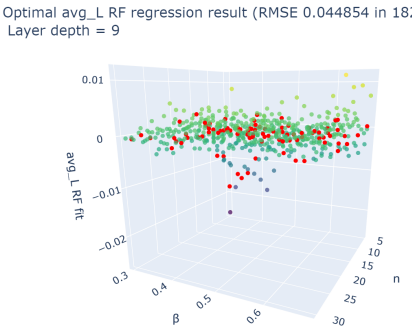
(f) Lift frequency 3

Fig. 2: Drag and lift raw data visualization part 2: frequency values

B 3D data regression results

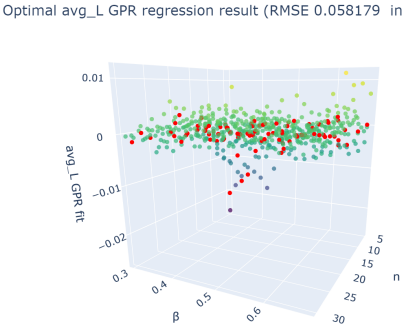


(a) RF lift

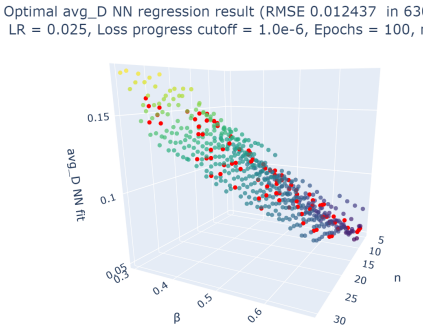


(b) RF drag

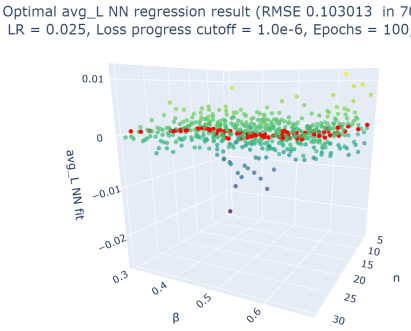
(c) GPR drag



(d) GPR drag

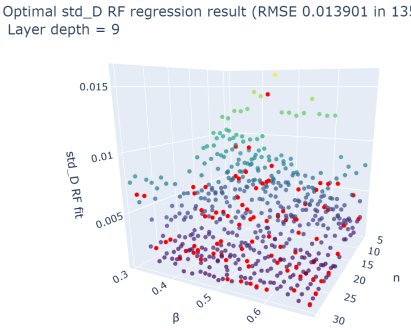


(e) NN drag

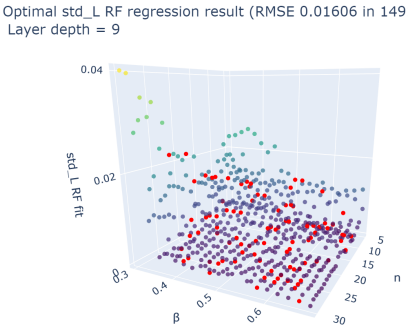


(f) NN lift

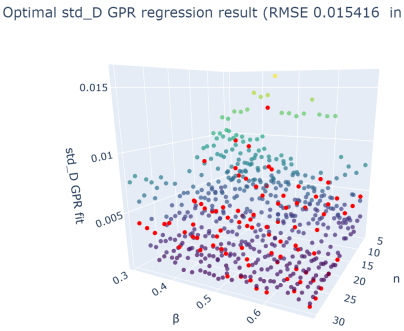
Fig. 3: Average drag and lift regression quality visualization in a 3D plot



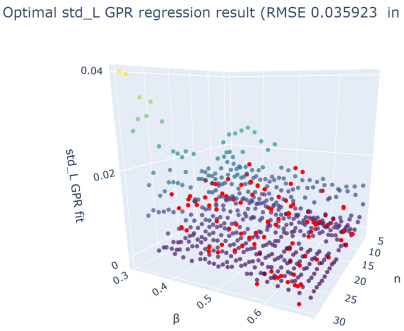
(a) RF lift



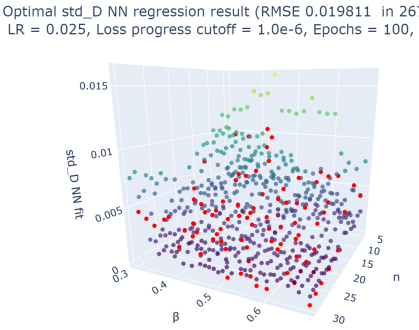
(b) RF drag



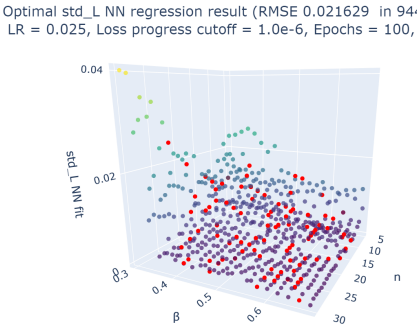
(c) GPR drag



(d) GPR drag

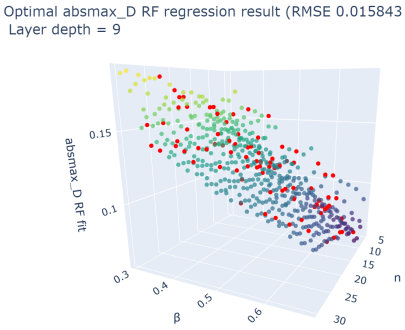


(e) NN drag

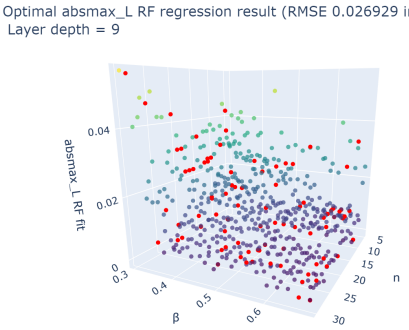


(f) NN lift

Fig. 4: Standard deviation drag and lift regression quality visualization in a 3D plot

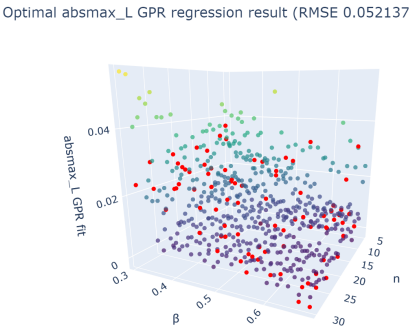


(a) RF lift

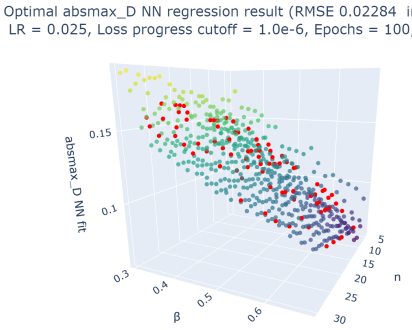


(b) RF drag

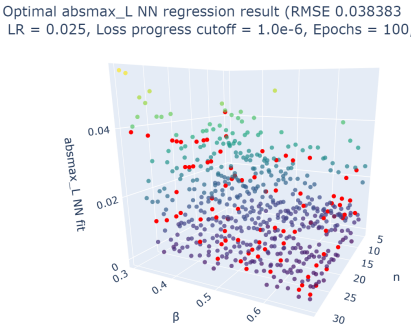
(c) GPR drag



(d) GPR drag

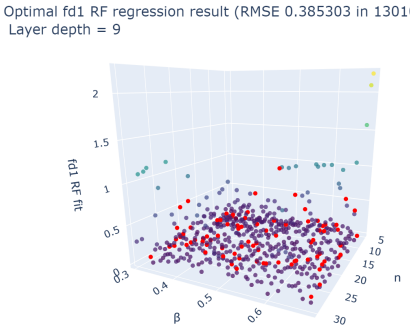


(e) NN drag

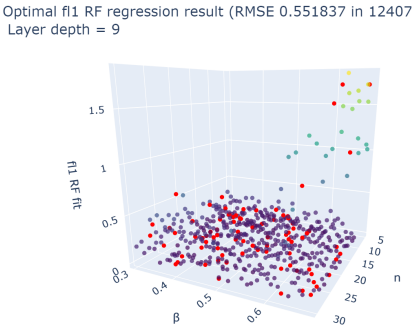


(f) NN lift

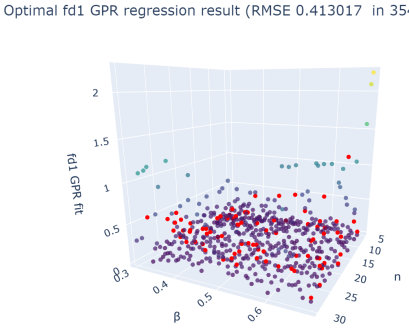
Fig. 5: Maximum absolute value (99% percentile) drag and lift regression quality visualization in a 3D plot



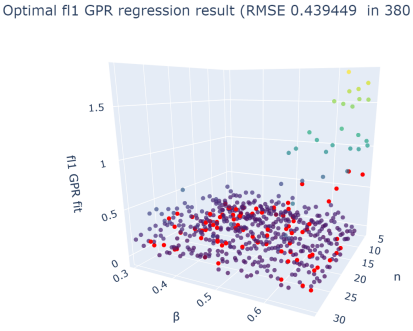
(a) RF lift



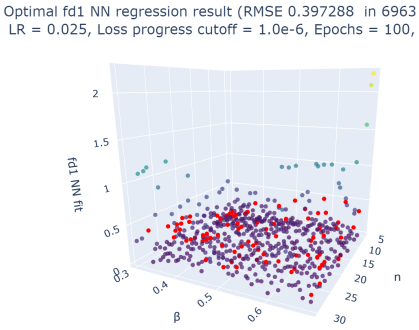
(b) RF drag



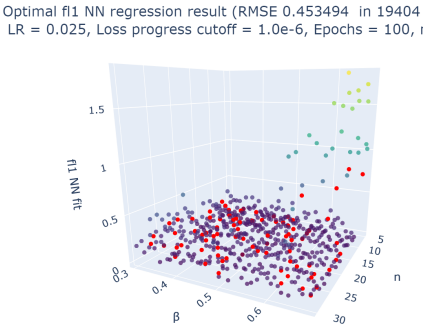
(c) GPR drag



(d) GPR drag

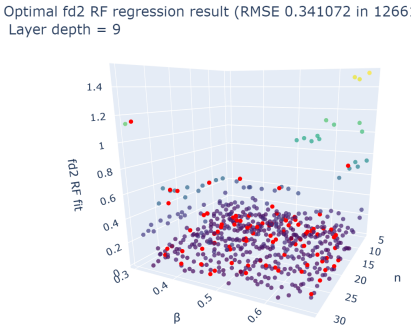


(e) NN drag

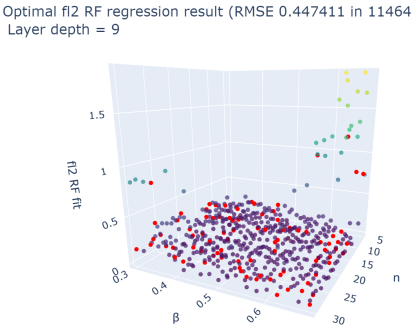


(f) NN lift

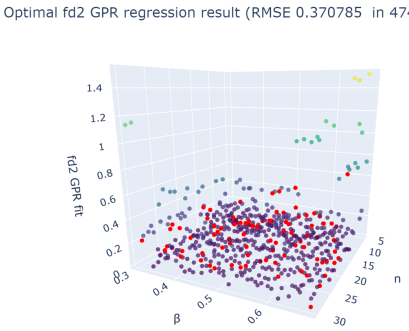
Fig. 6: First highest Fourier forcing frequency regression quality visualization in a 3D plot



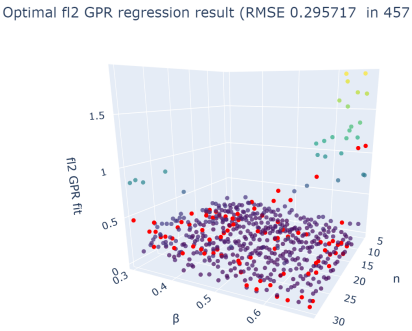
(a) RF lift



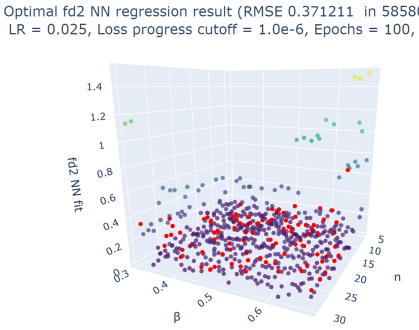
(b) RF drag



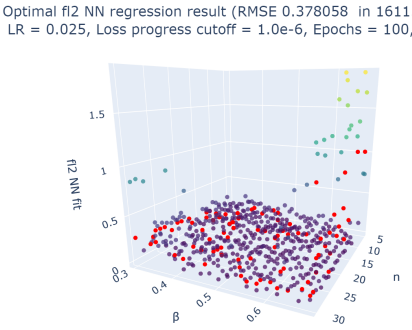
(c) GPR drag



(d) GPR drag

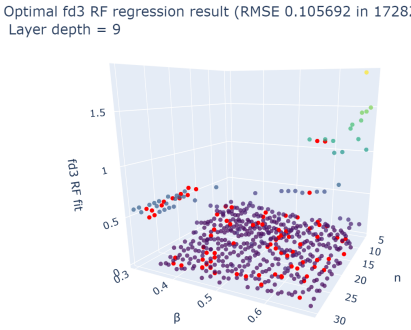


(e) NN drag

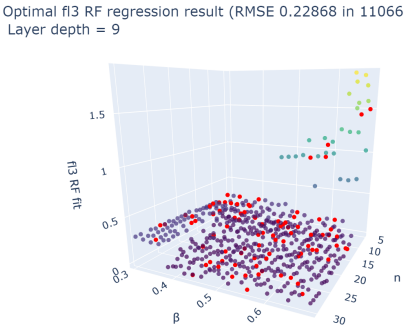


(f) NN lift

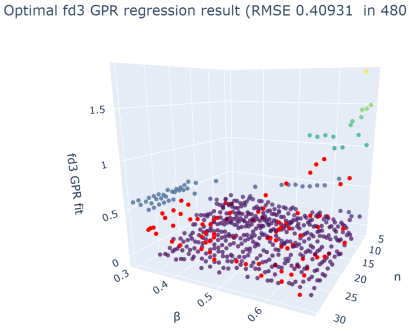
Fig. 7: Second highest Fourier forcing frequency regression quality visualization in a 3D plot



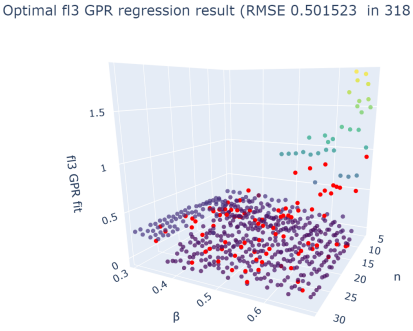
(a) RF lift



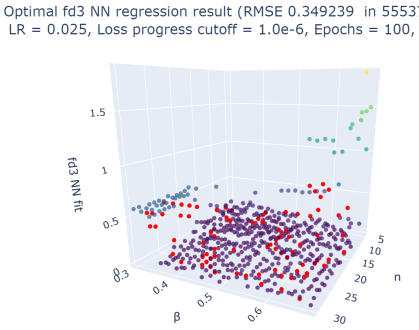
(b) RF drag



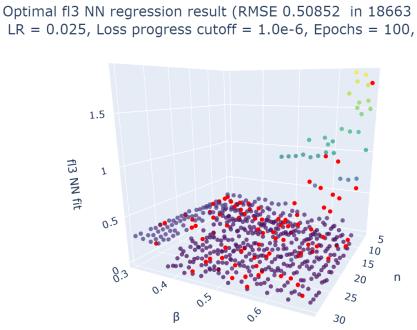
(c) GPR drag



(d) GPR drag



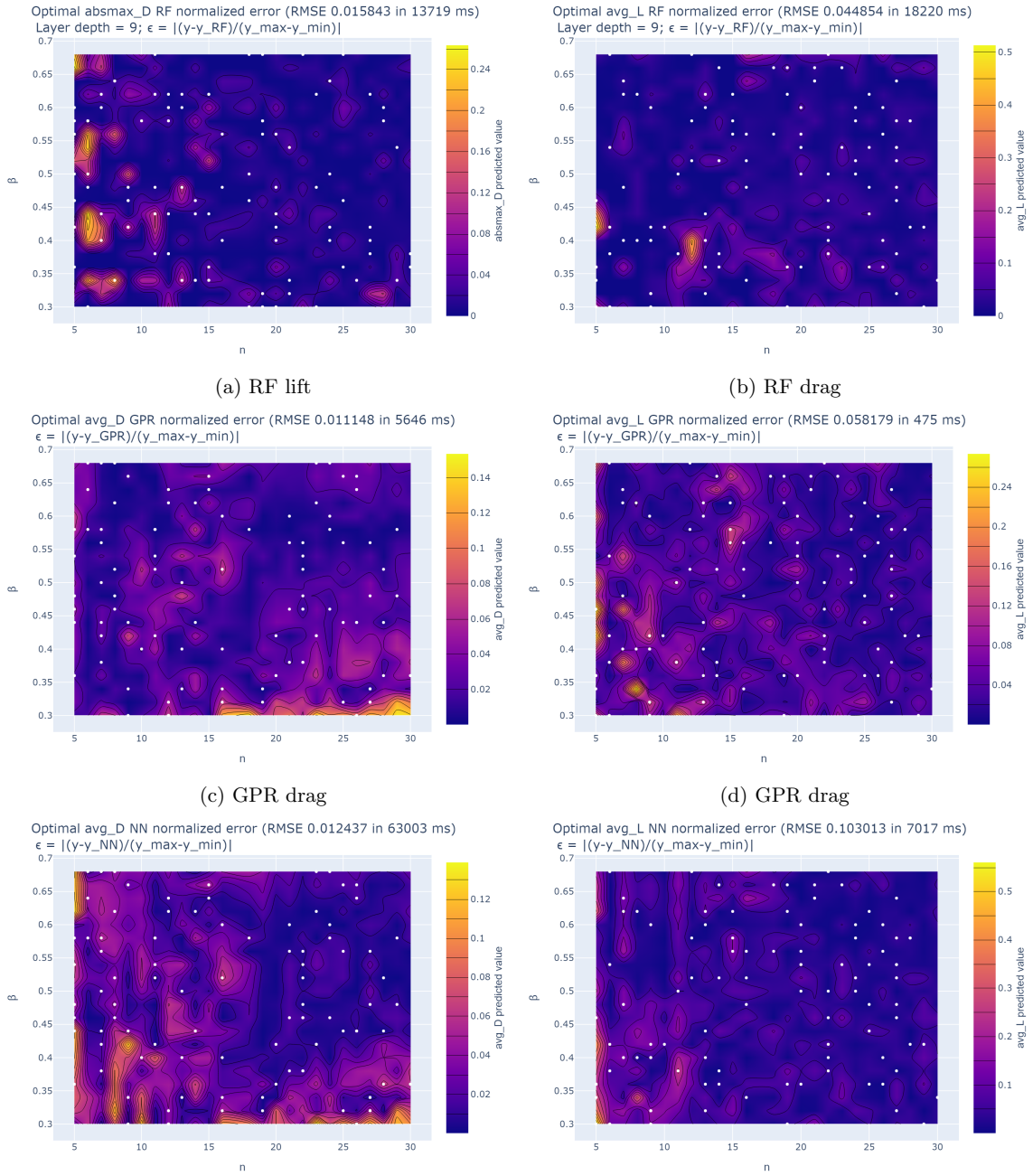
(e) NN drag



(f) NN lift

Fig. 8: Third highest Fourier forcing frequency regression quality visualization in a 3D plot

C Contour data regression errors



(a) RF lift

(b) RF drag

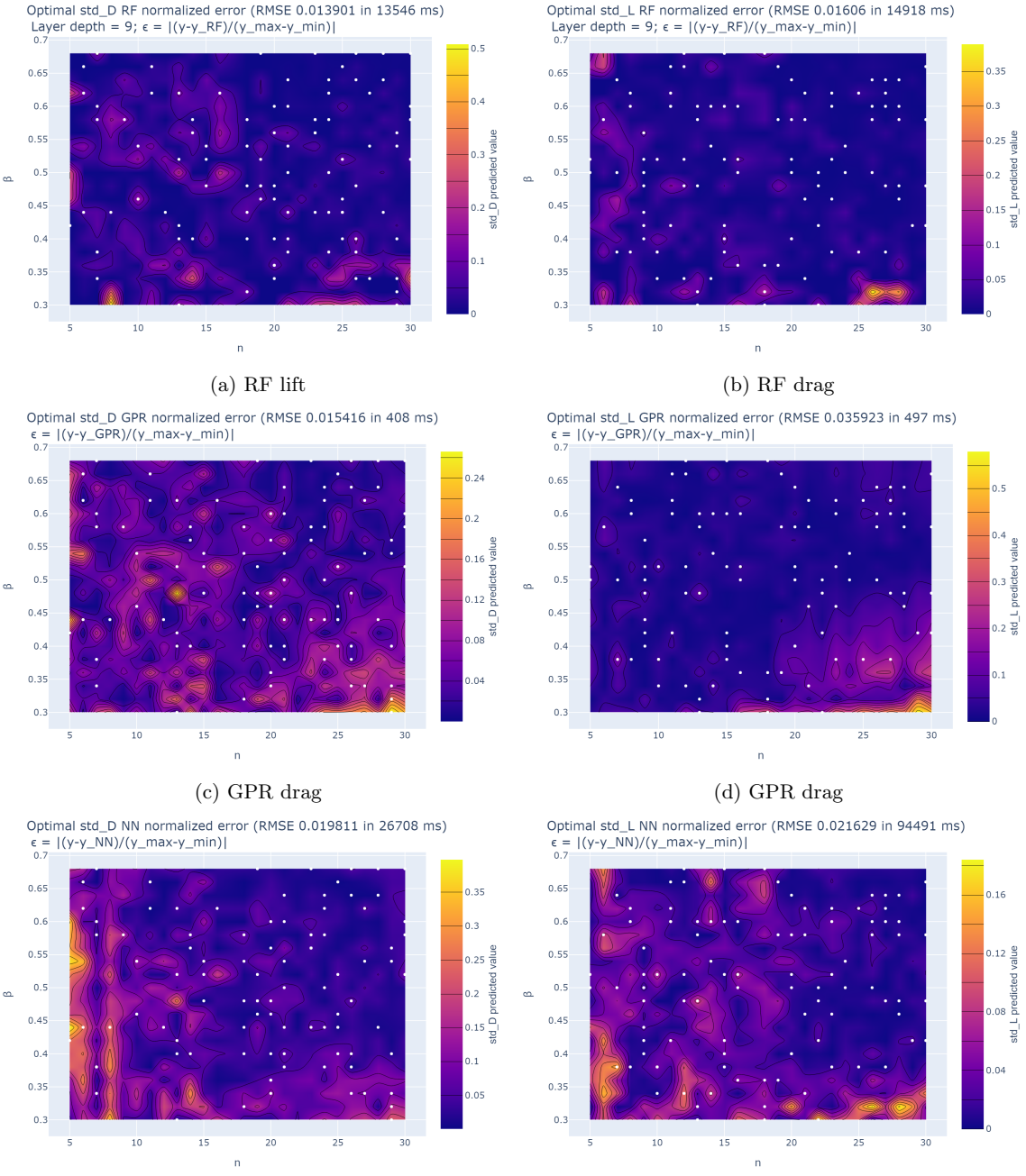
(c) GPR drag

(d) GPR drag

(e) NN drag

(f) NN lift

Fig. 9: Average drag and lift regression quality visualization in a contour plot



(a) RF lift

(b) RF drag

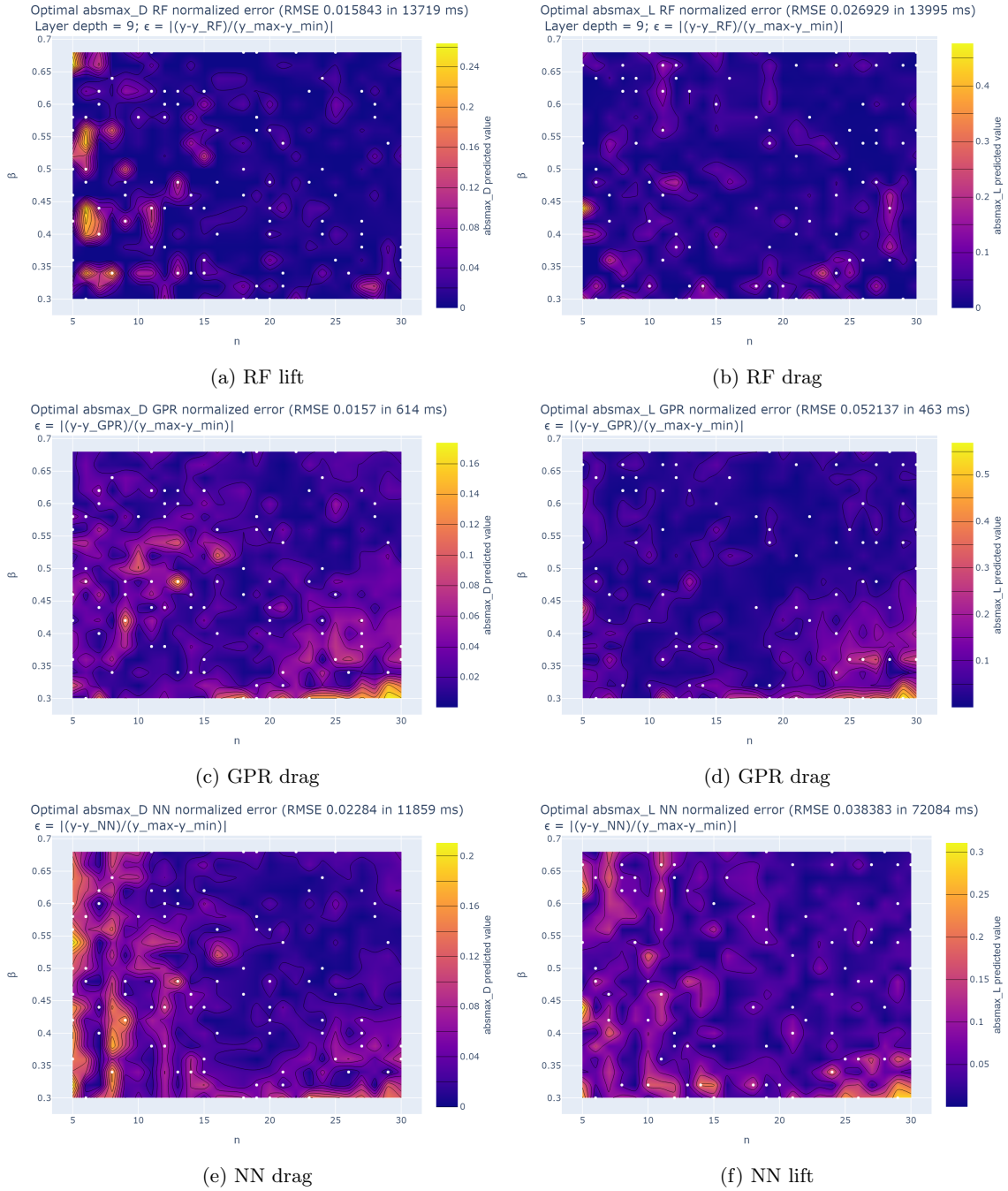
(c) GPR drag

(d) GPR drag

(e) NN drag

(f) NN lift

Fig. 10: Standard deviation drag and lift regression quality visualization in a contour plot



(a) RF lift

(b) RF drag

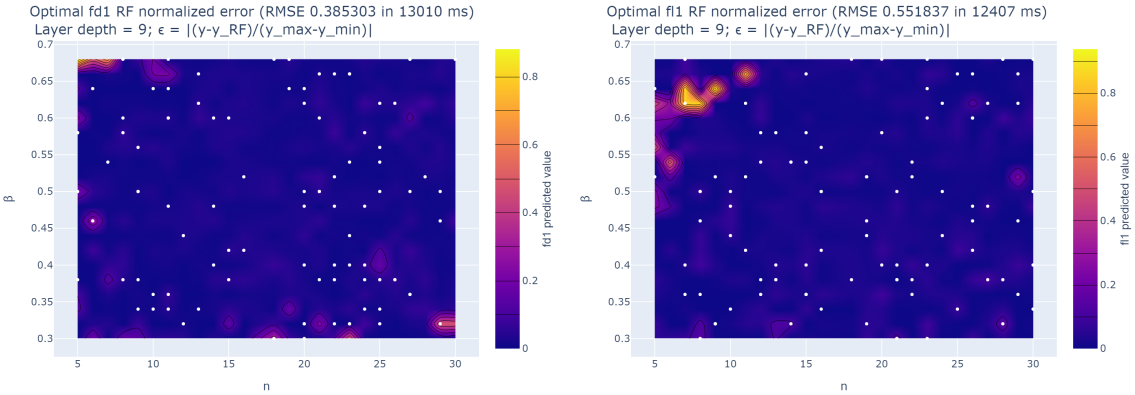
(c) GPR drag

(d) GPR drag

(e) NN drag

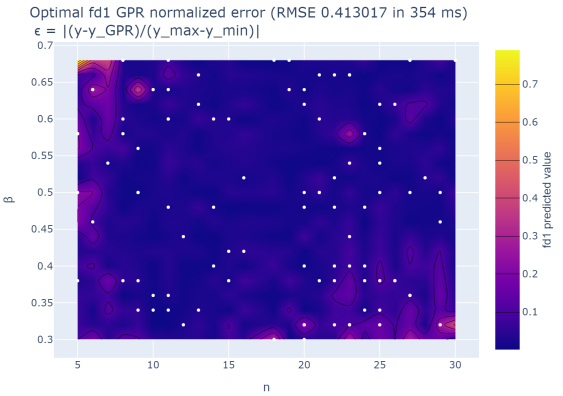
(f) NN lift

Fig. 11: Maximum absolute value (99% percentile) drag and lift regression quality visualization in a contour plot

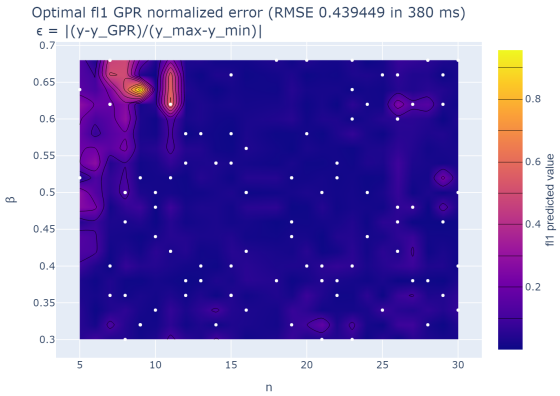


(a) RF lift

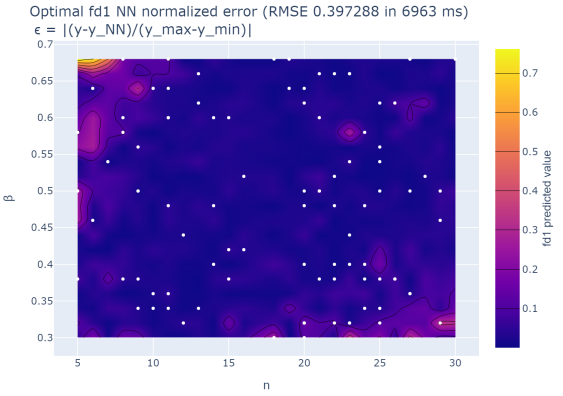
(b) RF drag



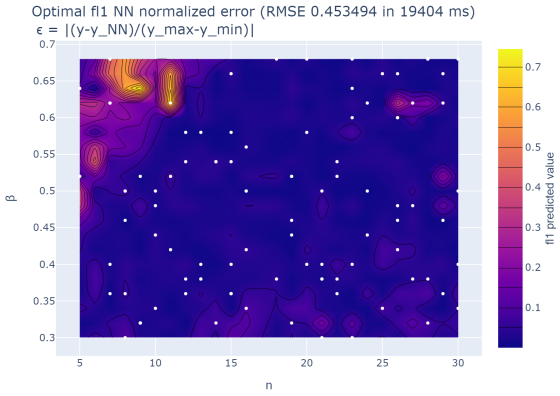
(c) GPR drag



(d) GPR drag

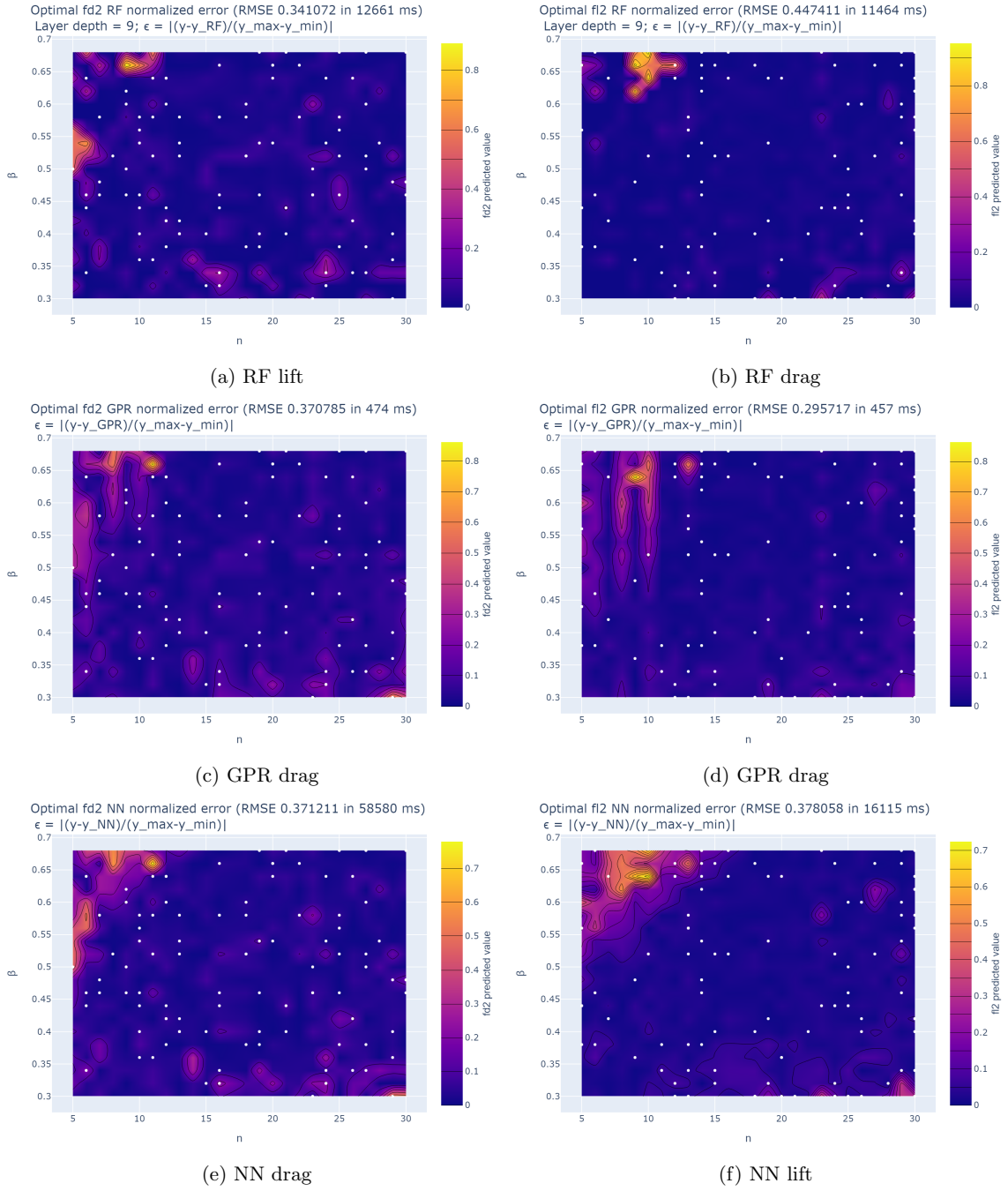


(e) NN drag



(f) NN lift

Fig. 12: First highest Fourier forcing frequency regression quality visualization in a contour plot



(a) RF lift

(b) RF drag

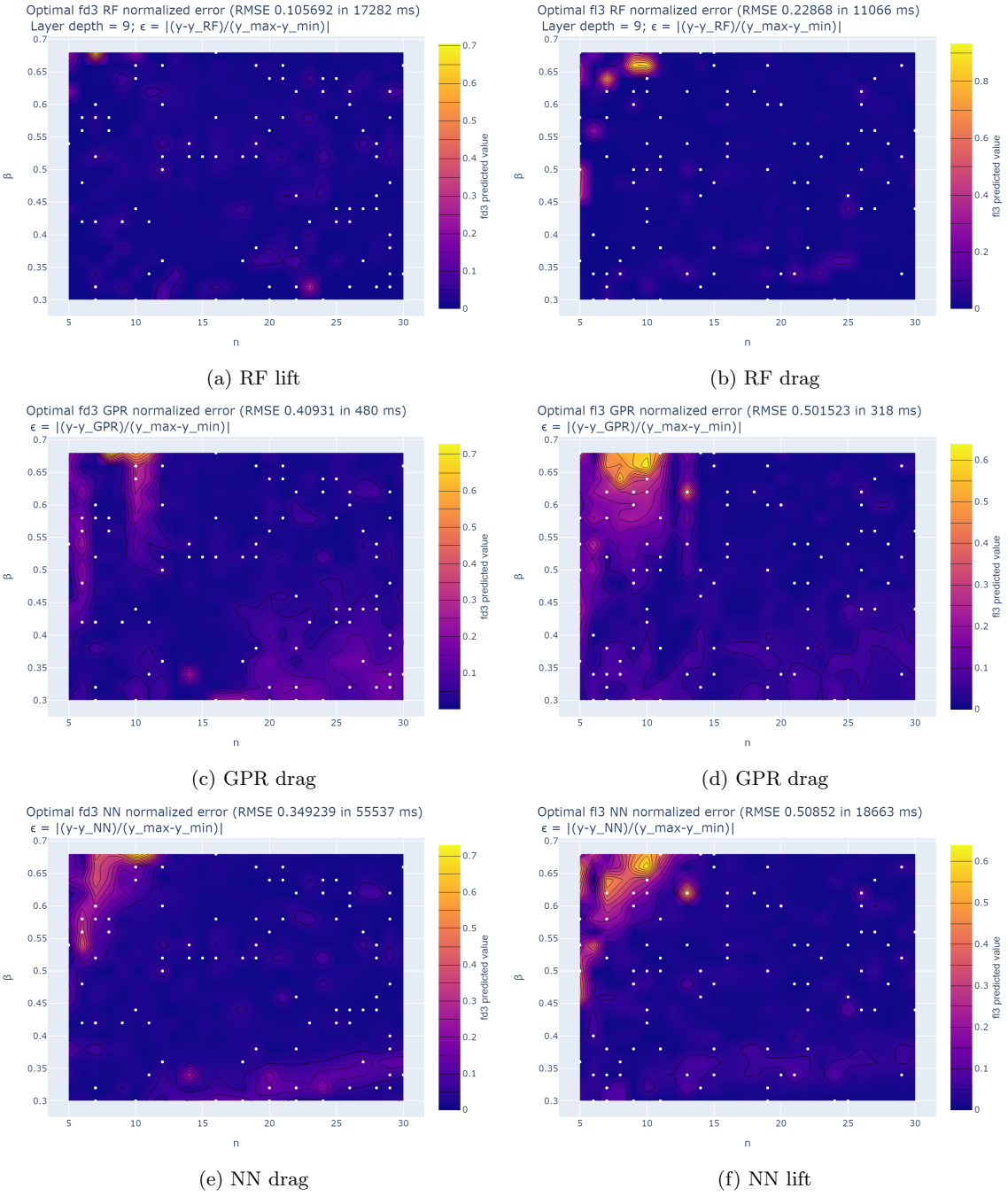
(c) GPR drag

(d) GPR drag

(e) NN drag

(f) NN lift

Fig. 13: Second highest Fourier forcing frequency regression quality visualization in a contour plot



(a) RF lift

(b) RF drag

(c) GPR drag

(d) GPR drag

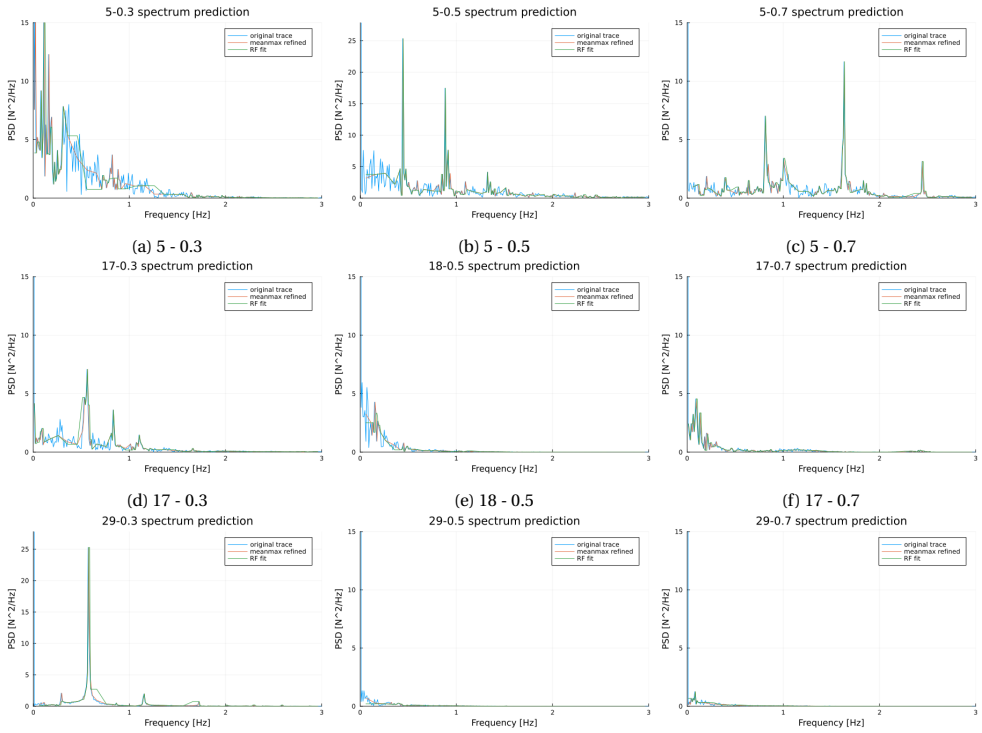
(e) NN drag

(f) NN lift

Fig. 14: Third highest Fourier forcing frequency regression quality visualization in a contour plot

C

OC data fit visual representation

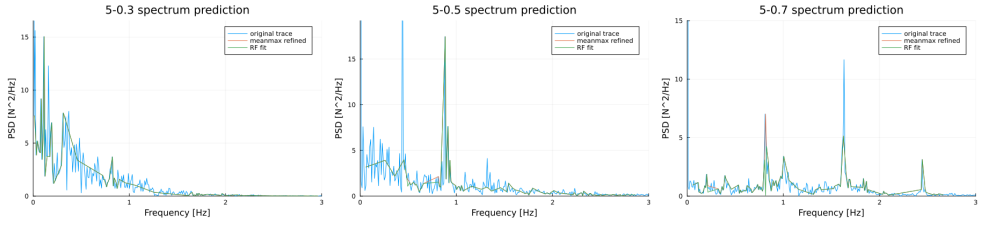


(a) 5 - 0.3 (b) 5 - 0.5 (c) 5 - 0.7

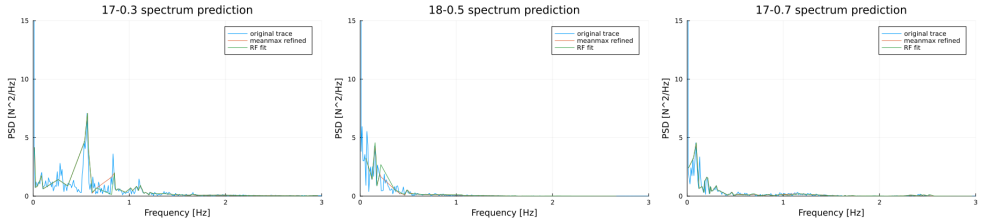
(d) 17 - 0.3 (e) 18 - 0.5 (f) 17 - 0.7

(g) 29 - 0.3 (h) 29 - 0.5 (i) 29 - 0.7

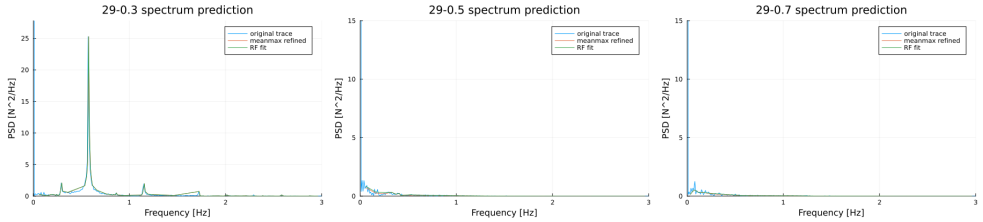
Figure C.1: Different selected fitment visualizations for the DD-RF model (OC data)



(a) 5 - 0.3 (b) 5 - 0.5 (c) 5 - 0.7

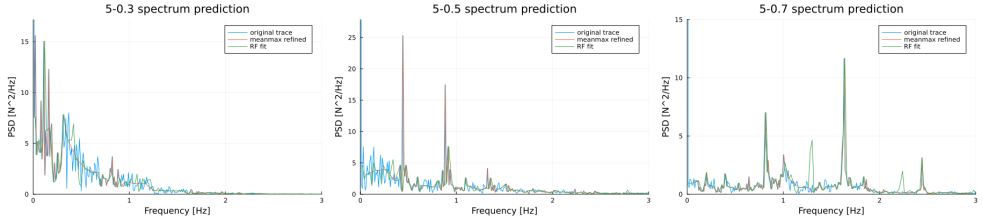


(d) 17 - 0.3 (e) 18 - 0.5 (f) 17 - 0.7

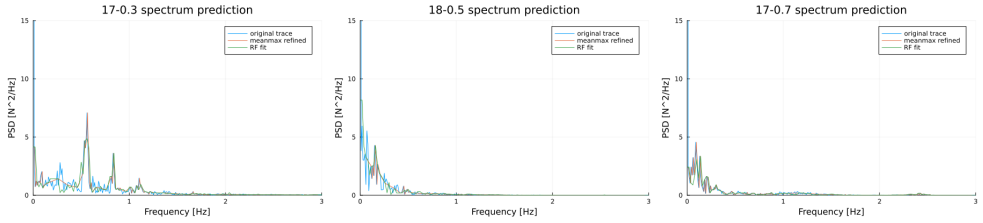


(g) 29 - 0.3 (h) 29 - 0.5 (i) 29 - 0.7

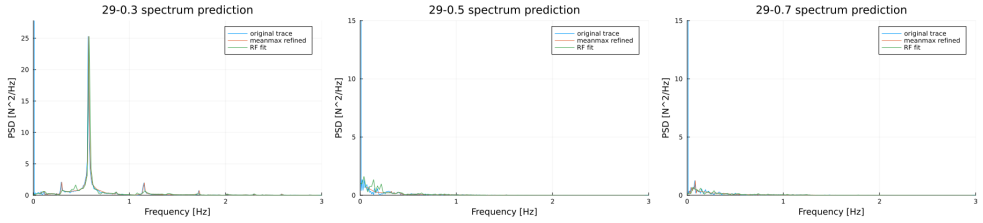
Figure C.2: Different selected fitment visualizations for the FD-RF model (OC data)



(a) 5 - 0.3 (b) 5 - 0.5 (c) 5 - 0.7

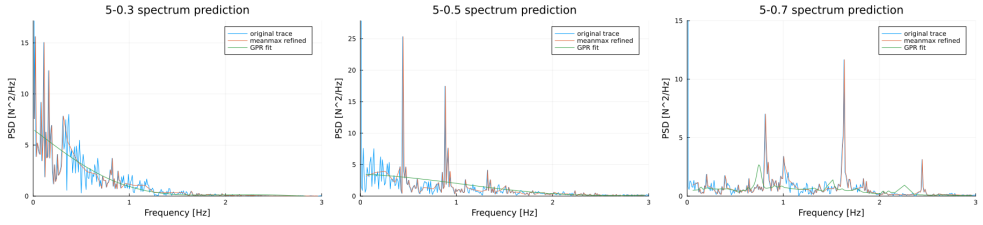


(d) 17 - 0.3 (e) 18 - 0.5 (f) 17 - 0.7

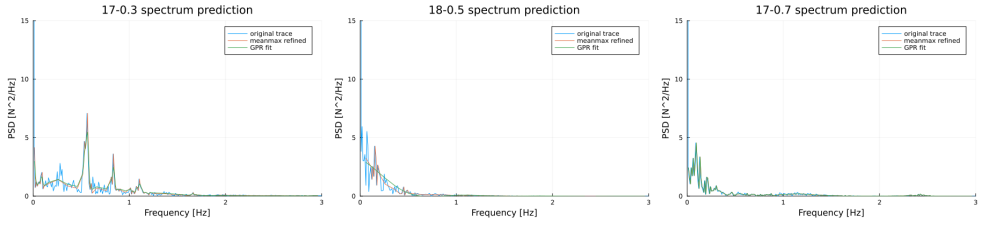


(g) 29 - 0.3 (h) 29 - 0.5 (i) 29 - 0.7

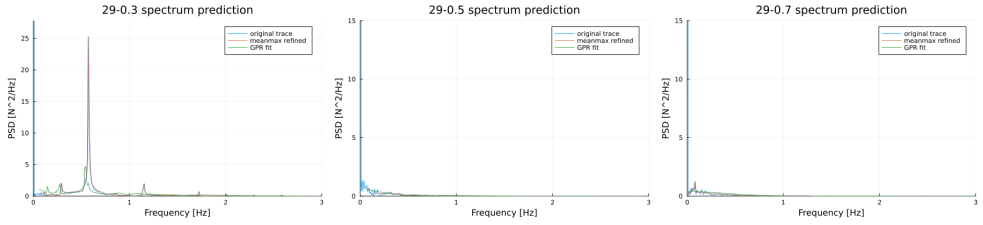
Figure C.3: Different selected fitment visualizations for the Full-RF model (OC data)



(a) 5 - 0.3 (b) 5 - 0.5 (c) 5 - 0.7

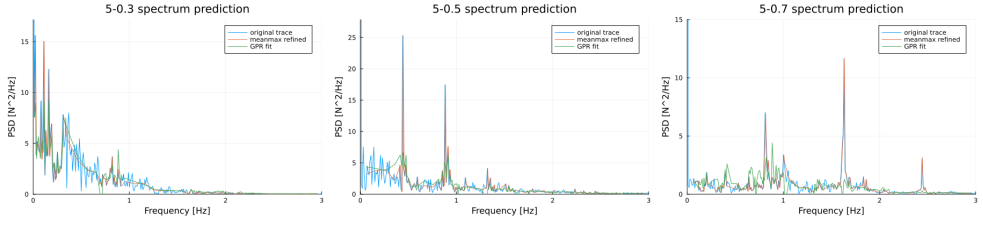


(d) 17 - 0.3 (e) 18 - 0.5 (f) 17 - 0.7

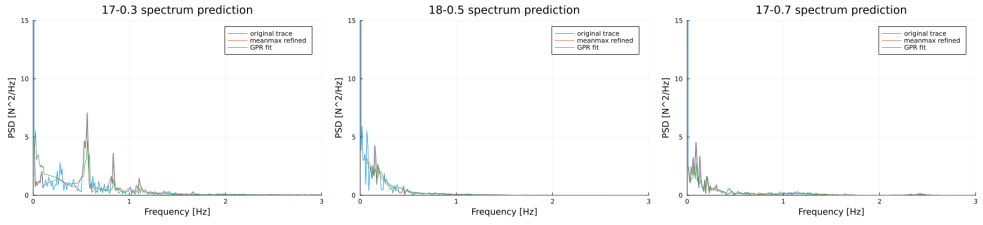


(g) 29 - 0.3 (h) 29 - 0.5 (i) 29 - 0.7

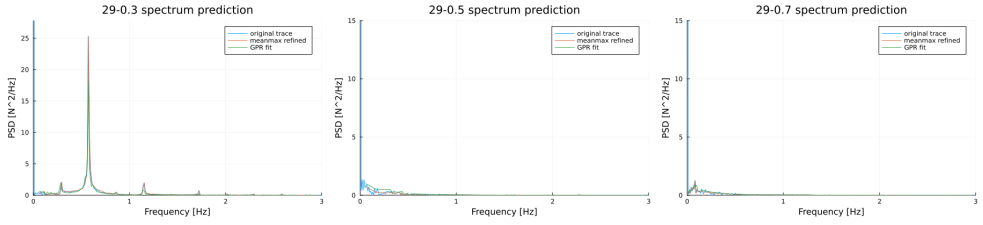
Figure C.4: Different selected fitment visualizations for the DD-GPR model (OC data)



(a) 5 - 0.3 (b) 5 - 0.5 (c) 5 - 0.7

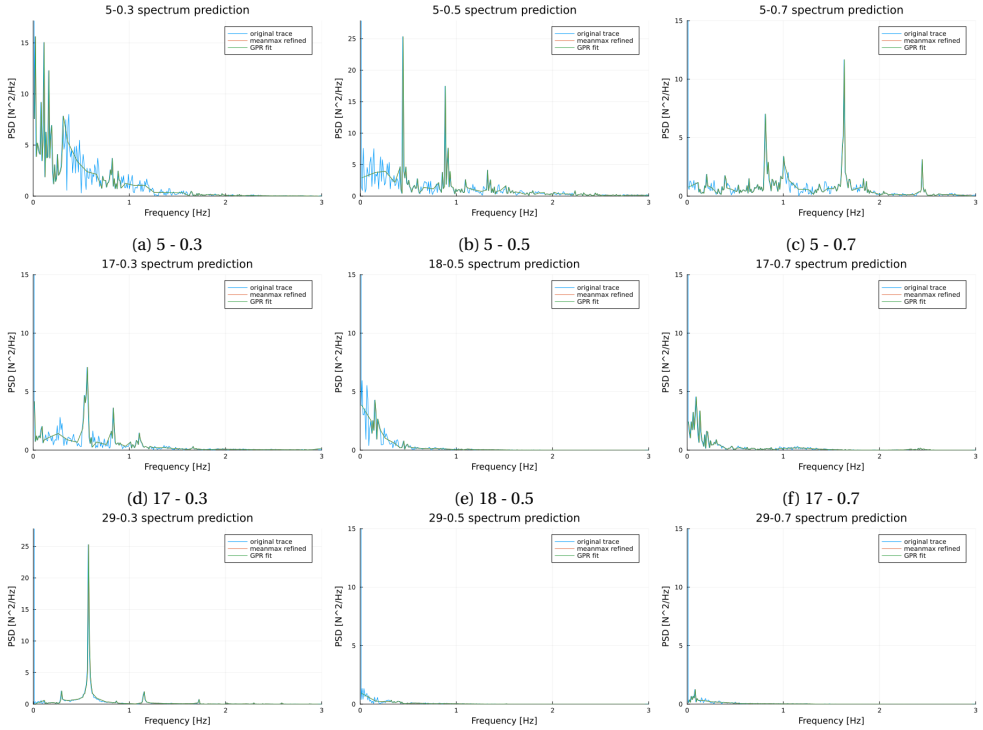


(d) 17 - 0.3 (e) 18 - 0.5 (f) 17 - 0.7



(g) 29 - 0.3 (h) 29 - 0.5 (i) 29 - 0.7

Figure C.5: Different selected fitment visualizations for the FD-GPR model (OC data)

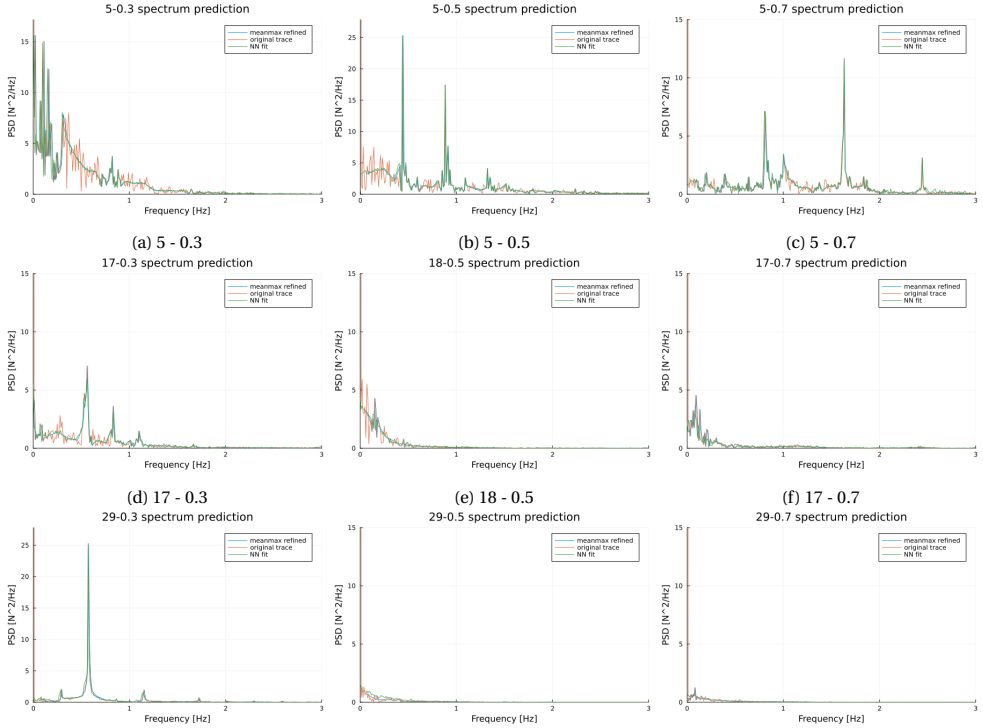


(a) 5 - 0.3 (b) 5 - 0.5 (c) 5 - 0.7

(d) 17 - 0.3 (e) 18 - 0.5 (f) 17 - 0.7

(g) 29 - 0.3 (h) 29 - 0.5 (i) 29 - 0.7

Figure C.6: Different selected fitment visualizations for the Full-GPR model (OC data)

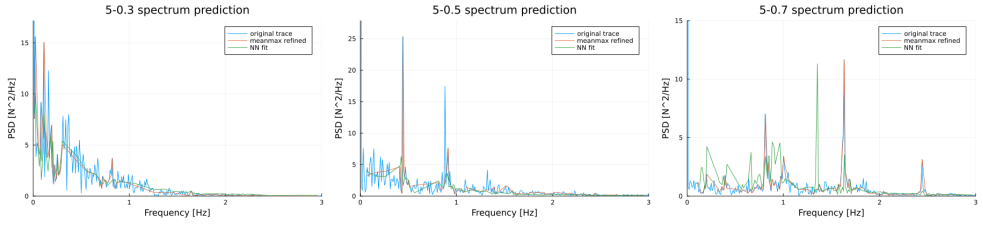


(a) 5 - 0.3 (b) 5 - 0.5 (c) 5 - 0.7

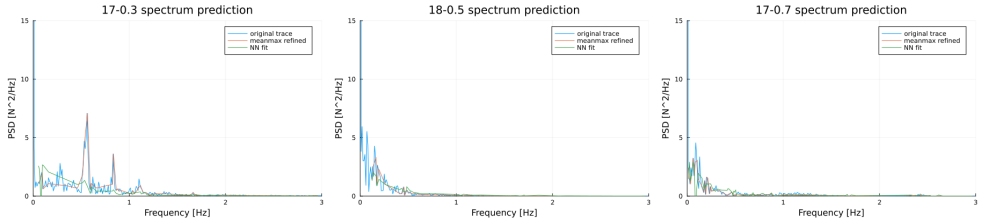
(d) 17 - 0.3 (e) 18 - 0.5 (f) 17 - 0.7

(g) 29 - 0.3 (h) 29 - 0.5 (i) 29 - 0.7

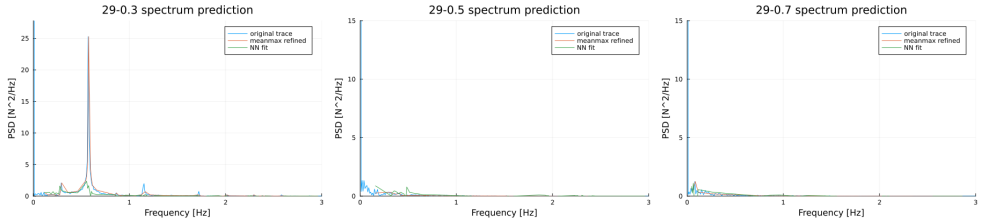
Figure C.7: Different selected fitment visualizations for the DD-NN model (OC data)



(a) 5 - 0.3 (b) 5 - 0.5 (c) 5 - 0.7

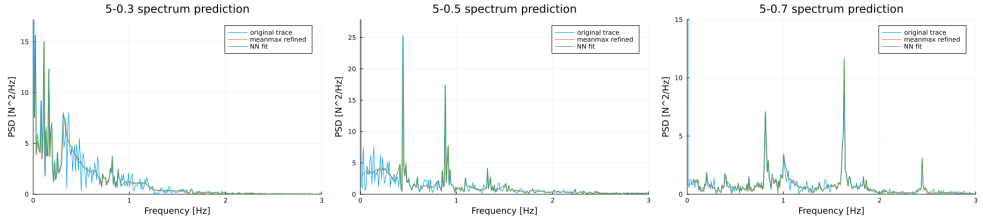


(d) 17 - 0.3 (e) 18 - 0.5 (f) 17 - 0.7

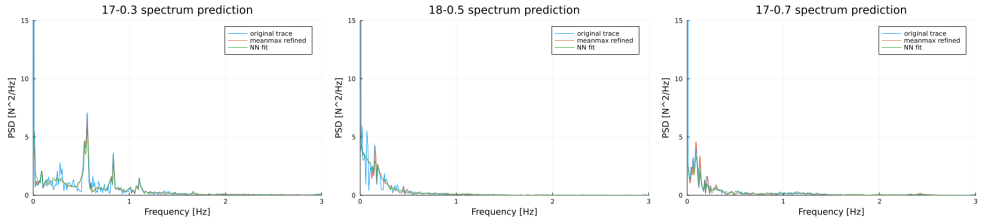


(g) 29 - 0.3 (h) 29 - 0.5 (i) 29 - 0.7

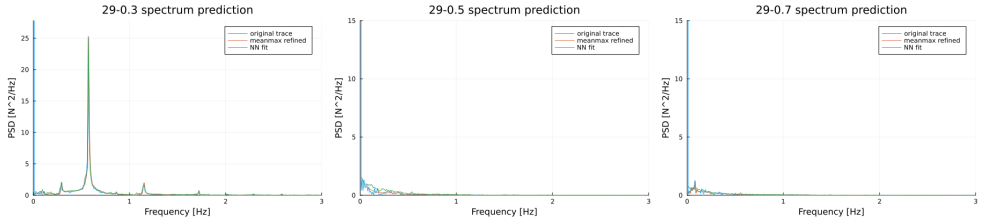
Figure C.8: Different selected fitment visualizations for the FD-NN model (OC data)



(a) 5 - 0.3 (b) 5 - 0.5 (c) 5 - 0.7



(d) 17 - 0.3 (e) 18 - 0.5 (f) 17 - 0.7



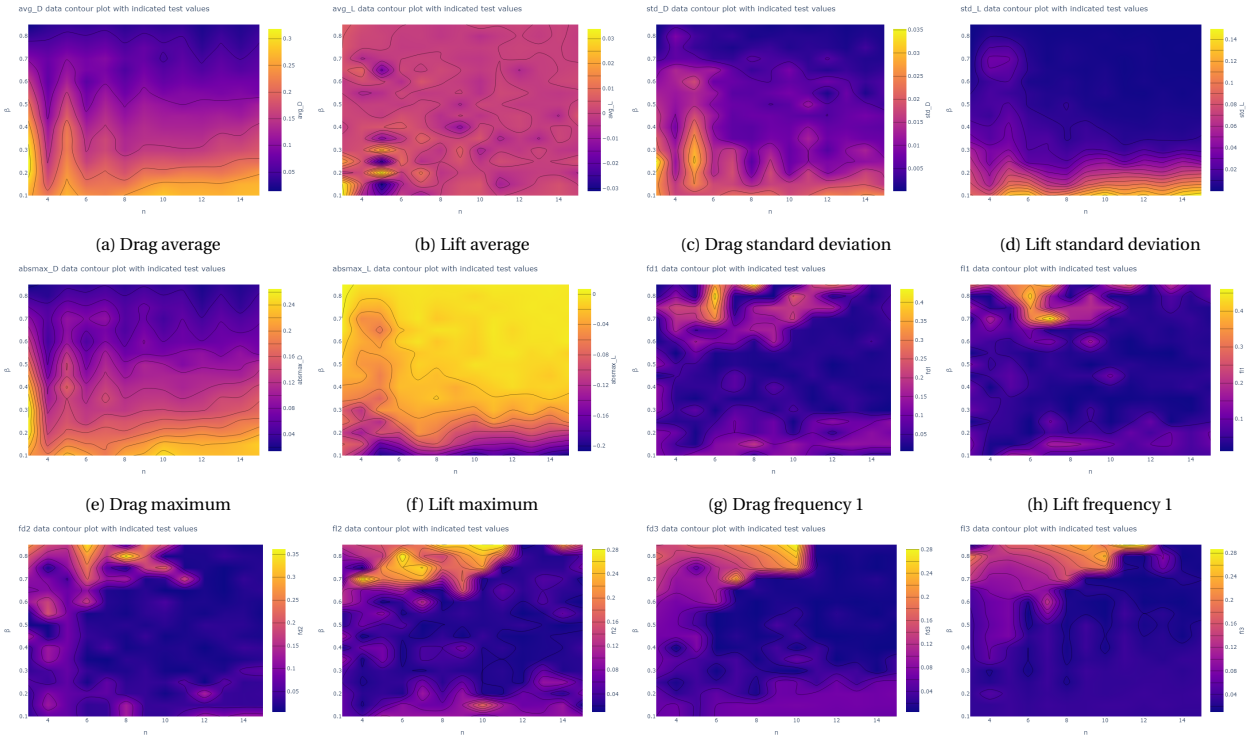
(g) 29 - 0.3 (h) 29 - 0.5 (i) 29 - 0.7

Figure C.9: Different selected fitment visualizations for the Full-NN model (OC data)

D

RD data fit visual representation

D.1. RD Engineering models



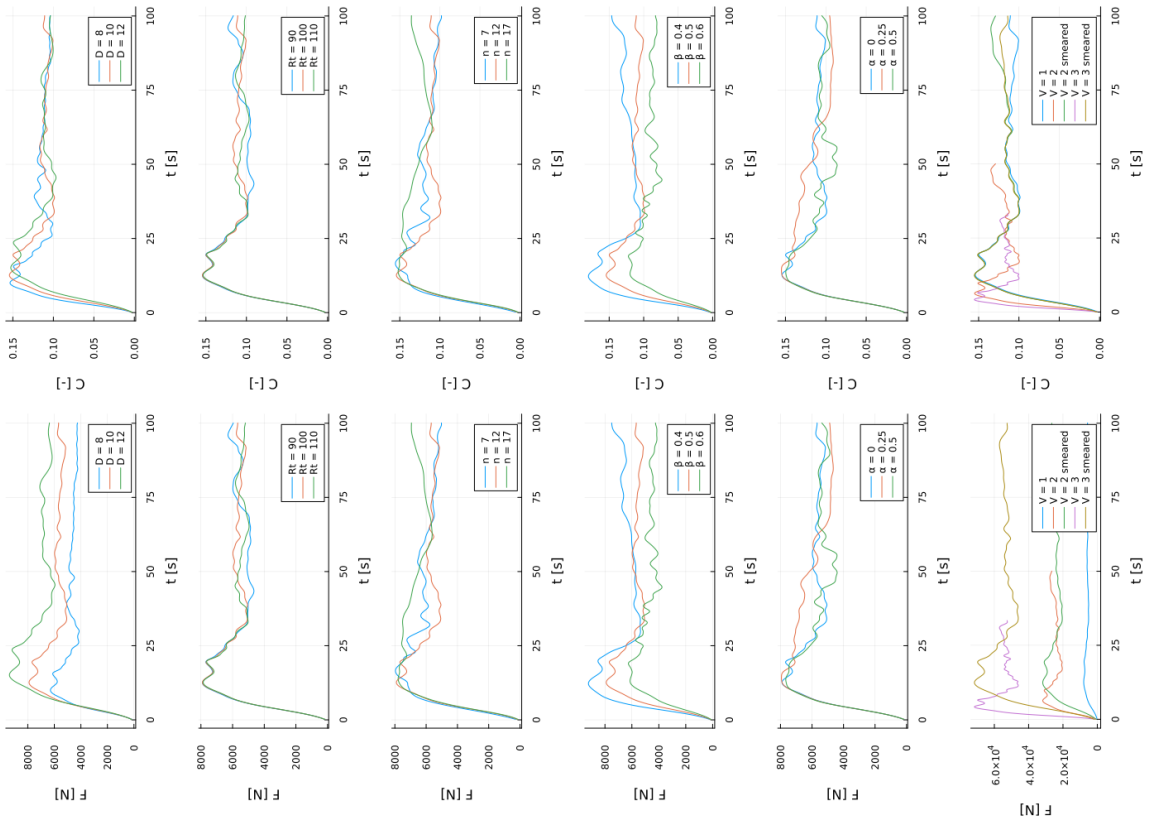
(a) Drag average (b) Lift average (c) Drag standard deviation (d) Lift standard deviation

(e) Drag maximum (f) Lift maximum (g) Drag frequency 1 (h) Lift frequency 1

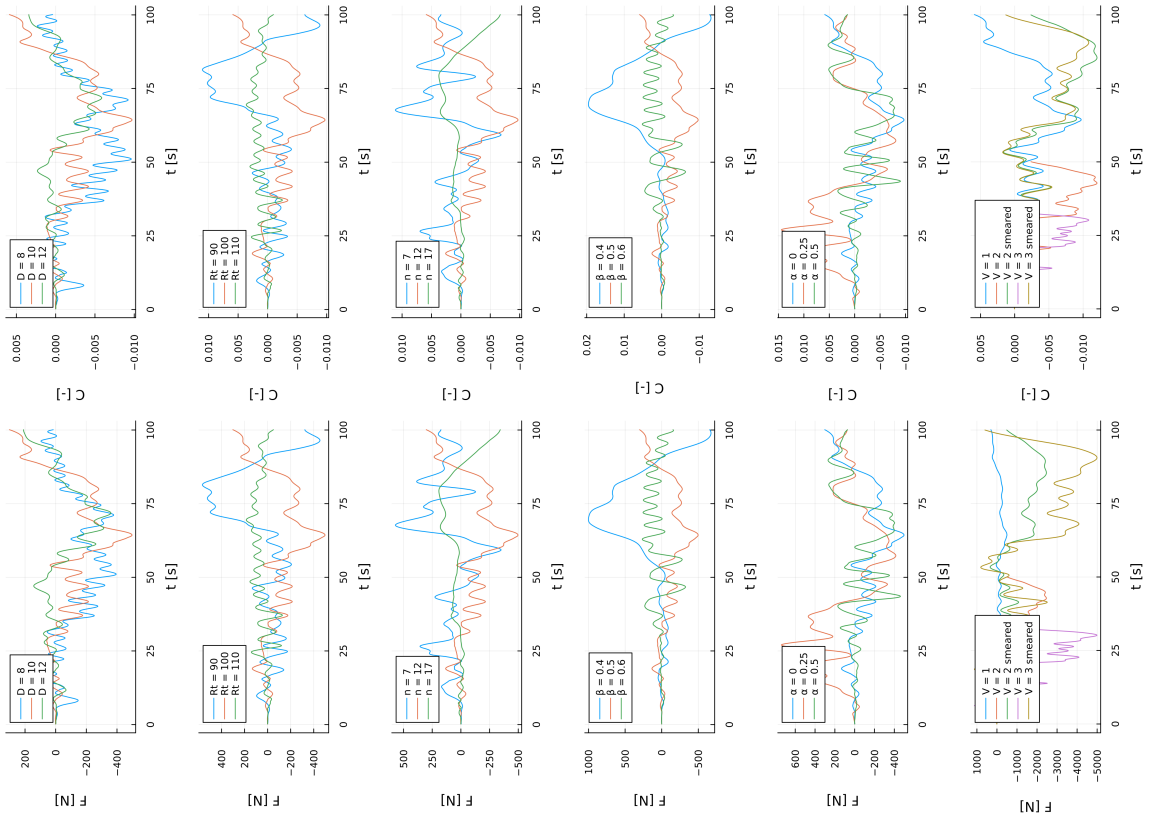
(i) Drag frequency 2 (j) Lift frequency 2 (k) Drag frequency 3 (l) Lift frequency 3

Figure D.1: Drag and lift engineering mean values: contour plot visualization

D.2. RD variable importance visualization



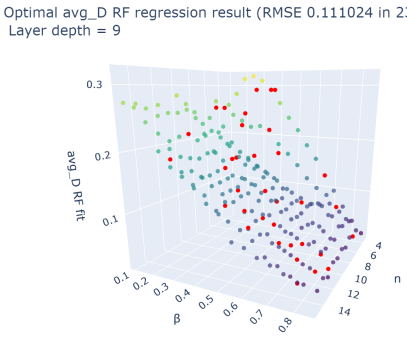
(a) Drag



(b) Lift

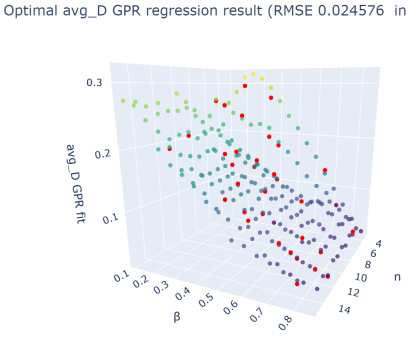
Figure D.2: Box run results to determine important design parameters

D RD 3D data regression results: import

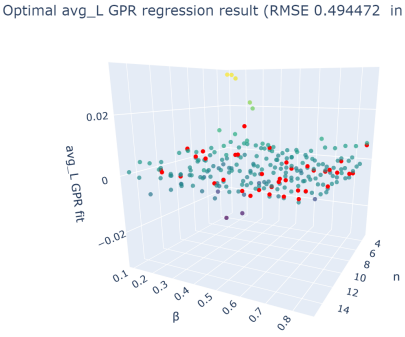


(a) RF lift

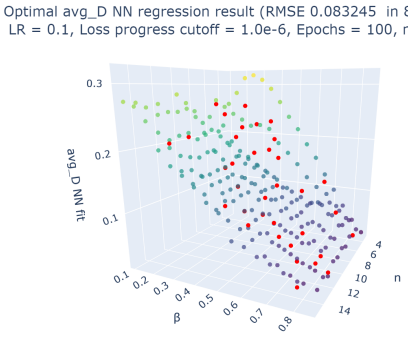
(b) RF drag



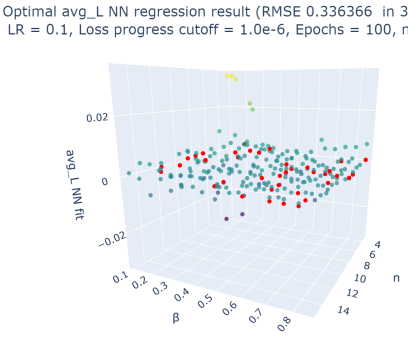
(c) GPR drag



(d) GPR drag

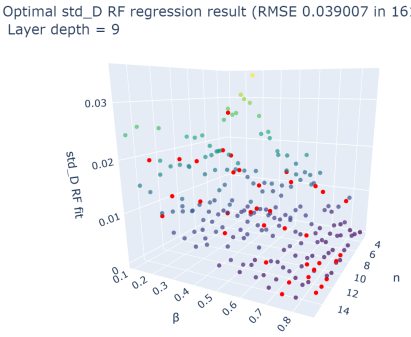


(e) NN drag

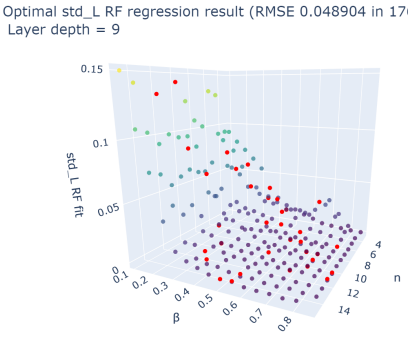


(f) NN lift

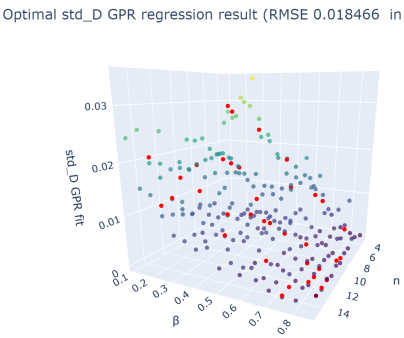
Fig. 15: Average drag and lift regression quality visualization in a 3D plot



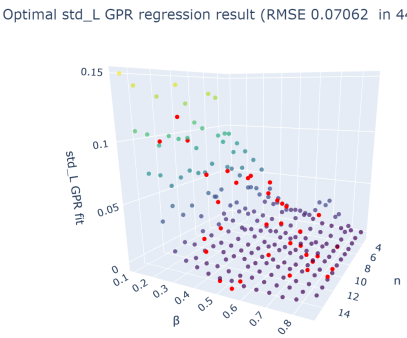
(a) RF lift



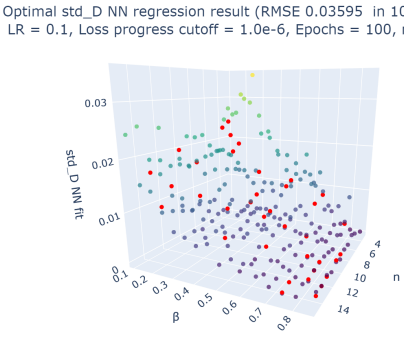
(b) RF drag



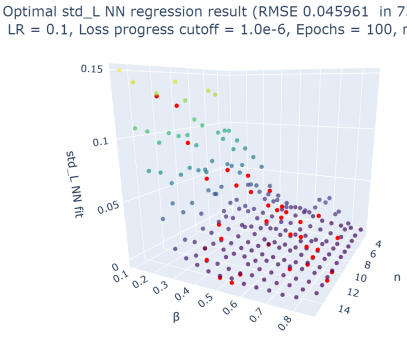
(c) GPR drag



(d) GPR drag

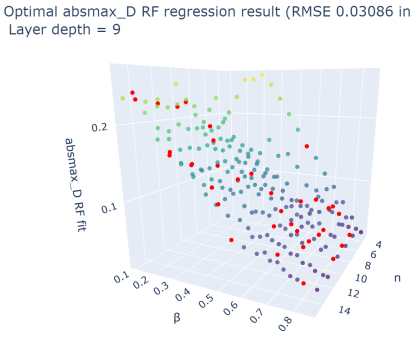


(e) NN drag

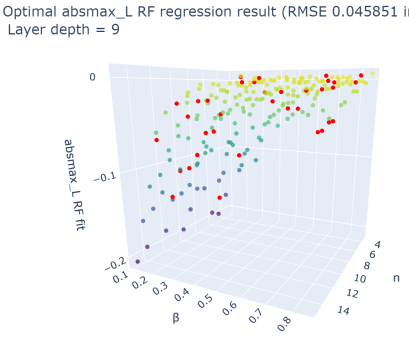


(f) NN lift

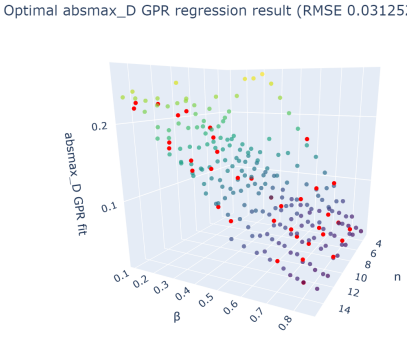
Fig. 16: Standard deviation drag and lift regression quality visualization in a 3D plot



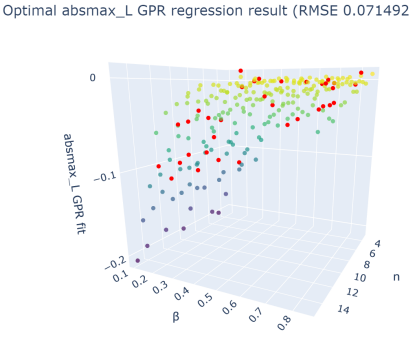
(a) RF lift



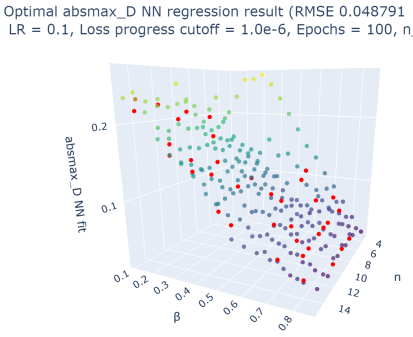
(b) RF drag



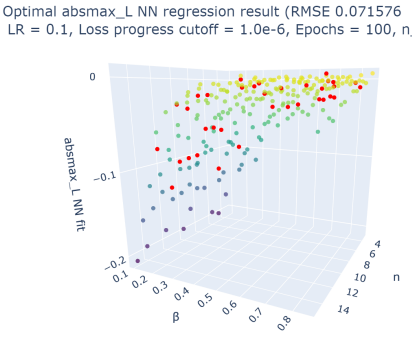
(c) GPR drag



(d) GPR drag

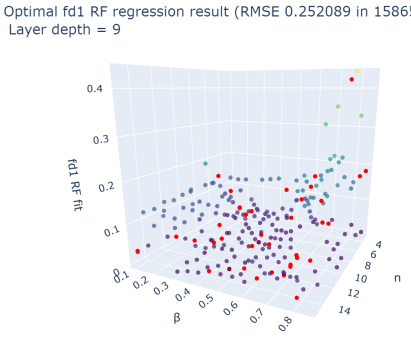


(e) NN drag

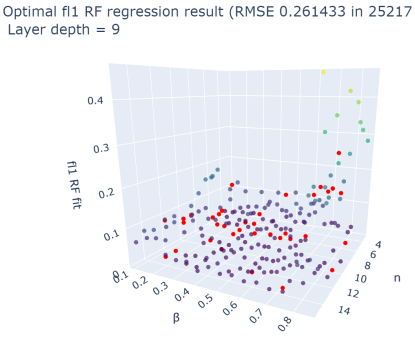


(f) NN lift

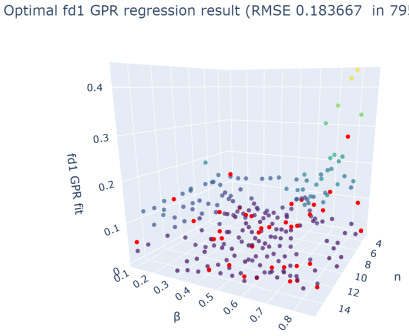
Fig. 17: Maximum absolute value (99% percentile) drag and lift regression quality visualization in a 3D plot



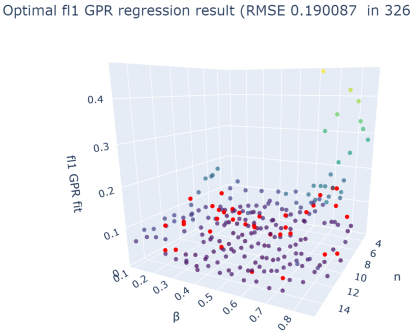
(a) RF lift



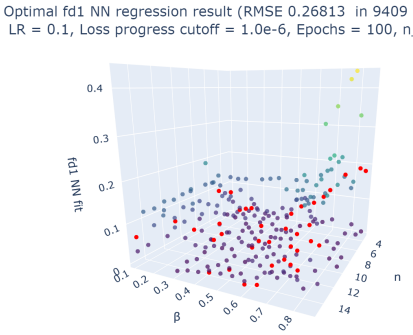
(b) RF drag



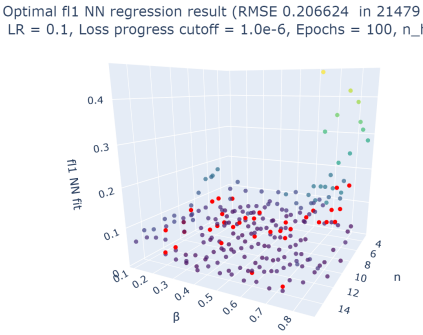
(c) GPR drag



(d) GPR drag

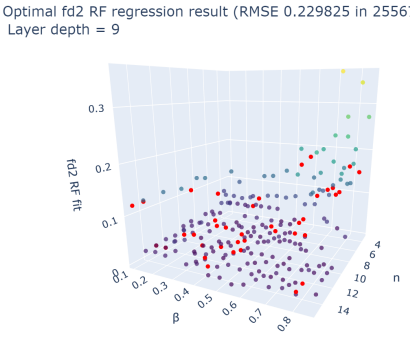


(e) NN drag

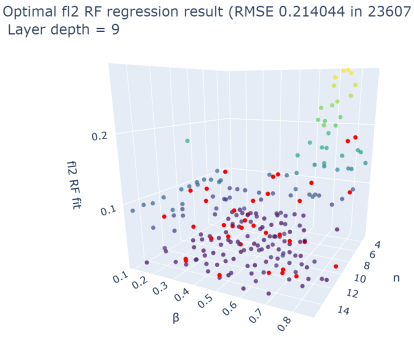


(f) NN lift

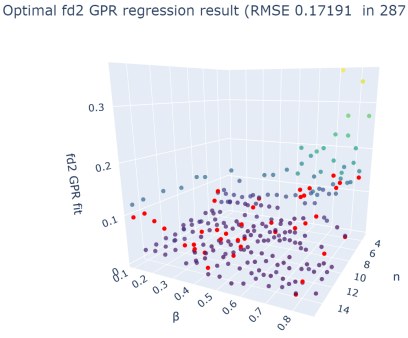
Fig. 18: First highest Fourier forcing frequency regression quality visualization in a 3D plot



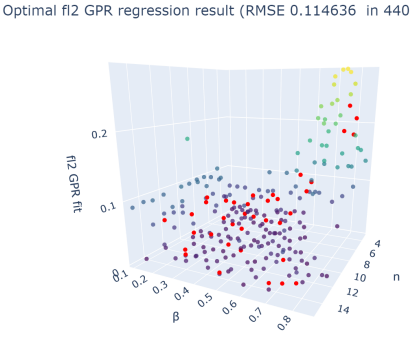
(a) RF lift



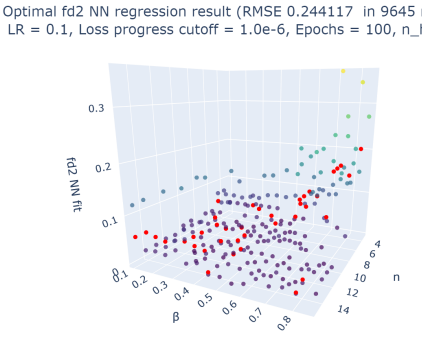
(b) RF drag



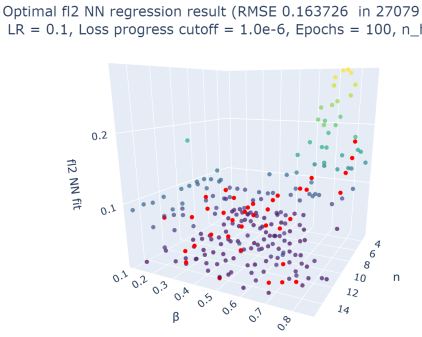
(c) GPR drag



(d) GPR drag

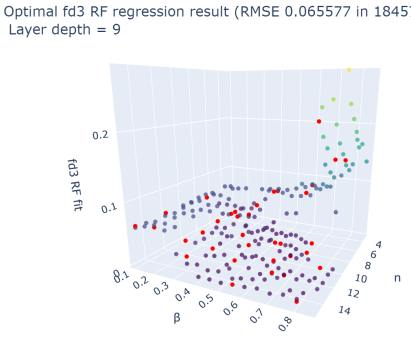


(e) NN drag

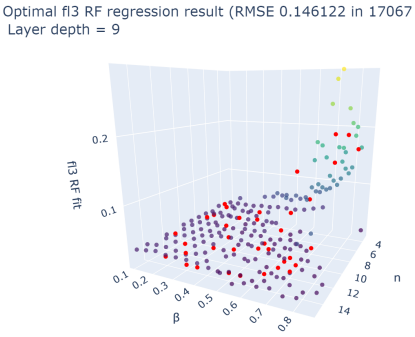


(f) NN lift

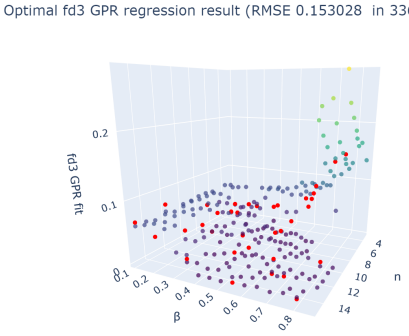
Fig. 19: Second highest Fourier forcing frequency regression quality visualization in a 3D plot



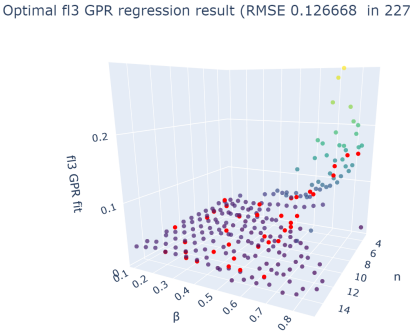
(a) RF lift



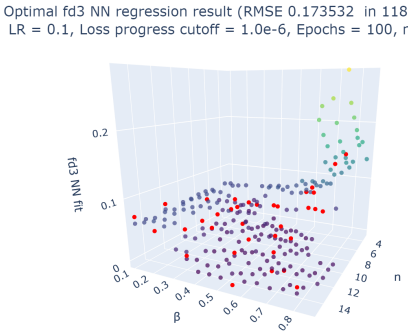
(b) RF drag



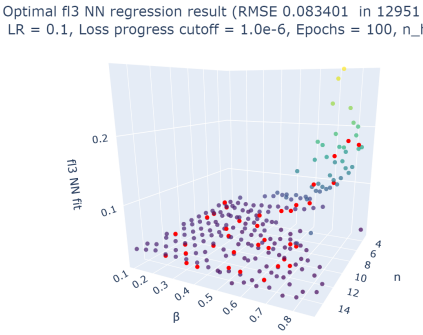
(c) GPR drag



(d) GPR drag



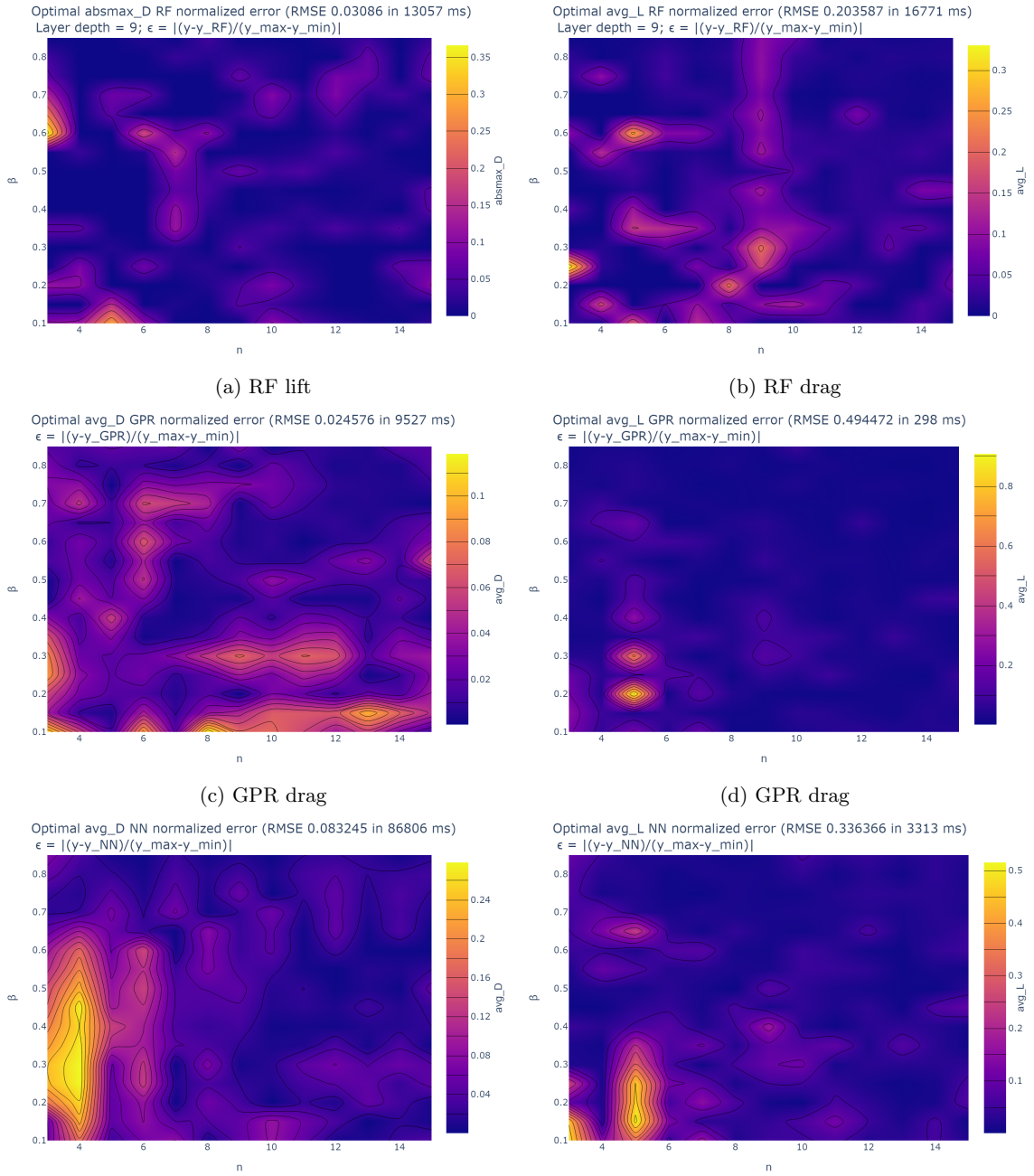
(e) NN drag



(f) NN lift

Fig. 20: Third highest Fourier forcing frequency regression quality visualization in a 3D plot

E RD Contour data regression errors: import



(a) RF lift

(b) RF drag

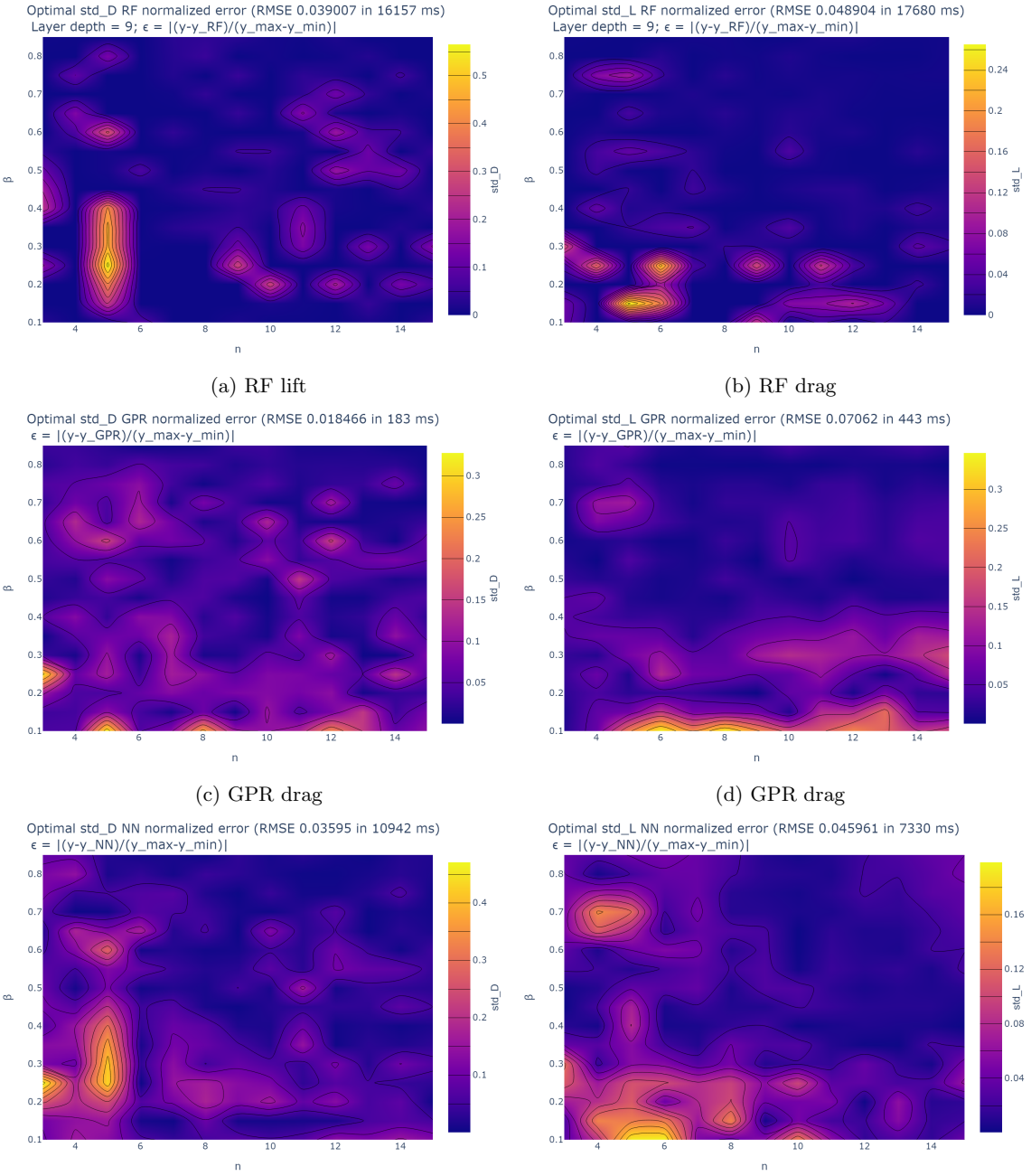
(c) GPR drag

(d) GPR drag

(e) NN drag

(f) NN lift

Fig. 21: Average drag and lift regression quality visualization in a contour plot



(a) RF lift

(b) RF drag

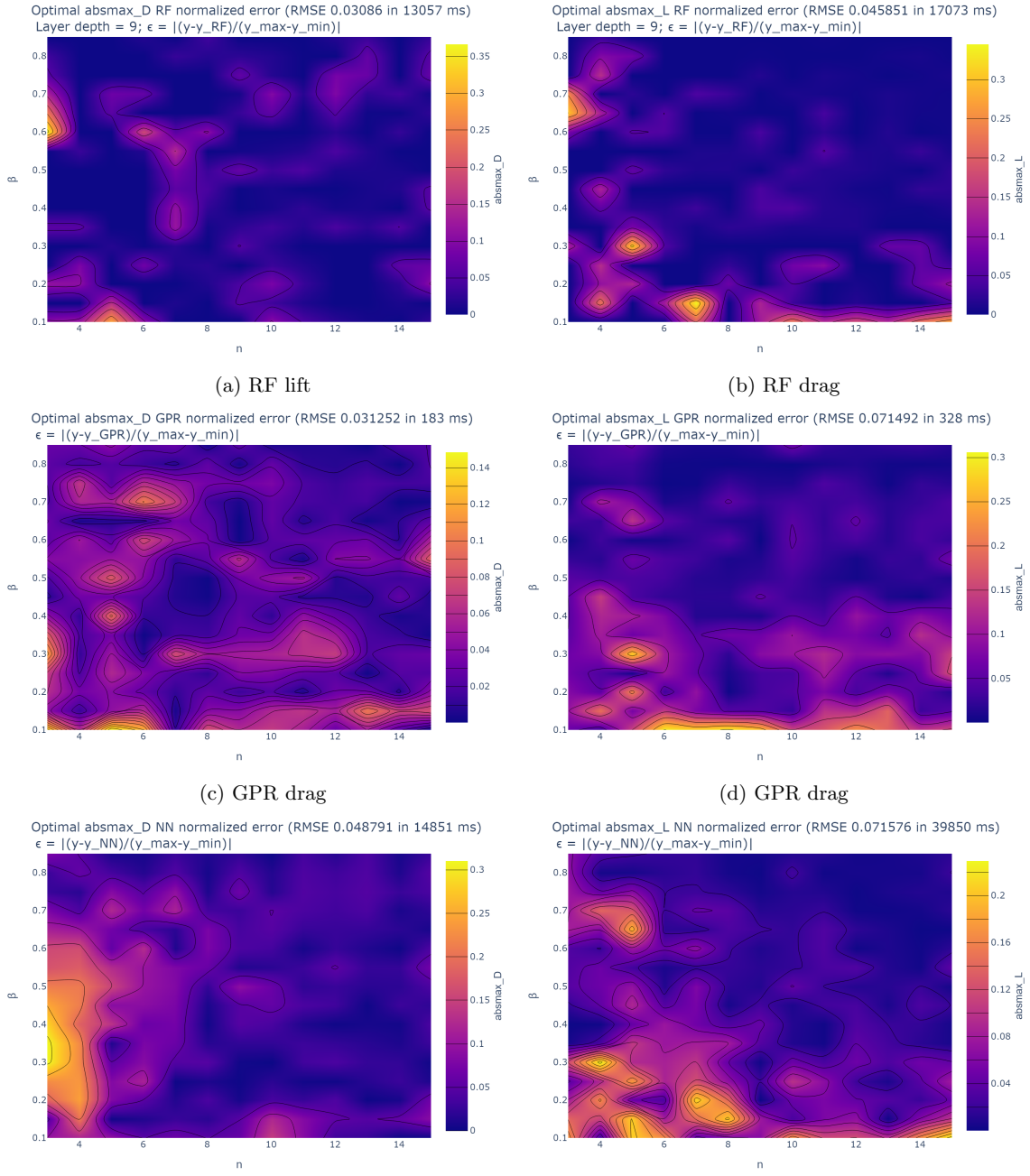
(c) GPR drag

(d) GPR drag

(e) NN drag

(f) NN lift

Fig. 22: Standard deviation drag and lift regression quality visualization in a contour plot



(a) RF lift

(b) RF drag

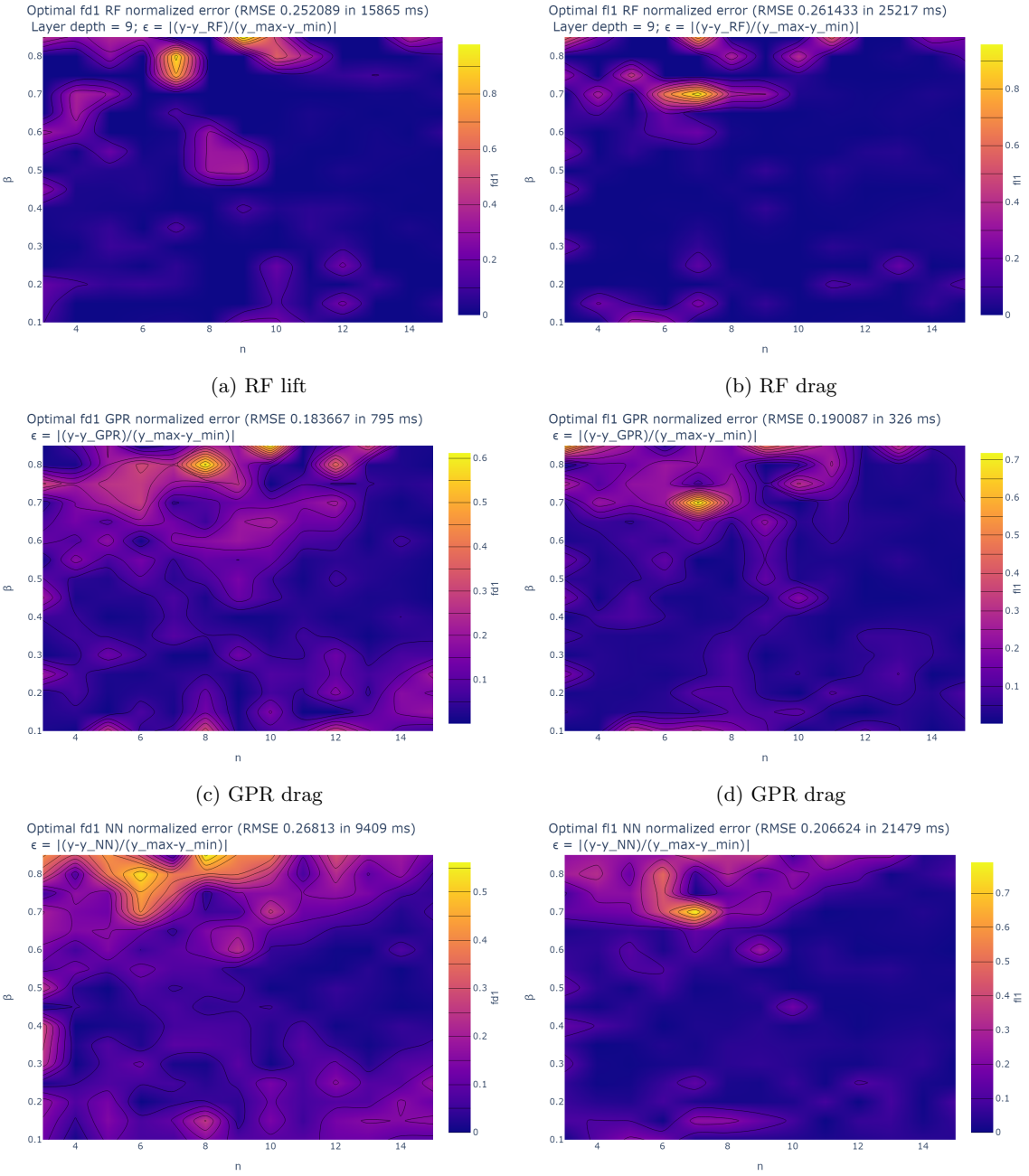
(c) GPR drag

(d) GPR drag

(e) NN drag

(f) NN lift

Fig. 23: Maximum absolute value (99% percentile) drag and lift regression quality visualization in a contour plot



(a) RF lift

(b) RF drag

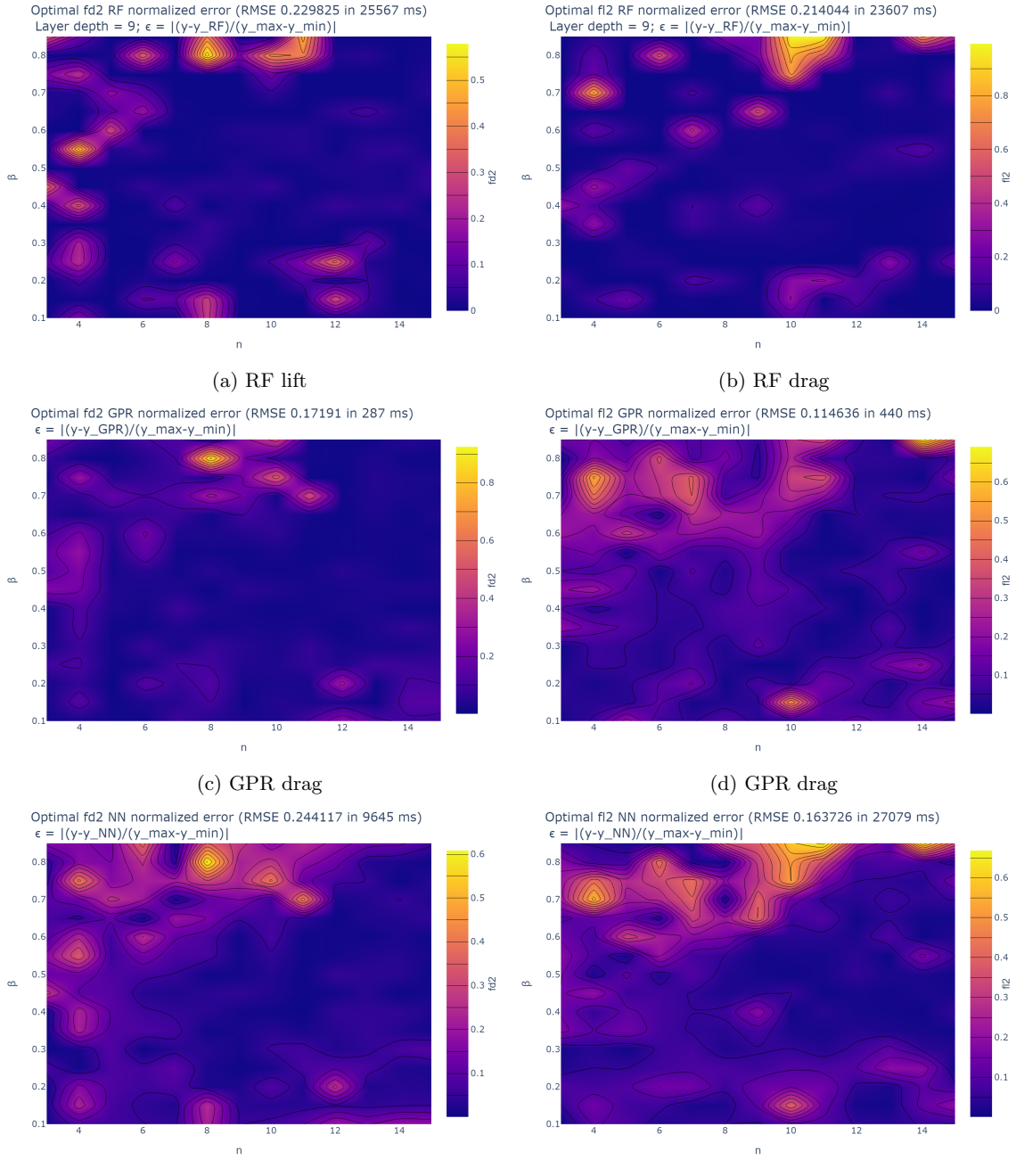
(c) GPR drag

(d) GPR drag

(e) NN drag

(f) NN lift

Fig. 24: First highest Fourier forcing frequency regression quality visualization in a contour plot



(a) RF lift

(b) RF drag

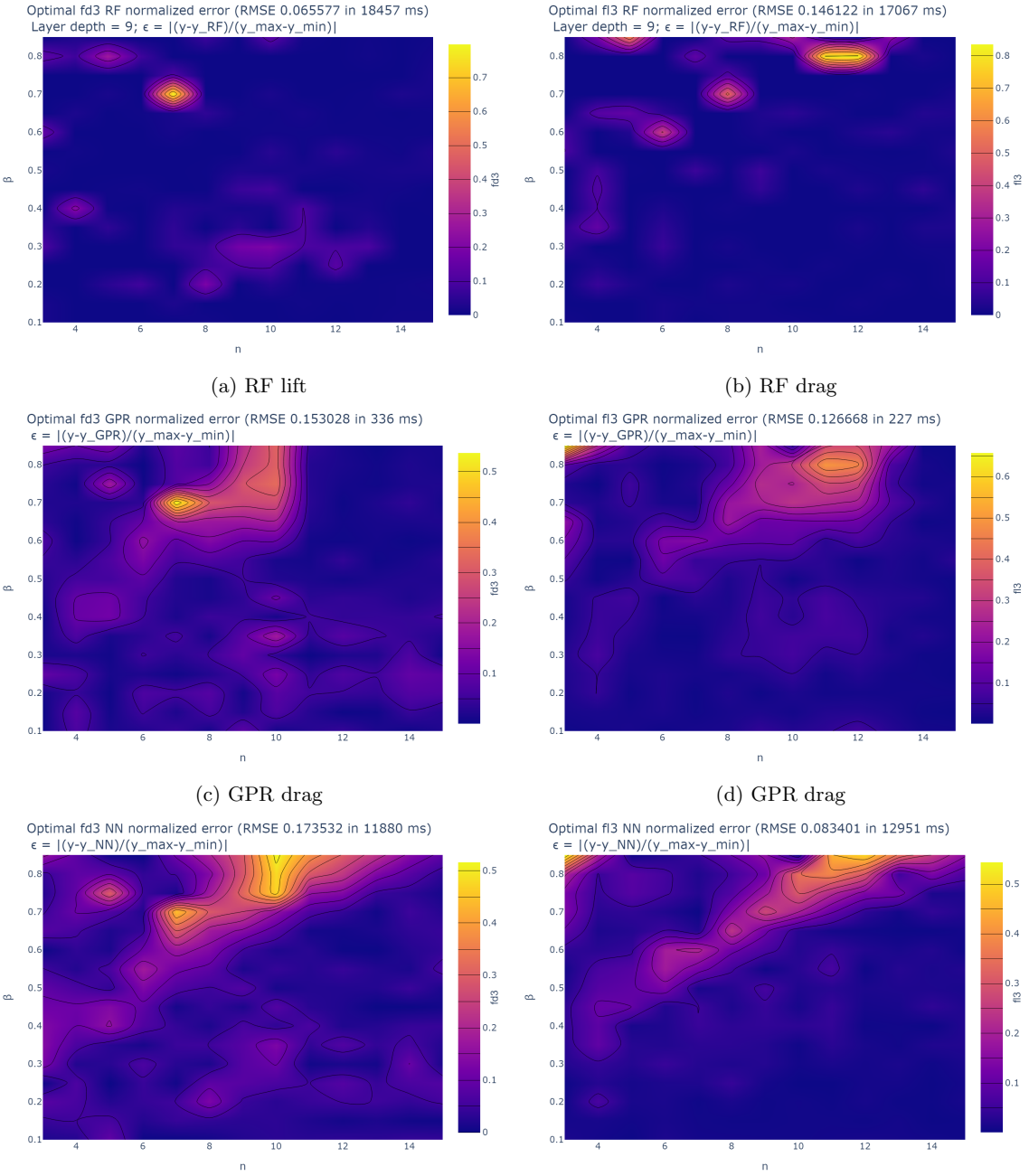
(c) GPR drag

(d) GPR drag

(e) NN drag

(f) NN lift

Fig. 25: Second highest Fourier forcing frequency regression quality visualization in a contour plot



(a) RF lift

(b) RF drag

(c) GPR drag

(d) GPR drag

(e) NN drag

(f) NN lift

Fig. 26: Third highest Fourier forcing frequency regression quality visualization in a contour plot