Studienarbeit

Investigation of Graph Neural Networks for Prediction of Aerodynamic Quantities in the Frequency Domain

Henrik Lange
Matriculation Number 5331730

Examiner:	Prof. Dr. Stefan Görtz Institute of Aircraft Design and Lightweight Structures Technische Universität Braunschweig
Supervisor:	Derrick Armando Hines Chaves
Written at:	German Aerospace Center (DLR)
Published:	February 2024

DLR e. V. Institut für Aerodynamik und Strömungstechnik
Lilienthalplatz 7, 38108 Braunschweig

Ihr Gesprächspartner	Prof. Dr. S. Görtz
Telefon	0531
Telefax 0531 295-	3357
E-Mail	stefan.goertz@dlr.de

07. July 2023

Description of Work for a "Studienarbeit"

Investigation of Graph Neural Networks for Prediction of Aerodynamic Quantities in the Frequency Domain

Student Name:	**Henrik Lange**
Enrollment number:	**5331730**
Duration:	**16.10.2023 – 31.01.2024**
First Examiner:	**Prof. Dr. Stefan Görtz**
Scientific Supervisor:	**Derrick Armando Hines Chaves**

The numerical investigation of dynamic responses to atmospheric turbulence as well as structural and flight dynamic excitations is an important task during the aircraft design and certification process. Efficient and high-fidelity methods are desirable because large parameter spaces spanned by, for example, Mach number, flight altitude, load case, and gust shape need to be covered [1,2]. Aerodynamic nonlinearities such as shocks and boundary-layer separation should be included to account for transonic flight conditions. The linear frequency-domain method computes aerodynamic responses using computational fluid dynamics by linearizing the Reynolds-averaged Navier–Stokes equations around a steady-state solution. While this enables efficient simulations for fixed operational conditions such as Mach number and angle of attack, it is still not suitable for direct incorporation in multi-query scenarios. This limitation can potentially be lifted using data-driven reduced order models. Within the past few years deep-learning based models have been investigated quite extensively for steady and unsteady aerodynamic predictions. In particular graph neural networks (GNN) which transfer information from the underlying computational grid in a graph, that is incorporated during the network training process, have yielded impressive results [3]. The objective of this Studienarbeit is to extended the existing GNN capabilities within SMARTy towards complex-valued data sets to enable handling of frequency domain analysis results. Newly derived capabilities should be investigated to identify potentials and limitations. For this an LFD dataset at different operational conditions will be set up, a model build and predictive capabilities of it will be evaluated by comparing to reference results.

Description of Work	**Duration in weeks**
Review of available reduced order models (ROM) for linear frequency domain problems with a focus on deep learning	2
Familiarize with linear frequency domain capabilities for gust response simulations	2
Familiarize with pytorch-geometric capabilities and existing GNN implementation in the DLR SMARTy toolbox	3
Extend the GNN implementation in SMARTy to handle complex-valued data	3
Investigate potentials and limitations of the implemented GNN-based method for frequency-domain data	3
Writing of Studienarbeit	2
Sum	**15**

[1] Thormann, R., and Widhalm, M., "Linear-Frequency-Domain Predictions of Dynamic-Response Data for Viscous Transonic Flows," AIAA Journal, Vol. 51, No. 11, 2013, pp. 2540–2557
[2] Bekemeyer, P., and Thormann, R. and Timme, S., "Frequency-Domain Gust Response Simulation Using Computational Fluid Dynamics," AIAA Journal, Vol. 55, No. 7, 2017, pp 2174–2185
[3] Hines, D., and Bekemeyer, P., "Graph neural networks for the prediction of aircraft surface pressure distributions," Aerospace Science and Technology, Vol. 137, 2023, 108268

Prof. Dr. Stefan Görtz

Declaration of Independence

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work. I also hereby declare that my thesis has not been prepared for another examination or assignment, either in its entirety or excerpts thereof.

Henrik Lange

Braunschweig, 2nd February 2024

Abstract

Modern aircraft development requires ever-faster evaluations of physical quantities considering large parameter spaces. Conventional computational methods are prohibitive due to the required computational effort. Data-driven surrogate models allow very fast evaluation times, by shifting computational effort to a prior training phase. Deep-learning models have proven capable of capturing high-complexity data.

Unsteady aerodynamic effects, e.g. caused by discrete gusts, must be considered during aircraft design. Conventional methods such as the linear frequency domain solver allow computing solutions in the frequency domain. This work investigates the applicability of graph neural networks as surrogate models for such problems. Graph neural networks can use the spatial information given by a computational mesh to enhance the prediction accuracy as prior work on steady aerodynamic problems has shown.

Two types of graph neural networks, namely a graph convolutional network with a *Residual Gated Graph ConvNets* convolution, and the graph network simulator model are compared using only global flow features M, α, and ω^* with models using pressure as an additional local flow feature. The complex-valued increment to the pressure coefficient at the airfoil surface is used as the target quantity. An interpolation model and a node-wise predicting fully-connected neural network are used as comparisons.

In this work, all models are compared using the NACA 64A010 airfoil as a two-dimensional test case. Therefore, a sampling strategy is described to generate a dataset. Subsequently, the models are optimized towards the available data.

Evaluating the models using four different metrics indicates that the two graph neural network approaches using local and global flow features deliver the best predictions. Especially, comparing the deep-learning results to the interpolation model shows advantages in the capability of capturing non-linear flow characteristics in the transonic flight regime. However, with the available dataset and chosen hyperparameters, none of the models delivers satisfactory predictions, especially at low reduced frequency transonic samples.

Further investigations test the models' behaviours on differently sized datasets, as well as datasets containing subsets of reduced frequencies respectively subsets of Mach numbers and angle of attack. Regarding the computational effort, all data-driven approaches show the potential to evaluate samples approximately four orders of magnitude faster than the linear frequency domain approach. Overall the graph network simulator model using local flow features shows the greatest potential. Finally, ideas regarding further improvements and continuing investigations are given.

Contents

Nomenclature **vii**

1 Introduction **1**
1.1 Related Work . . . 1
1.2 Research Objective . . . 2
1.3 Structure . . . 2

2 Problem Definition **4**
2.1 Linear Frequency Domain Method . . . 4
2.2 Resulting Task . . . 5

3 Data-Driven Methods **7**
3.1 Machine Learning . . . 7
3.2 Geometric and Graph Data . . . 7
3.3 Methods . . . 8
3.3.1 Interpolation . . . 8
3.3.2 Fully-Connected Neural Network . . . 9
3.3.3 Message Passing Neural Network . . . 10
3.3.4 Graph Convolutional Network . . . 11
3.3.5 Graph Network Simulator . . . 11
3.3.6 Model Summary . . . 13
3.3.7 Advantages and Disadvantages . . . 13
3.3.8 Software Implementation . . . 13

4 Test Case **14**
4.1 NACA 64A010 Airfoil . . . 14
4.2 Design of Experiments . . . 14
4.3 Mesh Description and Graph Creation . . . 15
4.4 Flow Simulations . . . 16
4.4.1 Steady Simulations . . . 16
4.4.2 Linear Frequency Domain Simulations . . . 16
4.4.3 Convergence . . . 16
4.5 Features . . . 17
4.6 Targets . . . 18
4.7 Resulting Dataset . . . 18
4.8 Train-Validation-Test-Split . . . 18

5 Model Selection **20**
5.1 Description of Models . . . 20
5.2 Hyperparameter Optimization . . . 21
5.2.1 Method . . . 21
5.2.2 Optimization Variables . . . 21
5.2.3 Results . . . 22

6 Results **23**
6.1 Metrics . . . 23
6.2 Optimized Models . . . 24
6.2.1 Evaluation Using the Metrics . . . 24
6.2.2 Low Mach Number Behaviour . . . 25
6.2.3 High Mach Number Behaviour . . . 27

6.3 Size of the Dataset 32
6.4 Unseen Reduced Frequencies 34
6.4.1 Creation of Reduced Dataset 34
6.4.2 Results 34
6.5 High Mach Number Models 36
6.5.1 Creation of Reduced Dataset 36
6.5.2 Results 36
6.6 Computational Effort 37
6.6.1 Training Time 37
6.6.2 Number of Parameters 38
6.6.3 Evaluation Time 38

7 Conclusions and Outlook 40
7.1 Conclusions 40
7.2 Outlook 42

Bibliography 48

List of Figures 50

List of Tables 51

A Appendix 52
A.1 Data-Driven Methods 52
A.2 Test Case 53
A.2.1 Reynolds Number 53
A.2.2 Features 53
A.3 Model Selection 54
A.3.1 Scaling 54
A.3.2 Hyperparameters 54
A.3.3 HPO Results 56
A.4 Results 59
A.4.1 Models after HPO 59
A.4.2 Complementary Figures for all Models 60
A.4.3 Size of Dataset 64
A.4.4 Unseen Frequencies 67
A.4.5 High Mach Number Models 68

Nomenclature

Latin

$\mathbf{A}$	Adjacency matrix
$\tilde{a}_{j,i}$	Distance of face normals between dual elements i and j
b	Bias
$\mathbf{b}$	Bias vector
c_1^i	Coordinate in direction 1 at the i-th node
c_l	Coefficient of lift
c_p	Coefficient of pressure
c_{ref}	Reference chord length
$\mathbf{E}$	Edge feature matrix
$e_{j,i}$	Directed edge between from node j to node i
f	Function represented by a surrogate model
$\mathcal{G}$	Graph
$\mathbf{g}$	Vector of geometric information
$\mathbf{h}^{(i)}$	i-th layer of FCNN
$\mathbf{h}_i^{(t)}$	Node feature vector at i-th node at t-th step
$\mathbf{I}$	Identity matrix
$\Im$	Imaginary part of complex number
k	Frequency of oscillation
L_g	Gust length
l	Number of layers
M	Mach number
M_t	Message passing function
$\mathbf{m}_i^t$	Aggregated message vector to node i at step t
$\mathbf{m}_{j,i}^t$	Message vector from note j to i at step t
$\mathcal{N}$	Function collection the neighbouring nodes
p	Pressure
$\mathbf{R}$	Vector of non-linear residual functions
$\Re$	Real part of complex number
Re	Reynolds number
$r_{j,i}$	Distance between nodes i and j
U_t	Update function
U_∞	Free-stream velocity
u	Velocity in x-direction
$\mathbf{u}$	Weight vector
$\tilde{u}_{SA}$	Friction velocity of turbulence model
V	Set of nodes
v_1^i	Surface normal at dual element i in direction 1
$v_{g,z}$	Vertical gust amplitude

$\mathbf{v}_g$	Gust disturbance vector
v	Velocity in y-direction
v_i	Node i
$\mathbf{W}$	Weight matrix
$\hat{\mathbf{w}}_g$	Complex-valued gust disturbance vector
w	Velocity in z-direction
$\mathbf{w}$	Vector of conservative variables
$\hat{\mathbf{w}}$	Complex-valued frequency domain response of conservative variables
$\dot{\mathbf{w}}$	Temporal change of the vector of conservative variables
$\mathbf{x}$	Input vector
$\mathbf{x}_i$	Node feature vector of node i
$\mathcal{X}$	Set of input vectors
x_0	Initial distance between gust and airfoil
$\mathcal{Y}$	Set of output values
y	Output value

Greek

α	Angle of attack
β	Model parameters
δ	Gaussian noise
ϵ	Gaussian process
$\boldsymbol{\kappa}$	Vector of global flow features
μ	Mean value
ν_t	Turbulent viscosity
ϕ	Basis function
φ	Arbitray function
$\boldsymbol{\varphi}$	Vector of phase shifts
ρ	Density
σ	Sigmoid logistic function
σ^2	Variance
$\Theta_{j,i}$	Angle between node i and j
$\tilde{\Theta}_{j,i}$	Angle between face of dual elements i and j
ω^*	Reduced frequency
$\Delta\mathbf{c}_p$	Vector of complex valued increments to the pressure coefficient

Indices

0	Value at equilibrium
all	Entire domain is regarded
D	Decoder
E	Encoder
max	Maximum
pred	Predicted value

surface	Only airfoil surface is regarded
true	True value

Abbreviations

AI	Artificial Intelligence
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
DL	Deep Learning
DLM	Doublet Lattice Method
DLR	Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center)
DoE	Design of Experience
FCNN	Fully-Connected Neural Network
GCN	Graph Convolutional Network
GMRes	Generalized Minimal Residual
GNN	Graph Neural Network
GNS	Graph Network Simulator
HPO	Hyperparameter Optimization
LFD	Linear Frequency Domain
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multi Layer Perceptron
MPNN	Message Passing Neural Network
MSE	Mean Squared Error
NACA	National Advisory Committee for Aeronautics
POD	Proper Orthonogal Decomposition
RBF	Radial Basis Function
RANS	Reynolds-Averaged Navier-Stokes
ROM	Reduced Order Model
ReLU	Rectified Linear Unit
SMARTy	Surrogate Modeling for Aero Data Toolbox Python Package
TPS	Thin Plate Spline
URANS	Unsteady Reynolds-Averaged Navier-Stokes

Chapter 1

Introduction

Future aircraft design by means of multidisciplinary design optimization requires computing results in various disciplines across large parameter spaces and multiple times to result in disruptive layouts. To be able to compute large parameter spaces numerous times, fast computation models are needed when computational resources are limited. Unsteady aerodynamic effects such as gust responses must be regarded according to the certification specification [1].

Linear potential methods, such as the doublet lattice method (DLM) [2], can be computed fast. However, they are not able to capture non-linearities that are present in the transonic regime. Computational fluid dynamics (CFD) is the current industry standard for computing aerodynamic simulations. This method delivers accurate results for unsteady aerodynamic effects, but these computations are very time-consuming [3]. The linear frequency domain (LFD) approach investigated in [4] and [5] assumes small perturbations and has proven to reduce computational effort by one to two orders of magnitude when compared to computations using the unsteady Reynolds-averaged Navier-Stokes (URANS) equations. However, the use of this method in an optimization cycle, while examining a large number of points in the flight envelope is still prohibitive regarding the computational cost. Consequently, fast and efficient surrogate models are needed.

Recent advancements in computing power make the processing of large amounts of data feasible. As fluid dynamics is typically a field of research that produces large amounts of data, the usage of data-driven methods running on high-performance computers is suitable [6]. Data-driven methods allow to shift the computational effort from the point of evaluation to a preliminary training phase. Thus, being suited for multi-query scenarios. In the field of deep-learning numerous new concepts have been developed recently. One of which are graph neural networks (GNNs) that use data in the form of graphs and can therefore make use of spatial information in the data structure [7, 8].

The text above gives a motivation for this paper. Subsequently, related work is reviewed in section 1.1 before a research objective is defined in section 1.2. Finally, the structure of this work is explained in section 1.3.

1.1 Related Work

Various approaches to building data-driven models for unsteady aerodynamic problems can be found in literature. Proper orthogonal decomposition (POD) allows to reduce the dimensionality of a physical flow by finding the principal components. Along with an interpolation approach, this allows the prediction of aerodynamic quantities. In [9] it is shown how POD can be used to update an aerodynamic model which predicts general aerodynamic forces. Another study applies POD to build a reduced order model (ROM) of a time-linearized solver leading to highly accurate predictions of aerodynamic and aeroelastic behaviour in transonic flows [10]. More recent approaches combine POD with sophisticated interpolation methods such as the Grassmann manifold interpolation in [11], which as of now is only applicable in the region close to a sample point.

A more complex method to model unsteady aerodynamic characteristics is the dynamic mode decomposition which is used in [12] to compute frequency domain generalized aerodynamic forces by obtaining the most dominant fluid dynamic modes caused by small amplitude excitations.

Deep Learning (DL) methods such as neural networks have been applied to unsteady aerodynamics in multiple papers. The time-dependent prediction of unsteady boundary layers with recursive neural networks is shown in [13]. Multi-layer functionals are proven to be a suitable representation of an unsteady aerodynamic response in [14]. In [15] recurrent neural networks are used for non-linear aeroelastic reduced order modelling. Multi-layer perceptrons are used in [16] in combination with Neuro-Fuzzy models to predict general aerodynamic forces in the time and frequency domain with high accuracy and a reduction in computational effort. Overall, neural network approaches show great capabilities to cover aerodynamic non-linearities.

A recent advancement in the field of deep learning are graph neural networks, which is the parent name of different concepts that are based on message-passing neural networks (MPNNs) that use graph-structured data to send messages to neighbouring nodes. Those have been proven superior on test cases in the field of chemistry [17]. A specific form of MPNNs are graph convolutional networks (GCNs) that extend convolutional networks to graphs [18]. As a subform of GCNs residual gated graph convolutional networks use a specific convolutional architecture that has proven to deliver accurate results on different benchmark tests [19]. The graph network simulator model architecture was first introduced in the field of subsurface flows and combines the concepts of fully-connected neural networks with graph convolutional networks in an encoder-processor-decoder structure to do time step predictions [20]. The graph network simulator has also successfully been proven to be able to predict steady-state flows including non-linear local effects around an airfoil [21].

1.2 Research Objective

The main objective of this work is to investigate the applicability of GNNs for the prediction of aerodynamic quantities in the frequency domain. More specifically, in this work, GNNs are applied as surrogates of the LFD solver which calculates frequency domain responses due to gust excitations. By comparing different GNN models to other data-driven approaches an evaluation regarding accuracy and computational effort can be done. Furthermore, to fulfill the objective the different approaches are investigated regarding specific strengths and weaknesses.

1.3 Structure

After this chapter gave a motivation and surveyed existing research studies, in chapter 2 the theoretical background of the physical problem and the resulting task is explained. Next, in chapter 3 different data-driven methods including two types of graph neural networks are explained. The different methods are evaluated using the test case that is explained in chapter 4. Based on the methods explained in chapter 3 specific models are described and their optimization is represented in chapter 5. The models are then analyzed in different investigations in chapter 6. Finally, conclusions are drawn and an outlook are given in chapter 7.

Chapter 2

Problem Definition

The linear frequency domain (LFD) method can be used to obtain a complex-valued solution to an excitation. In this work the goal is to build data-driven models that act as surrogates of the LFD solver. In section 2.1 the LFD solver is explained. Subsequently, section 2.2 explains the resulting task of this paper.

2.1 Linear Frequency Domain Method

The LFD has the ability to calculate the dynamic response behaviour from a harmonic disturbance, for example, the dynamic lift behaviour of an airfoil undergoing a harmonic pitch oscillation. This work only takes gust excitations into account. The certification specification [1] defines a discrete 1-cos gust shape, it can be seen in figure 2.1. The parameters in the figure are the free-stream velocity U_∞, the vertical gust amplitude $v_{g,z}$, the gust length L_g, and the initial distance between gust and airfoil x_0. Fourier-transforming a gust excitation into frequency domain allows a time-independent solution using a linear solver.

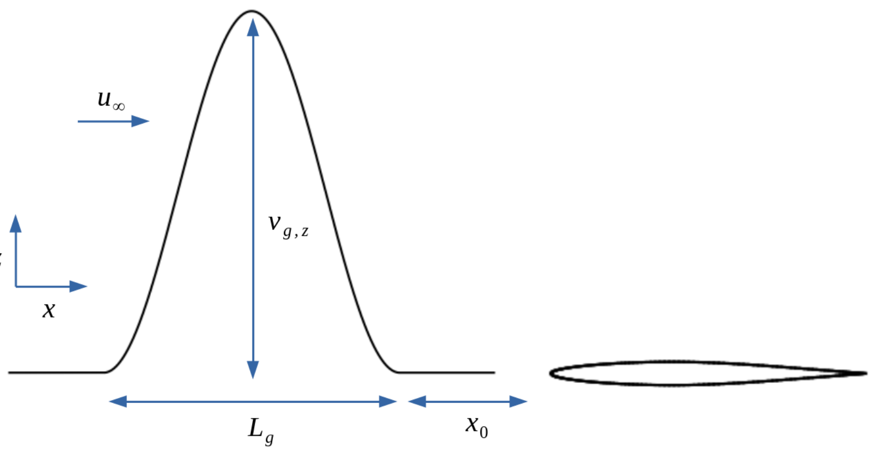


Figure 2.1: Sketch of 1-cos gust with airfoil and parameters [5]

Considering the semi-discrete form of the URANS equations the response to a gust excitation can be defined by

$$\dot{\mathbf{w}} = \mathbf{R}\left(\mathbf{w}, \mathbf{v}_g\right) \tag{2.1}$$

with the vector of conservative variables $\mathbf{w}$, the temporal change of the vector of conservative variables $\dot{\mathbf{w}}$, the gust disturbance vector $\mathbf{v}_g$, and the vector of non-linear residual functions $\mathbf{R}$.

A first-order Taylor expansion around an equilibrium point can be done by assuming small perturbations, and by defining the differences $\Delta\mathbf{w} = \mathbf{w} - \mathbf{w}_0$ and $\Delta\mathbf{v}_g = \mathbf{v}_g - \mathbf{v}_{g0}$ where

$\mathbf{w}_0$ and $\mathbf{v}_{g0}$ are the conservative values and the gust disturbance at the equilibrium:

$$\frac{\mathrm{d}\Delta\mathbf{w}}{\mathrm{d}t} = \mathbf{R}\left(\mathbf{w}_0, \mathbf{v}_{g,0}\right) + \frac{\partial\mathbf{R}}{\partial\mathbf{w}}\Delta\mathbf{w} + \frac{\partial\mathbf{R}}{\partial\mathbf{v}_g}\Delta\mathbf{v}_g \tag{2.2}$$

with $(\partial\mathbf{R}/\partial\mathbf{w})$ the Jacobian matrix with respect to the conservative variables and $(\partial\mathbf{R}/\partial\mathbf{v}_g)$ the change with respect to gust excitation. The term $\mathbf{R}\left(\mathbf{w}_0, \mathbf{v}_{g0}\right)$ is the non-linear steady flow residual, which is assumes $\mathbf{R}\left(\mathbf{w}_0, \mathbf{v}_{g0}\right) = 0$ for a fully converged steady-state flow simulation.

Assuming $\Delta\mathbf{w}$ and $\Delta\mathbf{v}_g$ to be harmonically changing in time allows to transfer equation 2.2 into the frequency domain:

$$\left(\frac{\partial\mathbf{R}}{\partial\mathbf{w}} - i\omega^*\mathbf{I}\right)\hat{\mathbf{w}} = -\frac{\partial\mathbf{R}}{\partial\mathbf{v}_g}\hat{\mathbf{v}}_\mathbf{g} \tag{2.3}$$

where $\hat{\mathbf{v}}_\mathbf{g}$ is the description of the gust disturbance. By solving the linear system of equations, the complex-valued solution vector $\hat{\mathbf{w}}$ is obtained. $\hat{\mathbf{w}}$ are complex-valued Fourier coefficients that explain the frequency domain response of the conservative values. ω^* is the reduced frequency normalized using the reference chord length c_{ref} and the freestream velocity U_∞:

$$\omega^* = \frac{\omega c_{ref}}{U_\infty} \text{ with} \tag{2.4}$$

$$\omega = 2\pi k \tag{2.5}$$

where k is the frequency of the oscillation.

The $(\partial\mathbf{R}/\partial\mathbf{w})$ and the grid-node velocity Jacobian $(\partial\mathbf{R}/\partial\mathbf{v}_g)$ matrices are derived through an analytical linearization or differentiation of the discrete URANS equations. The gust shape vector $\hat{\mathbf{v}}_g$ needs to be defined in order to solve equation 2.3 for $\hat{\mathbf{w}}$ with known $(\partial\mathbf{R}/\partial\mathbf{w})$, $(\partial\mathbf{R}/\partial\mathbf{v}_g)$ and ω^*. An analytical description is given by

$$\hat{\mathbf{v}}_g\left(\mathbf{x}, \omega^*\right) = v_{gz}e^{i\boldsymbol{\varphi}(\mathbf{x},\omega^*)} \tag{2.6}$$

with the vector of phase shifts $\boldsymbol{\varphi}$ that according to linear potential theory [2] is given by

$$\boldsymbol{\varphi}(\mathbf{x}, \omega^*) = (\mathbf{x} - x_0)\frac{\omega^*}{c_\text{ref}} \tag{2.7}$$

where $\mathbf{x}$ the spatial location vector is introduced as a new variable.

The equations above describe a way to solve for $\hat{\mathbf{w}}$ given a steady flow solution, and a reduced frequency. $\hat{\mathbf{w}}$ can be evaluated for multiple reduced frequencies to obtain a solution in the time domain [4, 5]. The LFD solver is implemented in the DLR-TAU code which is used in this work [22].

2.2 Resulting Task

According to [4] the LFD solver described in the section above allows for a reduction in CPU time of one to two orders of magnitude for unsteady aerodynamic predictions when compared to solving the URANS equations. However, using a data-driven surrogate with further reduced computational effort can be useful for multi-query scenarios.

A task must be defined regarding which part of the physical equations is to be replaced by a surrogate model. Considering the linear system in equation 2.3 gives multiple possibilities. The computational most effortful task is the creation of the Jacobian matrix ($\partial\mathbf{R}/\partial\mathbf{w}$). Consequently, an option could be to replace this computation with a surrogate model. However, in this paper a holistic approach is chosen: the task is to build a surrogate model that is able to predict the the complex-valued increment of the pressure coefficient $\mathbf{\Delta c_p}$ at the airfoil's surface given a steady flow solution $\mathbf{w}$ around an airfoil, and a reduced frequency ω^*. The usage of further input variables such as the Mach number or the angle of attack is allowed. $\mathbf{\Delta c_p}$ is chosen as target, since this distributed quantity over the surface delivers important information regarding gust-loads. In addition, it can be computed from the complex-valued Fourier coefficient $\hat{\mathbf{w}}$. Also, the value can be obtained as a direct output of the LFD solver in the DLR-TAU code [22]. Figure 2.2 visualizes the task.

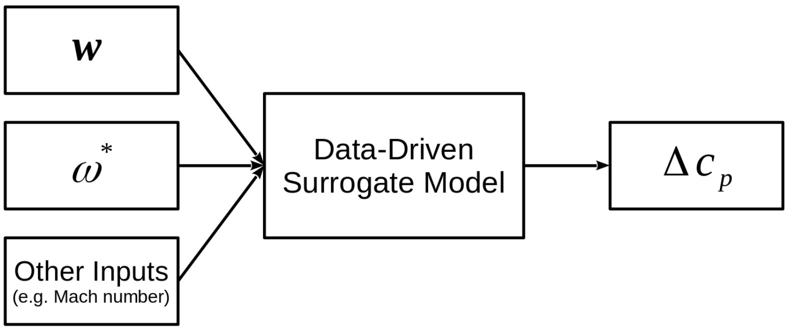


Figure 2.2: Schematic visualization of the task for a surrogate model

To allow a consistent description of the methods in the next chapter, the following formalization for the function f represented by the surrogate model is given by:

$$f\colon (\mathbf{w}(\boldsymbol{\kappa}), \boldsymbol{\kappa}, \omega^*, \mathbf{g}) \mapsto \mathbf{\Delta c_p} \tag{2.8}$$

where $\mathbf{w}$ is the vector of conservative variables of the flow solution at the airfoil surface, which are also called *local* flow features (e.g. pressure). $\boldsymbol{\kappa}$ is the vector of independent flow parameters, in the following they are also called *global* flow features (e.g. Mach number). ω^* is the reduced frequency. $\mathbf{g}$ is a vector containing geometric information. $\mathbf{\Delta c_p}$ is the vector of complex-valued pressure increments at the airfoil surface.

Chapter 3

Data-Driven Methods

The previous chapter stated the task for the surrogate methods. In this chapter, a description is provided for the six data-driven methods explored in this study with a focus on the graph neural network approaches. Physics-based modelling describes a system with equations that explain the underlying physical phenomena. In contrast, data-driven methods make use of existing data that describes a physical phenomenon. Data-driven methods aim to find structures in the data to generalize from to make predictions.

3.1 Machine Learning

Machine learning (ML) is an artificial intelligence (AI) approach. Whereas rule-based systems generate an output based on a hand-designed program that evaluates an input, classical machine learning uses hand-designed features that generate an output. Statistical methods such as regression models are a form of classical ML. Deep learning (DL) belongs to the field of ML. DL is more powerful by using several mapping layers. Thus, abstract representations of the input are generated with to goal of generating mappings of these more abstract features that are the output. The most commonly known form of DL is the multi-layer perceptron (MLP) that is explained in subsection 3.3.2, while more recent DL approaches based on graph neural networks are described in subsections 3.3.4 and 3.3.5 [23, 24].

For this study regression models are of interest. Depending on the models' capability to handle complex values, the regression models in this work can be seen as a universal approximator of the form $f\colon \mathbb{R}^n \to \mathbb{R}^m$ or $f\colon \mathbb{R}^n \to \mathbb{C}^m$ (with f from eq. 2.8, n inputs and m outputs) [23, 25].

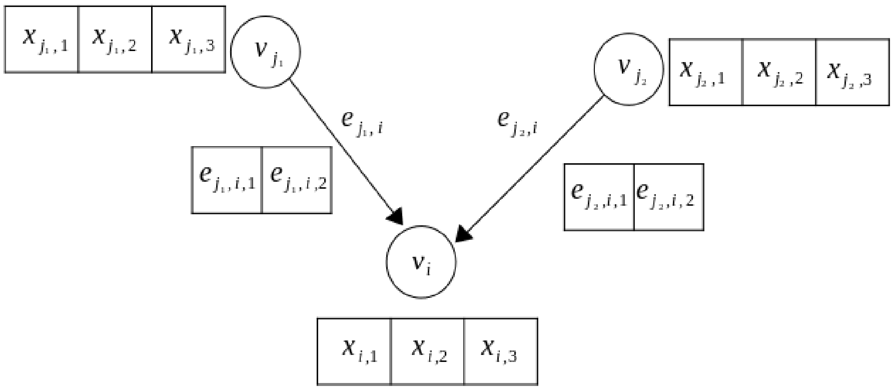
3.2 Geometric and Graph Data

Most of the methods explained in section 3.3 use geometrical information. The graph neural network approaches use the data stored in an attributed graph that is created based on the CFD mesh. Hence, this section explains the available geometric information and how it can be transformed into an attributed graph.

Attributed Graphs
V is a set of n nodes v_i. Two nodes v_i and v_j can be connected by a directed edge $e_{j,i}$. Through binary entries in the adjacency matrix $\mathbf{A} \in \mathbb{N}^{n,n}$ m edges are defined. This defines a graph $\mathcal{G} = (V, \mathbf{A})$.

By adding d_1 features in a node feature vector $\mathbf{x}_i \in \mathbb{R}^{d_1}$ to each node, and by adding d_2 features in an edge feature vector $\mathbf{e}_{j,i} \in \mathbb{R}^{d_2}$ to each edge an attributed graph is constructed. With all node feature vectors and all edge feature vectors collected in matrices

$\mathbf{X} \in \mathbb{R}^{n,d_1}$ and $\mathbf{E} \in \mathbb{R}^{n,d_2}$ an attributed graph can be written as: $\mathcal{G} = (V, \mathbf{A}, \mathbf{X}, \mathbf{E})$ [26]. Figure 3.1 gives an exemplary visualization.



$x_{j_1,1}$ $x_{j_1,2}$ $x_{j_1,3}$
v_{j_1}
v_{j_2}
$x_{j_2,1}$ $x_{j_2,2}$ $x_{j_2,3}$
$e_{j_1,i}$
$e_{j_2,i}$
$e_{j_1,i,1}$ $e_{j_1,i,2}$
$e_{j_2,i,1}$ $e_{j_2,i,2}$
v_i
$x_{i,1}$ $x_{i,2}$ $x_{i,3}$

Figure 3.1: Exemplary attributed graph with node v_i that is connected to two neighbours. Each node has three features, each edge has two features.

Graph Creation on Basis of a CFD Mesh

A CFD mesh consists of finite volumes. For the application of the different modelling approaches the spatial information needs to be in graph representation. The transformation is done using the function `Graph` from the DLR SMARTy-Toolbox [27].

The connectivity of the mesh can be translated to a graph $\mathcal{G} = (V, \mathbf{A})$ with n nodes. More geometrical information is added as node and edge features resulting in an attributed graph. The coordinates of node i are (c_1^i, c_2^i) and the surface normals at dual element i are (v_1^i, v_2^i). They are added as node feature vectors $\mathbf{x}_i$ to the node feature matrix $\mathbf{X} \in \mathbb{R}^{n,4}$. The displacements or distance vectors between node j and i in polar coordinates $(r_{j,i}, \theta_{j,i})$ and the face normals between dual elements j and i in polar coordinates $(\tilde{a}_{j,i}, \widetilde{\theta}_{j,i})$ are added as edge feature vectors $\mathbf{e}_{i,j}$ to the edge feature matrix $\mathbf{E} \in \mathbb{R}^{m,4}$. Note that subsequently the graph can be extended by adding $\mathbf{w}(\boldsymbol{\kappa})$, $\boldsymbol{\kappa}$ and ω^* as node features depending on the methodology as explained later in subsection 3.3.4 and 3.3.5.

3.3 Methods

Multiple DL approaches are compared against each other as well as against an interpolation model as a baseline comparison.

3.3.1 Interpolation

For the purpose of baseline comparison, one of the interpolation capabilities implemented in the SMARTy-Toolbox is used. Given global flow features $\boldsymbol{\kappa}$ and the reduced frequency ω^* the interpolation model has to predict the entire vector $\mathbf{\Delta c_p}$ at once. The goal of an interpolation model is to find a mapping between a set of input vectors $\mathcal{X} = \{(\boldsymbol{\kappa}_1, \omega_1^*)^\top, ..., (\boldsymbol{\kappa}_n, \omega_n^*)^\top\}$ and a set of scalar output values $\mathcal{Y} = \{\Delta c_{p,1}, ..., \Delta c_{p,n}\}$. The prediction of multiple target quantities as $\mathbf{\Delta c_p}$ is possible by defining a single interpolation model for each quantity. Assuming Gaussian noise δ in the data an interpolation model can be written as:

$$\Delta c_p(\boldsymbol{\kappa}, \omega^*) = g(\boldsymbol{\kappa}, \omega^*) + \delta \tag{3.1}$$

$$\text{and } g(\boldsymbol{\kappa}, \omega^*) = \phi(\boldsymbol{\kappa}, \omega^*)^\top \mathbf{u} \tag{3.2}$$

with basis functions $\phi(\boldsymbol{\kappa}, \omega^*) = (\phi_1(\boldsymbol{\kappa}, \omega^*), ..., \phi_m(\boldsymbol{\kappa}, \omega^*))^\top$ and the weight vector which is usually defined as $\mathbf{u} \in \mathbb{R}^m$. Since the goal is to predict complex values in this study it is defined as $\mathbf{u} \in \mathbb{C}^m$. The nature of the interpolation can be altered by the choice of the basis function [27].

Radial basis function (RBF) interpolation allows to find a high-order description of high dimensional data [28]. Gaussian process regression is a non-parametric probabilistic method for modelling and predicting the distribution of a dependent variable based on a set of input data points, utilizing Gaussian processes to capture uncertainty and provide regression analysis.

RBF Interpolation
Radial Basis Function (RBF) Interpolation is a technique that uses a combination of radial basis functions centred at data points to approximate and interpolate high dimensional data at non-data points, where the influence of each data point diminishes with distance from the centre [28, 29].

RBFs can be described as:

$$\phi_i(\boldsymbol{\kappa}, \omega^*) = \varphi(||(\boldsymbol{\kappa}, \omega^*)^\top - (\boldsymbol{\kappa}_\mathbf{i}, \omega_i^*)^\top||), i = 1, ..., n \tag{3.3}$$

where function φ can be chosen. Further theory can be found in [29]. Four different RBF types are implemented in SMARTy, they can be found in table A.1 [27].

Gaussian Process Regression
Gaussian process regression (Kriging) is a probabilistic method for modelling and predicting the distribution of a dependent variable based on a set of input data points, utilizing Gaussian processes to capture uncertainty and provide a flexible framework for regression analysis [27, 30].

The functional relationship between the scalar output Δc_p and the input vector $(\boldsymbol{\kappa}, \omega^*)^\top \in \mathbb{R}^n$ can be defined as

$$\Delta c_p(\boldsymbol{\kappa}, \omega^*) = f(\boldsymbol{\kappa}, \omega^*)\beta + \epsilon(\boldsymbol{\kappa}, \omega^*). \tag{3.4}$$

It is defined by the regression model $f\colon \mathbb{R}^n \to \mathbb{R}^p$ with unknown parameters $\beta \in \mathbb{R}^p$ and a Gaussian process $\epsilon(\boldsymbol{\kappa}, \omega^*)$, which is defined by an expected value, variance, and covariance. Since this is not the focus of this work, more theory can be found in [27], [31], [32], and [33]. Kriging models can be altered by using different correlation kernels. The kernels in table A.2 are used in the following study.

3.3.2 Fully-Connected Neural Network

A fully-connected neural network (FCNN) is also called a multi-layer perceptron (MLP). It is based on the simpler idea of a *perceptron*. The FCNN in this paper uses $w_i(\boldsymbol{\kappa})$, $\boldsymbol{\kappa}$, ω^* and (c_1^i, c_2^i), (v_1^i, v_2^i) as geometrical features at one node to predict $\Re(\Delta c_p)_i$ and $\Im(\Delta c_p)_i$.

The index i refers to values at a specific node. A perceptron multiplies an input vector $\mathbf{x} \in \mathbb{R}^n$ with weights $\mathbf{u} \in \mathbb{R}^n$, adds a bias $b \in \mathbb{R}$ and passes it through a (non-linear) activation function η in order to produce an output $y \in \mathbb{R}$ [34]:

$$y = \eta(\mathbf{u}^\top \mathbf{x} + b) \tag{3.5}$$

In an MLP multiple *neurons* as in equation 3.5 are organized in l layers. Each unit in one layer is connected to every unit in the subsequent layer. The output of the $(i+1)$-th layer $\mathbf{h}^{(i+1)} \in \mathbb{R}^p$ with p neurons is computed on basis of the i-th layer $\mathbf{h}^{(i)} \in \mathbb{R}^q$ with q neurons. All weights are organized in a weight matrix $\mathbf{W}^{(i)} \in \mathbb{R}^{p,q}$ and the biases organized in the bias vector $\mathbf{b}^{(i)} \in \mathbb{R}^q$:

$$\mathbf{h}^{(i)} = \eta^{(i)} \left(\mathbf{W}^{(i)} \mathbf{h}^{(i-1)} + \mathbf{b}^{(i)} \right) \tag{3.6}$$

By defining the input vector $(w_i(\boldsymbol{\kappa}), \boldsymbol{\kappa}, \omega^*, (c_1^i, c_2^i), (v_1^i, v_2^i))^\top \in \mathbb{R}^n$ as the first layer and the output $(\Re(\Delta c_p)_i, \Im(\Delta c_p)_i)^\top \in \mathbb{R}^m$ as the last layer, equation 3.6 can fully describe an MLP [23].

While the number of inputs n and outputs m, and therefore the size of the first and last layer are defined by the data, the number of hidden layers l and the number of neurons per layer can be changed. Both are hyperparameters and can strongly influence the performance of a neural network. The model uses the ReLU activation function according to [35]. It is trained iteratively using backpropagation according to [36].

3.3.3 Message Passing Neural Network

As a basis for GNNs message passing neural networks (MPNNs) must be introduced. MPNNs assume graph data and compute a new node feature vector $\mathbf{h}_i^{t+1} \in \mathbb{R}^{d_{1,t+1}}$ at node v_i on basis of the node feature vector of the previous step at node v_i $\mathbf{h}_i^t \in \mathbb{R}^{d_{1,t}}$, and at all the neighbours $v_j \in \mathcal{N}(v_i)$ $\mathbf{h}_j^t \in \mathbb{R}^{d_{1,t}}$, and the edge feature vectors $\mathbf{e}_{j,i}$ for all edges connected to v_i.

With the input features as starting point $\mathbf{h}_i^0$ the computation is done in the following three steps [17]:

1. In the *message passing* step a message vector $\mathbf{m}_{j,i}^{t+1}$ is passed from each neighbor $v_j \in \mathcal{N}(v_i)$ to node v_i using function M_t with learnable parameters. The information of $\mathbf{h}_i^t$, $\mathbf{h}_j^t$, and $\mathbf{e}_{j,i}$ is taken into account:

$$\mathbf{m}_{j,i}^{t+1} = M_t \left(\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{j,i} \right) \tag{3.7}$$

2. The messages of all neighbours are then collected using a permutation invariant function in the *aggregation* step:

$$\mathbf{m}_i^{t+1} = \bigoplus_{v_j \in \mathcal{N}(v_i)} \mathbf{m}_{j,i}^{t+1} \tag{3.8}$$

3. Finally, an *update* of the node feature vector is generated using function U_t with learnable parameters:

$$\mathbf{h}_i^{t+1} = U_t \left(\mathbf{h}_i^t, \mathbf{m}_i^{t+1} \right) \tag{3.9}$$

The three steps result in the following equation:

$$\mathbf{h}_i^{t+1} = U_t \left(\mathbf{h}_i^t, \bigoplus_{v_j \in \mathcal{N}(v_i)} M_t \left(\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{j,i} \right) \right) \tag{3.10}$$

3.3.4 Graph Convolutional Network

The models in this subsection and subsection 3.3.5 assume that data in the form of attributed graphs is present (see sec. 3.2). The first approach is the graph convolutional network (GCN) which is introduced in two versions. For the first in addition to the geometric information (c_1^i, c_2^i) and (v_1^i, v_2^i) all available flow features $\mathbf{w}(\boldsymbol{\kappa})$, $\boldsymbol{\kappa}$ and ω^* are added to the node feature vector. This model uses local flow information and is therefore called local model. The second only uses global flow features: $\boldsymbol{\kappa}$ and ω^*. It is referred to as global model. Both models predict the real and imaginary part of the increment to the pressure coefficient at each node which results in the two vectors $\Re(\boldsymbol{\Delta}\mathbf{c_p})$ and $\Im(\boldsymbol{\Delta}\mathbf{c_p})$.

Residual Gated Graph ConvNets
The structure of an MPNN can be altered by the choice of the functions M_t, U_t, and the aggregation function. The *Residual Gated Graph ConvNet* proposed in [19] uses the sigmoid logistic function σ, linear transformation with the weights $\mathbf{W}_2$, $\mathbf{W}_3$, and $\mathbf{W}_4$, as well as the pointwise product $\odot$ for the message passing. The aggregation step uses the sum. In the update step, an activation function η and the weight matrix $\mathbf{W}_1$ is used. The resulting function is the following:

$$\mathbf{h}_i^{t+1} = \eta \left(\mathbf{W}_1 \mathbf{h}_i^t + \sum_{v_j \in \mathcal{N}(v_i)} \sigma \left(\mathbf{W}_3 \mathbf{h}_i^t + \mathbf{W}_4 \mathbf{h}_j^t \right) \odot \mathbf{W}_2 \mathbf{h}_j^t \right) \tag{3.11}$$

As one can see in the above equation, this approach does not make use of the edge features.

The selected version of the model works according to equation 3.11 with ReLU as activation function. Throughout this work models based purely on graph convolutional layers are called GCNs and are not to be confused with the simpler most common method that uses a message passing based on first-order approximation of spectral graph convolutions [18]. The number of hidden layers $1 \leq t \leq l$ and the number of neurons per hidden layer alter the topology of the GCN. They can be used as hyperparameters.

3.3.5 Graph Network Simulator

The graph network simulator (GNS) combines concepts of the MLP and the GCN model and was proposed in [37]. In contrast to the GCN model, the GNS model also considers the graph edge features. The model is built as an encoder-processor-decoder structure.

Given an attributed graph $\mathcal{G} = (V, \mathbf{A}, \mathbf{X}, \mathbf{E})$ as in subsection 3.3.4 an encoder multi-layer perceptron MLP_E is used to produce an encoded representation $\mathbf{h}_i^0 \in \mathbb{R}^{d_l}$ of the node features $\mathbf{x}_i$ that are defined in $\mathbf{X}$, with d_l latent dimensions. As for the GCN method two models with different inputs are compared: the first using local and global flow features $\mathbf{x}_i = (w_i(\boldsymbol{\kappa}), \boldsymbol{\kappa}, \omega^*, (c_1^i, c_2^i), (v_1^i, v_2^i))^\top$ and the second only using global flow features $\mathbf{x}_i = (\boldsymbol{\kappa}, \omega^*, (c_1^i, c_2^i), (v_1^i, v_2^i))^\top$. Both use the same edge features $\mathbf{e}_{i,j} = (r_{j,i}, \theta_{j,i}, \tilde{a}_{j,i}, \tilde{\theta}_{j,i})$.

$$\mathbf{h}_i^0 = \mathrm{MLP_E}(\mathbf{x}_i) \tag{3.12}$$

After application of the encoder to all nodes a new attributed graph $\mathcal{G}^0 = (V, \mathbf{A}, \mathbf{X}^0, \mathbf{E})$ is obtained.

Subsequently, a GCN with l message passing layers is used to process the graph data. In the $1 \leq t \leq l$ layers the message passing is done using an MLP of the following structure for the function M_t from equation 3.7:

$$\mathbf{m}_{j,i}^{t+1} = \mathrm{MLP}_{M_t}\left(\mathbf{h}_i^t \oplus \left(\mathbf{h}_j^t - \mathbf{h}_i^t\right) \oplus \mathbf{e}_{j,i}\right) \tag{3.13}$$

The aggregation (eq. 3.8) is done by taking the mean of all messages:

$$\mathbf{m}_i^{t+1} = \frac{1}{|\mathcal{N}(v_i)|} \sum_{j \in \mathcal{N}(v_i)} \mathbf{m}_{j,i}^{t+1} \tag{3.14}$$

where $\mathcal{N}(v_i)$ is the set of neighbours of node v_i. The update (eq. 3.9) is then computed using another MLP:

$$\mathbf{h}_i^{t+1} = \mathrm{MLP}_{U_t}\left(\mathbf{h}_i^t \oplus \mathbf{m}_i^{t+1}\right) \tag{3.15}$$

The resulting attributed graph $\mathcal{G}^l = (V, \mathbf{A}, \mathbf{X}^l, \mathbf{E})$ is finally decoded using a decoder MLP. This results in the following prediction:

$$(\Re(\Delta c_p)_i, \Im(\Delta c_p)_i)^\top = \mathrm{MLP}_\mathrm{D}(\mathbf{h}_i^l) \tag{3.16}$$

The topology of the encoder, processor, and decoder can be altered by several parameters. Those can be used as hyperparameters.

		Baseline	ML Comparison	GNN			
		Interpolation	**FCNN**	**GCN**		**GNS**	
				Global	Local	Global	Local
Features	Global	$\boldsymbol{\kappa}$	$\boldsymbol{\kappa}$	$\boldsymbol{\kappa}$	$\boldsymbol{\kappa}$	$\boldsymbol{\kappa}$	$\boldsymbol{\kappa}$
	Local		$w_i(\boldsymbol{\kappa})$		$\mathbf{w}(\boldsymbol{\kappa})$		$\mathbf{w}(\boldsymbol{\kappa})$
	Geometric		(c_1^i, c_2^i), (v_1^i, v_2^i)	$(\mathbf{c_1}, \mathbf{c_2})$, $(\mathbf{v_1}, \mathbf{v_2})$	$(\mathbf{c_1}, \mathbf{c_2})$, $(\mathbf{v_1}, \mathbf{v_2})$	$(\mathbf{c_1}, \mathbf{c_2})$, $(\mathbf{v_1}, \mathbf{v_2})$, $(\mathbf{r_j}, \theta_\mathbf{j})$, $(\tilde{\mathbf{a}}_\mathbf{j}, \tilde{\theta}_\mathbf{j})$	$(\mathbf{c_1}, \mathbf{c_2})$, $(\mathbf{v_1}, \mathbf{v_2})$, $(\mathbf{r_j}, \theta_\mathbf{j})$, $(\tilde{\mathbf{a}}_\mathbf{j}, \tilde{\theta}_\mathbf{j})$
Targets		$\mathbf{\Delta c_p}$	$\Re(\Delta c_p)_i$, $\Im(\Delta c_p)_i$	$\Re(\mathbf{\Delta c_p})$, $\Im(\mathbf{\Delta c_p})$	$\Re(\mathbf{\Delta c_p})$, $\Im(\mathbf{\Delta c_p})$	$\Re(\mathbf{\Delta c_p})$, $\Im(\mathbf{\Delta c_p})$	$\Re(\mathbf{\Delta c_p})$, $\Im(\mathbf{\Delta c_p})$

Table 3.1: Overview of the features and targets used for the different models using the generic variables introduced in chapter 2. The differentiation between vectors and values with the suffix i gives information on whether the approach works for all nodes at once or node-wise.

3.3.6 Model Summary

Table 3.1 summarizes the features and targets used by the different approaches using the formal description introduced in chapter 2. After the description of a specific test case in chapter 4 table 5.1 in chapter 5 states the actual variables used in this study.

3.3.7 Advantages and Disadvantages

Including the differentiation between global and local models subsections 3.3.1 to 3.3.5 define six different models. Table 3.2 gives the advantages and disadvantages of the models that are known upfront. More insight is gained throughout this paper.

Model	**Advantages**	**Disadvantages**
Interpolation	- Easiest approach - No steady solution required - Able to predict complex values	- No usage of local flow information - No usage of geometrical information
FCNN	- Usage of local flow information - FCNN widely used	- Model must be evaluated at each node -No usage of neighbourhood information
GCN Global	- No steady solution required - Uses geometrical information	- Graph input required
GCN Local	- Usage of local flow information - Uses geometrical information	- Graph input required
GNS Global	- No steady solution required - Uses most geometrical information - Encoder-Processor-Decoder structure gives latent feature representation	- Graph input required - New, rarely used approach
GNS Local	- Usage of local flow information - Uses most geometrical information - Encoder-Processor-Decoder structure gives latent feature representation	- Graph input required - New, rarely used approach

Table 3.2: Advantages and disadvantages of the different models

3.3.8 Software Implementation

The methods explained in the subsections 3.3.1, 3.3.2, 3.3.4 and 3.3.5 are implemented in the DLR-SMARTy-Toolbox as the modules `Interpolation`, `FullyConnectedNN`, `GCNModel` and `GNSModel`. These models work based on the python packages PyTorch (`torch`) and PyTorchGeometric (`pyg`) [27, 38, 39].

Chapter 4

Test Case

In this chapter, the test case which is used in this study is explained. Therefore, the goal is to create a dataset containing aerodynamic data in the frequency domain for different conditions on which the methods explained in section 3.3 can be compared.

4.1 NACA 64A010 Airfoil

As a two-dimensional test case, the NACA 64A010 airfoil is selected. It is a symmetric airfoil according to the NACA 6A-Series [40]. A two-dimensional test case is used to limit the complexity of the first investigation of the methods. Subsequently, more complex test cases as a three-dimensional wing or a full aircraft are conceivable. The specific airfoil is selected since it has previously been used to investigate unsteady aerodynamics in [4] and [41].

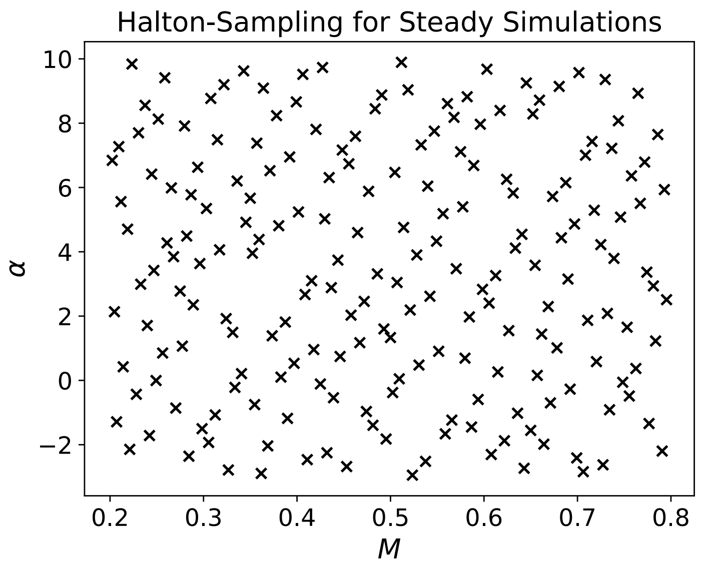


Figure 4.1: Sampling of M and α as input variables for steady-state simulations

4.2 Design of Experiments

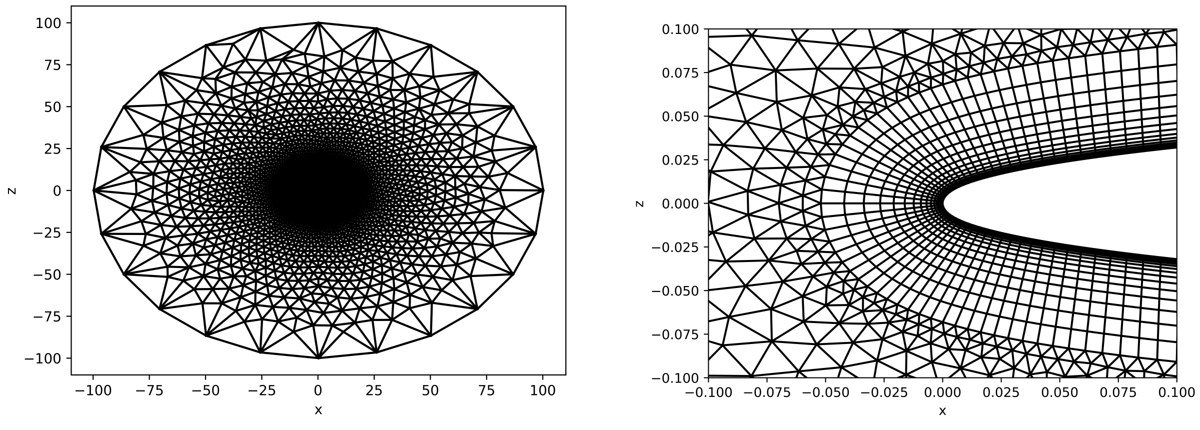
The design of experiments (DoE) is done in two steps, first, a set of steady simulations is computed. In the second step, they are used as input for the LFD simulations. The Halton sequence is a quasi-random number sequence that allows the creation of a reproducible set of M and α combinations with a random appearance [42]. Furthermore, it is implemented in the DLR SMARTy-Toolbox [27]. For the sampling of the steady flow solution, the Mach number and the angle of attack are used as independent flow variables $\boldsymbol{\kappa} = (M, \alpha)^\top$ since they significantly influence the aerodynamic behaviour of an airfoil. 200 combinations of the Mach number in the range $[0.2, 0.8]$ and the angle of attack in the range $[-3°, 10°]$ are created. The ranges are selected this way to cover a wide range of possible flight conditions in the subsonic regime. The distribution can be seen in figure 4.1.

Next to the output of a steady-state solution, the LFD-solver expects other input variables. The type of excitation defines how the system is excited. For this study, only excitation due to gusts are considered. As explained in section 2.1 the reduced frequency ω^* influences the gust shape and is used as the third input variable. For each steady-state flow solution, respectively for each combination of M and α, a set of 20 equidistantly distributed reduced frequencies in the range [0, 4] completes the three-dimensional sample space. The selected range covers a wide range of unsteady behaviour. Consequently, 4000 samples with different combinations of M, α and ω^* can be produced.

Using the same set of reduced frequencies for each combination of Mach number and angle of attack can be explained by examining a possible use case: given a steady-state flow solution and a set of reduced frequencies, unsteady flow solutions have to be predicted.

4.3 Mesh Description and Graph Creation

For the CFD simulations in this study the mesh consists of a structured mesh close to the airfoil (see figure 4.2b) and an unstructured mesh for the rest of the computational domain (see figure 4.2a). The flow solution is computed in a radius of 100 chord lengths around the airfoil.



(a) Entire domain with unstructured mesh **(b)** Structured mesh close to the airfoil

Figure 4.2: Mesh used in the study

The present mesh is used to create an attributed graph in accordance with section 3.2. The result is a graph $\mathcal{G}_{\text{all}} = (V_{\text{all}}, \mathbf{A}_{\text{all}})$ with $n_{\text{all}} = 10727$ nodes and $m_{\text{all}} = 54314$ edges. However, since the goal is to predict the surface pressure increments only the surface graph is used. The result is a graph $\mathcal{G}_{\text{surface}} = (V_{\text{surface}}, \mathbf{A}_{\text{surface}})$ with $n_{\text{surface}} = 200$ nodes and $m_{\text{surface}} = 400$ edges. Subsequently, when referring to the graph used in this study, the index surface is neglected. Adding the node and edge features results in an attributed graph $\mathcal{G} = (V, \mathbf{A}, \mathbf{X}, \mathbf{E})$ with $V \in \mathbb{R}^{200}$, $\mathbf{A} \in \mathbb{R}^{200,200}$, $\mathbf{X} \in \mathbb{R}^{200,4}$ and $\mathbf{E} \in \mathbb{R}^{400,4}$.

4.4 Flow Simulations

Based on the DoE described in section 4.2 first 200 steady simulations and subsequently 4000 LFD simulations are executed using the DLR TAU code [22].

4.4.1 Steady Simulations

As described the steady flow simulations use the Mach number and the angle of attack as independent inputs. Selected parameters are explained here. The Reynolds number is fixed to $Re = 12.5 \cdot 10^6$. As explained in the appendix A.2.1 this leads to simulations at *different altitudes*. The Spalart-Allmaras one-equation turbulence model is used [43]. The iterative solving is done for a maximum of 10000 time steps or until the density residual `Rrho` has reach a value of $1 \cdot 10^{-10}$ or smaller [22]. The solver outputs values on the airfoil surface (surface solution) and for the volume (volume solution). It is further looked at in section 4.5.

4.4.2 Linear Frequency Domain Simulations

The LFD solver uses the steady solutions and the reduced frequency as variable inputs. The reduced frequency is used to define an excitation of the type gust. The gust direction is vertical. The simulation iteratively runs for a maximum of 10000 iterations. Furthermore, the total linear residual from the Generalized Minimal Residual algorithm (GMRes) solver `Res-GMRES` is monitored [22]. The simulation is stopped once it reaches a value of $1 \cdot 10^{-11}$ or lower. Beyond that, the LFD solver uses, where applicable, similar settings as the steady solver.

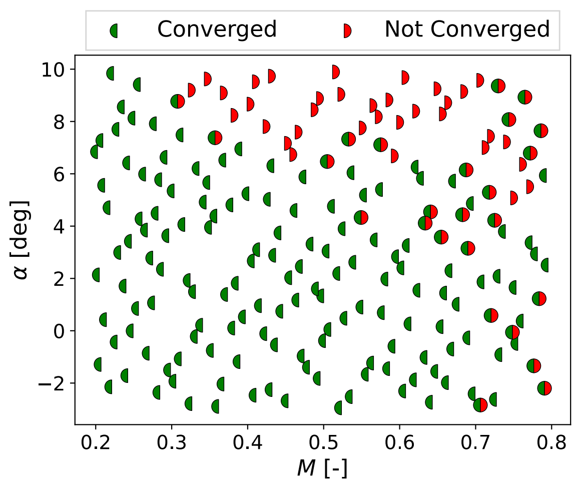


Figure 4.3: Converged and not converged samples in the M-α-plane

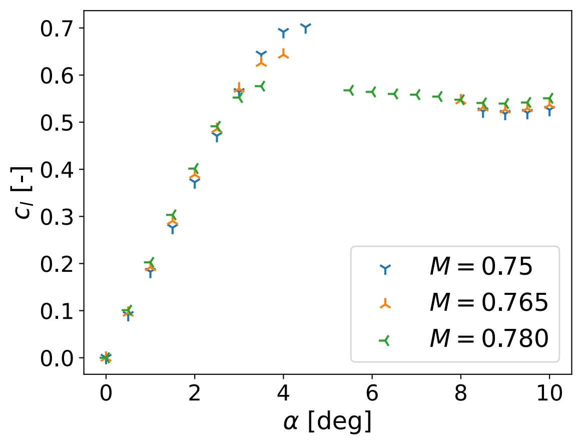


Figure 4.4: c_l vs α for different Mach numbers

4.4.3 Convergence

Not all simulations reached the convergence criteria defined in subsections 4.4.1 and 4.4.2 after the maximum number of iterations. Simulations that are not converged should be excluded from the dataset since they do not represent a physically correct solution. Of the 4000 sample points generated in the DoE 771 did not reach the convergence criteria in either the steady simulation, the LFD simulation, or both. The distribution of the resulting 3229 samples in the M-α-plane can be seen in figure 4.3. As an overall tendency simulations with high values for M and α did not converge more often. However, in the top right corner, a small region with converged simulation at high M and high α can be found. These samples are investigated by examining the coefficient of lift c_l over the angle of attack α for three Mach numbers in the affected area.

Figure 4.4 shows the lift coefficient for different combinations of Mach and angle of attack. Only converged simulations are shown. It can be seen that simulations converge with increasing alpha until a $c_{l,\max}$ is reached. In the following stall region, the simulations do not converge. At higher Mach numbers converged samples can be found again. They are separated from the rest of the dataset and since they refer to a usually unwanted flight condition (post-stall) they are excluded from the dataset.

The resulting dataset contains 3144 samples and can be seen in figure 4.5. It has to be mentioned that this is a two-dimensional figure representing a three-dimensional space. Consequently, the figure gives no information about how many LFD simulations converged at each M-α combination.

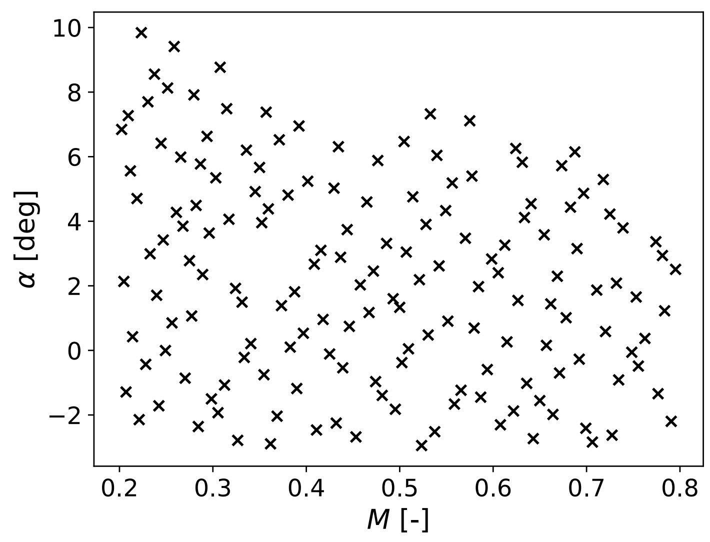


Figure 4.5: Final set of samples in the M-α-plane

4.5 Features

Possible features that can be used for the construction of a surrogate model are the outputs from the steady simulations as well as M, α, and ω^*. The goal of this section is to find the smallest number of targets that deliver the highest amount of information.

The steady simulation delivers the following values: density ρ, velocities u, v, w, pressure p, and two values from the turbulence model $\tilde{u}_{SA}$, ν_t. As explained in subsection 2.2 only the values on the surface of the airfoil are used. For viscous fluids the no-slip condition on a solid boundary is true [44]. Only values on the airfoils boundary are taken into account some of the possible features are zero: u, v, w, $\tilde{u}_{SA}$, and ν_t. Since they do not deliver any information they are eliminated.

The remaining possible features are plotted against each other in figure A.1. As one can expect, the distribution of M, α, and ω^* are distributed according to the DoE since they are the independent variables. Pressure p and density ρ have an almost linear relationship (fig. A.1; bottom row, fourth plot from the left). Consequently, only one feature is used since the informational content is similar for both. To choose the feature with more information the variance of both is computed and divided by the mean:

$$\frac{\sigma_p^2}{\mu_p} = 0.01634 \text{ and}$$
$$\frac{\sigma_\rho^2}{\mu_\rho} = 0.01603$$

The value is higher for pressure. Thus, four possible features are determined. M, α, and ω^* are called *global* features from now on since they apply for all locations on the airfoils. p is called a *local* feature because a specific value for each location on the airfoil is used.

4.6 Targets

As the unsteady pressure coefficient increments Δc_p are complex values in the frequency domain approach, the features must be handled accordingly. The Interpolation model that is used as a baseline can predict complex numbers. However, the DL are not able to directly predict complex values with their current implementation in the SMARTy-Toolbox. Complex-valued neural networks are a current research subject and could be a future approach [45, 46]. For this study, the complex numbers are separated into their real and imaginary part. Subsequently, at each location on the airfoil, two targets have to be predicted.

4.7 Resulting Dataset

The data is collected in a dataset with 3144 samples (according to 4.4.3). Each sample has three global features M, α, ω^*, and one local feature p. Each sample has two targets $\Re(\Delta c_p)$, and $\Im(\Delta c_p)$. The local feature and the targets each have 200 individual values for each airfoil location for each sample. Furthermore, each sample is supplemented by the geometrical information that is stored in the graph (see sec. 4.3). To investigate the influence of local and global features subsets of the entire dataset can be used. A detailed description of the used data for each model is given in section 5.1.

4.8 Train-Validation-Test-Split

The resulting dataset is split into three partitions:

1. The data that is used to train the models is called *training* data.
2. For iterative training processes the model is evaluated during the training process using the *validation* data.
3. *Test* data is used to finally evaluate the model on unseen data.

Different splitting strategies are possible. A possible use case for the resulting surrogate models is to predict the target quantities for multiple reduced frequencies given a steady solution. Consequently, the splitting is only done between the steady samples: all samples (at different ω^*) for a combination of M and α are either in the training, validation, or test dataset. 60% of the data is used as training data, 20% of the data is used as validation data, and 20% of the data is used as test data. The split is chosen since the available dataset can be considered as large. This allows to assign a big portion of the data for validation and testing. Due to the elimination of not converged samples the number of samples for a combination of M and α can vary. This results in the following split:

- Number training samples: 1875
- Number validation samples: 634
- Number test samples: 635

The split visualized in the steady plane can be seen in 4.6. From the visualization, it becomes clear, that the training, validation, and test samples are distributed evenly across the M-α-plane.

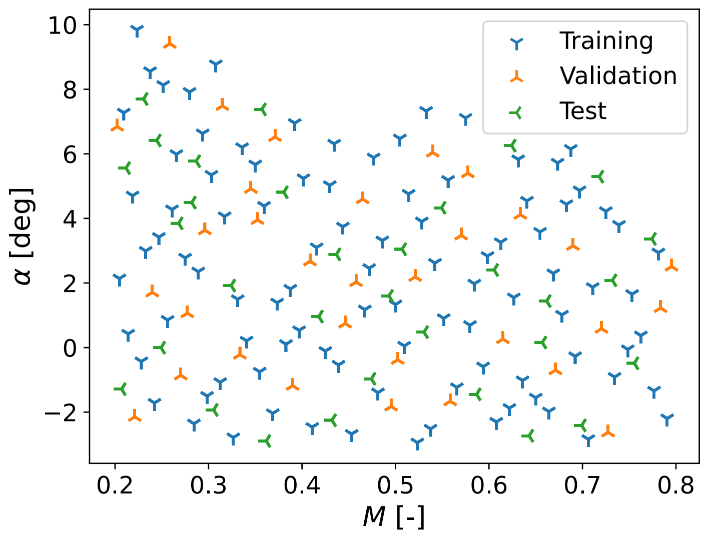


Figure 4.6: Train-Validation-Test-Split in the M-α-plane

Chapter 5

Model Selection

In this section models based on the methods explained in section 3.3 are constructed and optimized for the test case (chapter 4).

5.1 Description of Models

After in chapter 4 features are selected, M and α are used as independent and global flow features κ. The vector containing the pressure at the surface $\mathbf{p}(M, \alpha)$ is used as local flow feature $\mathbf{w}(\kappa)$. The resulting features and targets are presented in table 5.1.

		Baseline	ML Comparison	GNN			
		Interpolation	**FCNN**	**GCN**		**GNS**	
				Global	Local	Global	Local
Features	Global	M, α, ω^*	M, α, ω^*	M, α, ω^*	M, α, ω^*	M, α, ω^*	M, α, ω^*
	Local		p_i		$\mathbf{p}$		$\mathbf{p}$
	Geometric		(c_1^i, c_2^i), (v_1^i, v_2^i)	$(\mathbf{c_1}, \mathbf{c_2})$, $(\mathbf{v_1}, \mathbf{v_2})$	$(\mathbf{c_1}, \mathbf{c_2})$, $(\mathbf{v_1}, \mathbf{v_2})$	$(\mathbf{c_1}, \mathbf{c_2})$, $(\mathbf{v_1}, \mathbf{v_2})$, $(\mathbf{r_j}, \theta_\mathbf{j})$, $(\tilde{\mathbf{a}}_\mathbf{j}, \tilde{\theta}_\mathbf{j})$	$(\mathbf{c_1}, \mathbf{c_2})$, $(\mathbf{v_1}, \mathbf{v_2})$, $(\mathbf{r_j}, \theta_\mathbf{j})$, $(\tilde{\mathbf{a}}_\mathbf{j}, \tilde{\theta}_\mathbf{j})$
Targets		$\mathbf{\Delta c_p}$	$\Re(\Delta c_p)_i$, $\Im(\Delta c_p)_i$	$\Re(\mathbf{\Delta c_p})$, $\Im(\mathbf{\Delta c_p})$	$\Re(\mathbf{\Delta c_p})$, $\Im(\mathbf{\Delta c_p})$	$\Re(\mathbf{\Delta c_p})$, $\Im(\mathbf{\Delta c_p})$	$\Re(\mathbf{\Delta c_p})$, $\Im(\mathbf{\Delta c_p})$

Table 5.1: Overview of the features and targets used for the different models. The differentiation between vectors and values with the suffix i gives information on whether the approach works for all nodes at once or node-wise.

It is known that $\mathbf{\Delta c_p}$ does only consist of real values when $\omega^* = 0$. This knowledge is used and implemented into the prediction function of all models. This function is only called for prediction, but not during training.

5.2 Hyperparameter Optimization

Hyperparameter optimization (HPO) is used to find a set of hyperparameters that leads to the highest possible prediction accuracy.

5.2.1 Method

The python framework `optuna` is used for optimization [47]. It uses the tree-structured parzen estimator approach to perform sequential model-based global optimization of a fitness function [48]. The framework allows to define lists or ranges for multiple independent variables (see subsec. 5.2.2).

In this study, the goal is to find the set of hyperparameters that minimizes the mean absolute error (MAE) as defined in section 6.1. Therefore, each model is trained for a number of trials. Subsequently, the MAE is evaluated using the validation dataset, because the test dataset is only used in the final evaluation of the models. Finally, the model with the lowest MAE is selected. Each modelling approach is optimized using 50 trials. The DL approaches are trained for 2000 epochs using the training dataset. During training the MAE is monitored on the validation dataset. After 50 epochs without improvement, the training is stopped. This reduces the computational cost of the already computationally expensive HPO.

5.2.2 Optimization Variables

This subsection gives an overview of which hyperparameters are used in the optimization. Therefore, some of the capabilities of the optimization framework must be introduced:

- `suggest_categorical`: variable is chosen from a list. The entries are not bound to a specific data type.
- `suggest_int`: integer from a range is chosen. Additionally, the range can be defined in a logarithmic domain.
- `suggest_float`: float from a range is chosen. Additionally, the range can be defined in a logarithmic domain.

Note that all models have a very large number of hyperparameters. Using all of them as variables is not feasible in the frame of this paper. In the following important parameters that define the topology and learning characteristics of each model are selected and shortly explained. Further parameters are not explained in this text.

Interpolation
For the Interpolation models multiple approaches (see tab. A.1 and A.2) are defined through the variable `kernel`. The `augmentation` parameter defines whether the Interpolation model uses a trend function. The `regularization` parameter can turn the interpolation approach into a regression approach. `scale` defines whether the features and targets are scaled (see appendix A.3.1). Further details about the defined possibilities for the hyperparameters can be found in table A.3 in the appendix.

FCNN
The FCNNs topology is defined by the `numHiddenLayers`. Each layer contains `numNeurons-PerLayer` neurons. For the training process, an initial `learningRate` defines how fast the models' weights are adapted. Throughout the training process, the `learningRate` is adapted in two steps by multiplying it with the factor `gamma`. This multiplication is done at training epoch `milestoneFactor` and at `milestoneFactor`·2. Further details regarding the defined possibilities for the hyperparameters can be found in table A.4 in the appendix.

GCN
The same hyperparameters as for the FCNN can be defined for the GCN Global and GCN Local models. Further details about the defined possibilities for the hyperparameters can be found in tables A.5 and A.6 in the appendix.

GNS
The GNS models are built in an encoder-processor-decoder structure. `numHiddenLayers-Encoder` defines the number of hidden layers in the encoder. `numLayersProcessor` is the number of layers in the processor. `numNeurons` defines the number of neurons in each layer in the encoder, the processor and the decoder. The parameters defining the training process (`learningRate`, `gamma`, `milestoneFactor`) are the same as for the other DL models. Further details about the defined possibilities for the hyperparameters can be found in tables A.7 and A.8 in the appendix.

5.2.3 Results

Figures A.2 to A.7 can be found in the appendix. The optimization history shows the evolution of the objective value over the number of trials. It can be seen that the number of 50 trials is enough for all models to converge. The hyperparameter importance is calculated according to [49]. It can be seen that for all models the hyperparameters that define the training process, e.g. `learningRate`, are more important than the parameters that define the topology of the models. The resulting values for the different hyperparameters of all models can be found in tables A.3 to A.8. Using these hyperparameters the models are evaluated in the next chapter.

Chapter 6

Results

In this chapter, the optimized models are evaluated (sec. 6.2) on different metrics that are introduced in section 6.1. Furthermore, three investigations are conducted: in section 6.3 the influence of the size of the dataset is evaluated, in section 6.4 the models are evaluated on unseen reduced frequencies, and in section 6.5 investigates if training the models on high Mach number cases improves the prediction capabilities in that regime.

6.1 Metrics

Different metrics are used to evaluate the accuracy of the Δc_p-predictions of the different modelling approaches described in section 3.3. All metrics compare the true values $\Delta c_{p,\text{true}}$ to the predicted values $\Delta c_{p,\text{pred}}$. Four different metrics are used. For each sample (Δc_p-distribution around an airfoil) the different metrics compute one value representing the error of n Δc_p-values. When dealing with complex values and not stated differently the metrics are computed for the real and imaginary parts separately, the outcome is then averaged.

Mean Absolute Error
The mean absolute error (MAE) is the mean of the $L1$-norm for all n values of one sample:

$$\text{MAE}(\Delta c_{p,\text{true}}, \Delta c_{p,\text{pred}}) = \frac{1}{n}\sum_{i=0}^{n-1} |\Delta c_{p,\text{true},i} - \Delta c_{p,\text{pred},i}| \tag{6.1}$$

Mean Squared Error
The mean squared error (MSE) is similar to the MAE, but uses the $L2$-norm instead:

$$\text{MSE}(\Delta c_{p,\text{true}}, \Delta c_{p,\text{pred}}) = \frac{1}{n}\sum_{i=0}^{n-1} (\Delta c_{p,\text{true},i} - \Delta c_{p,\text{pred},i})^2 \tag{6.2}$$

Since MSE uses the square of differences it is more sensitive to outliers than MAE.

Coefficient of Determination
The coefficient of determination (also called R^2-score) gives the proportion of variance of the original dataset that is explained by the prediction. Perfect predictions lead to the maximum value of 1, whereas a model that predicts the mean regardless of the inputs leads to a score of 0:

$$R^2(\Delta c_{p,\text{true}}, \Delta c_{p,\text{pred}}) = 1 - \frac{\sum_{i=1}^{n}(\Delta c_{p,\text{true},i} - \Delta c_{p,\text{pred},i})^2}{\sum_{i=1}^{n}(\Delta c_{p,\text{true},i} - \overline{\Delta c_{p,\text{true}}})^2} \tag{6.3}$$

where $\overline{\Delta c_{p,\text{true}}} = \frac{1}{n}\sum_{i=1}^{n} \Delta c_{p,\text{true},i}$.
For this and the two above metrics the python package `scikit-learn` is used [50].

Relative Error
The MAE is dependent on the range of the considered values. Thus, samples with higher values are more likely to lead to higher errors. By using a relative error as in [51] this can be mitigated. The outcome of equation 6.1 is divided by the range of the true values of the sample:

$$\text{Relative Error} = \frac{\text{MAE}(\Delta c_{p,\text{true}}, \Delta c_{p,\text{pred}})}{\max\limits_{1\leq i\leq n} \Delta c_{p,\text{true},i} - \min\limits_{1\leq i\leq n} \Delta c_{p,\text{true},i}} \tag{6.4}$$

6.2 Optimized Models

The models are trained for 2000 epochs each. Due to convergence issues, the GNS Local model is trained for 4000 epochs (more explanation in section 6.3). Afterwards, the model state from the training epoch with the lowest validation error is used[1]. The hyperparameters are set according to the results of the HPO section 5.2.

	MAE	**MSE**	**R^2**	**relative Error**
Interpolation	$9.875 \cdot 10^{-4}$	$3.510 \cdot 10^{-5}$	0.773	0.0160
FCNN	$5.877 \cdot 10^{-4}$	$2.336 \cdot 10^{-5}$	0.849	0.0089
GCN Global	$5.683 \cdot 10^{-4}$	$1.224 \cdot 10^{-5}$	0.921	0.0097
GCN Local	$\mathbf{5.473 \cdot 10^{-4}}$	$1.692 \cdot 10^{-5}$	0.890	**0.0083**
GNS Global	$5.785 \cdot 10^{-4}$	$1.337 \cdot 10^{-5}$	0.913	0.0097
GNS Local	$6.157 \cdot 10^{-4}$	$\mathbf{9.285 \cdot 10^{-6}}$	**0.940**	0.0115

Table 6.1: Models after HPO evaluated on different metrics. The best score for each metric is highlighted.

6.2.1 Evaluation Using the Metrics

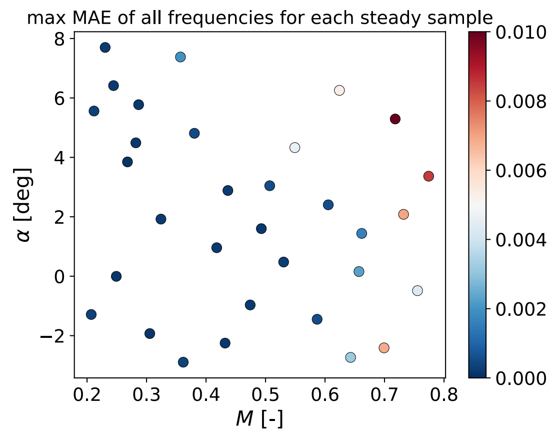
Table 6.1 gives the accuracy of the six models evaluated on the four metrics introduced in section 6.1. Since the evaluation is done using all samples in the test dataset the resulting values are averaged over the number of samples. The GCN Local model and the GNS Local model both perform best on two metrics each. Interpolation scores worst on all metrics. None of the scores deviates from the others by more than an order of magnitude.

Subsequently, only the Interpolation model and the GNS Local model are further investigated to maintain clarity. These two models are compared in the text, analogue figures for the other models can be found in the appendix A.4. In figure 6.1 the two models are compared regarding the samples with the maximum MAE at each steady sample location. A similar trend can be seen for both models: higher errors can be found at higher Mach numbers. This trend is more pronounced for the Interpolation model (see subfig. 6.1a). Similar trends can be seen for all models and across all metrics.

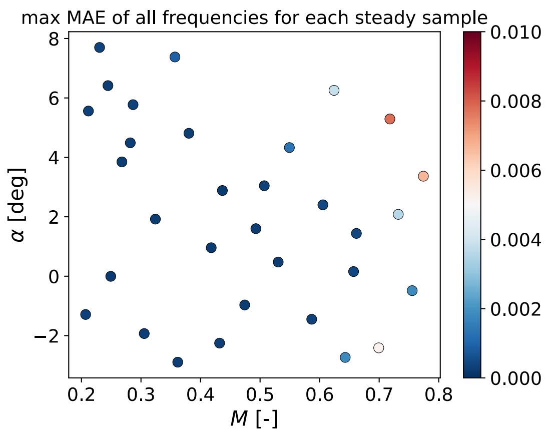
A decaying accuracy at higher Mach numbers might be explained by the fact that the number of samples (in all partitions of the dataset) is smaller for higher Mach numbers as explained in subsection 4.4.3. Thus, all the models could be biased towards lower

[1]The SMARTy callback `EarlyStoppingByImprovement` with patience equal to number of training epochs is used.

Mach numbers. Another explanation might be given by the fact that more non-linear aerodynamic effects occur at higher M. As a result of the hyperparameter optimization (HPO) a linear kernel was selected for the Interpolation model. Consequently, it is not capable of covering non-linearities.



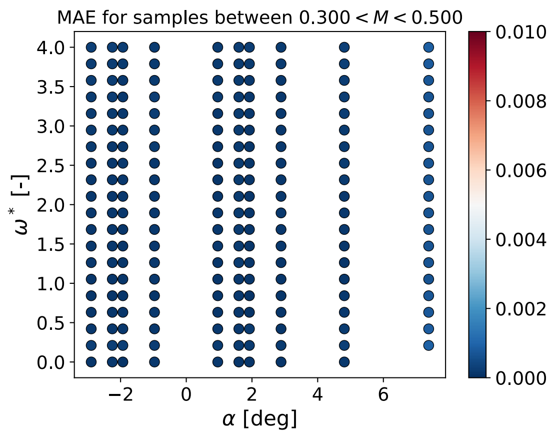
(a) Interpolation



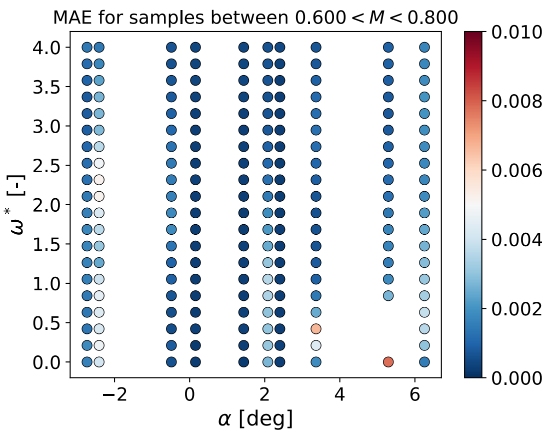
(b) GNS Local

Figure 6.1: Maximum MAE of the Interpolation and the GNS Local model evaluated on the test data shown in the M-α plane.

Figure 6.2 presents the MAE on the test data for the GNS Local model on the α-ω^* plane at specified Mach number ranges. This confirms the trend that the model performs better on samples at lower Mach numbers.



(a) $0.3 \leq M \leq 0.5$

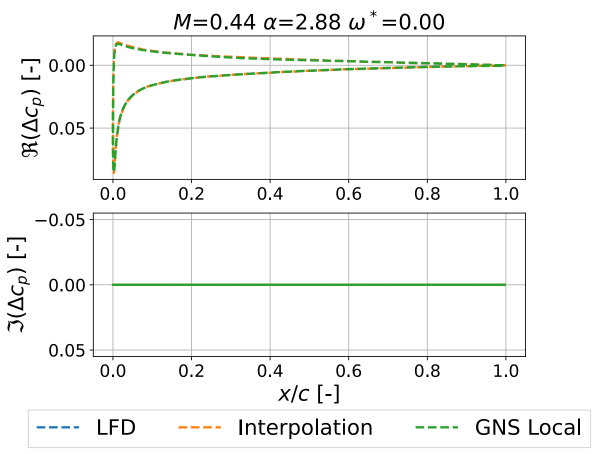


(b) $0.6 \leq M \leq 0.8$

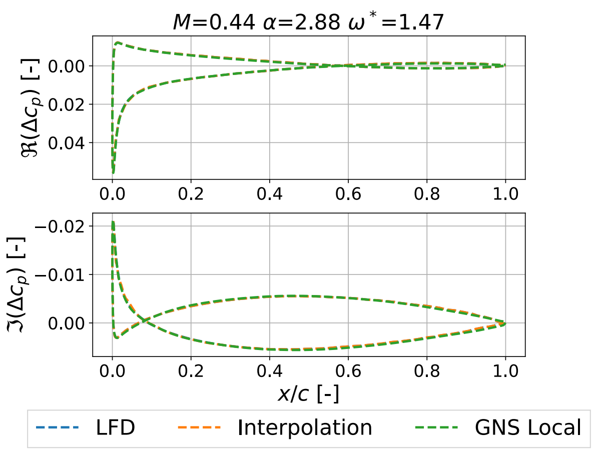
Figure 6.2: MAE for the GNS Local model using all test samples in a specified range of Mach numbers shown in the α-ω^* plane.

6.2.2 Low Mach Number Behaviour

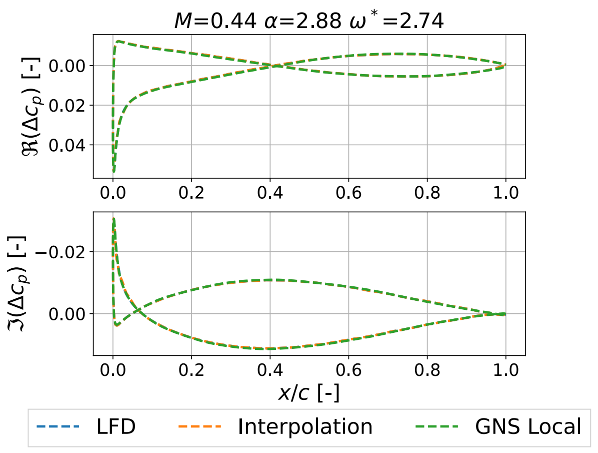
A random sample at a low Mach number ($M = 0.44$) is analyzed. Regarding M and α, the sample is located in the middle of the sample space, as visible in subfigure 6.3e. Therefore, the model can learn from a lot of surrounding data. Linear aerodynamic behaviour can be expected for this sample. Figures 6.3 show the real and imaginary part of the increment to the pressure coefficient for the Interpolation and the GNS Local models for four selected reduced frequencies (the remaining models can be found in the figure A.10). The considered reduced frequencies are selected in a way that the borders of the sample space ($\omega^* = 0.0$, $\omega^* = 4.0$) and two frequencies in the middle of the sample space are investigated. These same frequencies are examined throughout this chapter.



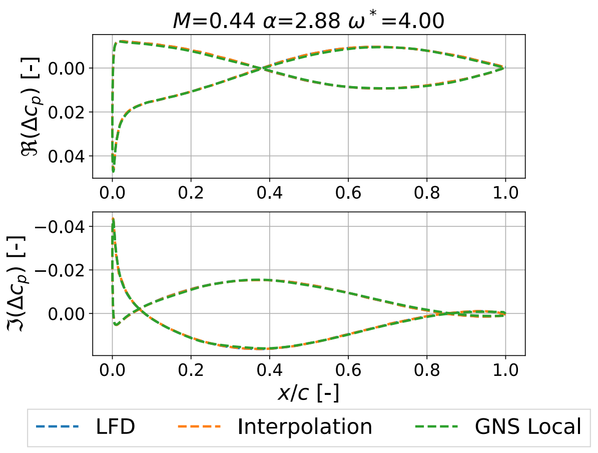
(a) $\omega^* = 0.0$



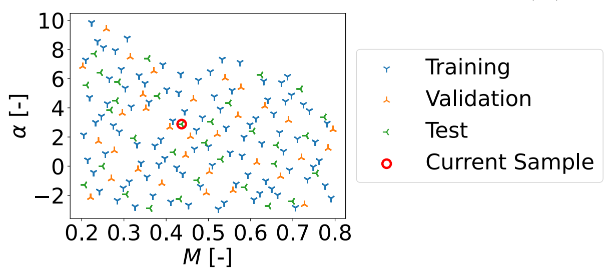
(b) $\omega^* = 1.47$



(c) $\omega^* = 2.74$



(d) $\omega^* = 4.00$



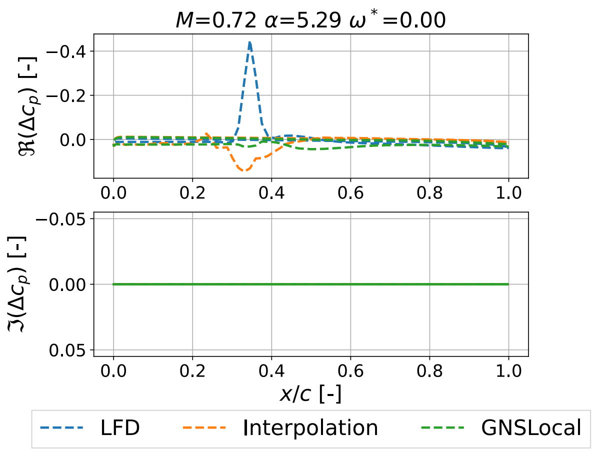
(e) Location of sample in M-α plane

Figure 6.3: Distribution of complex pressure increment of Interpolation and GNS Local model compared to LFD at $M = 0.44$, $\alpha = 2.88$.

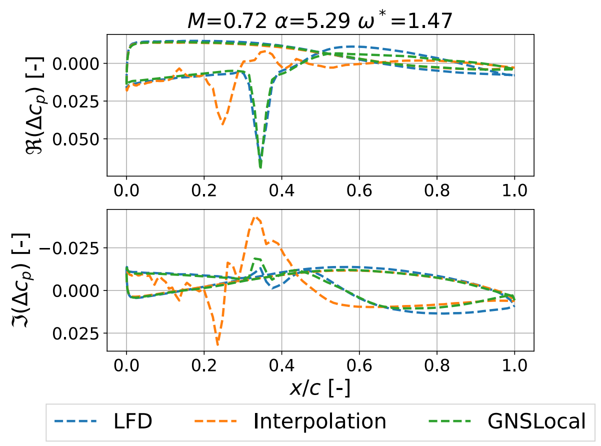
As the figures show, both modelling approaches give near *perfect* predictions at this M-α location for all ω^*. Similar behaviour can be seen for all models. The Interpolation model is the simplest approach. Consequently, if a surrogate model is only needed to predict the aerodynamic behaviour at low Mach numbers and a sufficient amount of training data is available, the Interpolation model might be a suitable choice.

6.2.3 High Mach Number Behaviour

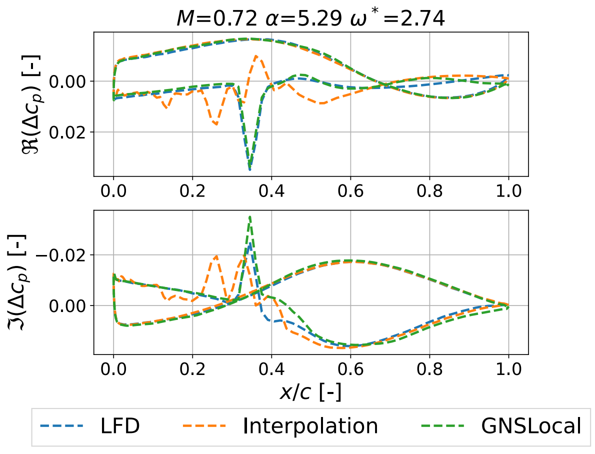
In contrast to the samples at lower Mach numbers, more complex flow behaviour with non-linear characteristics can be expected at higher Mach numbers. As figure 6.1 has already shown lower prediction accuracy is expected in this area. Consequently, in this subsection samples at high Mach numbers are looked at.



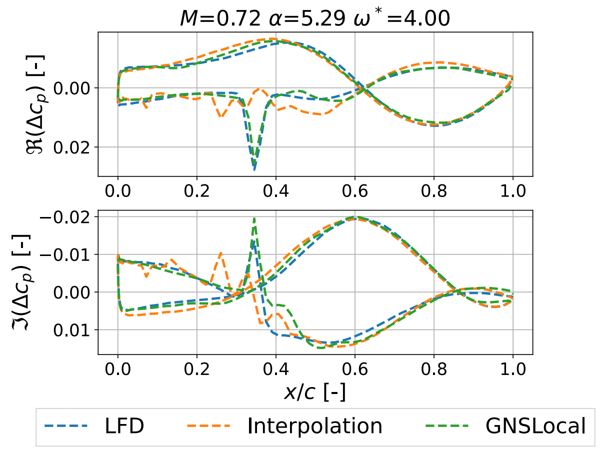
(a) $\omega^* = 0.0$



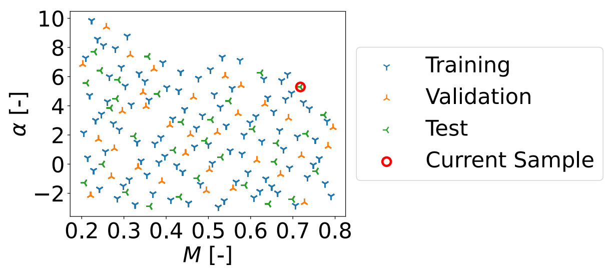
(b) $\omega^* = 1.47$



(c) $\omega^* = 2.74$



(d) $\omega^* = 4.00$

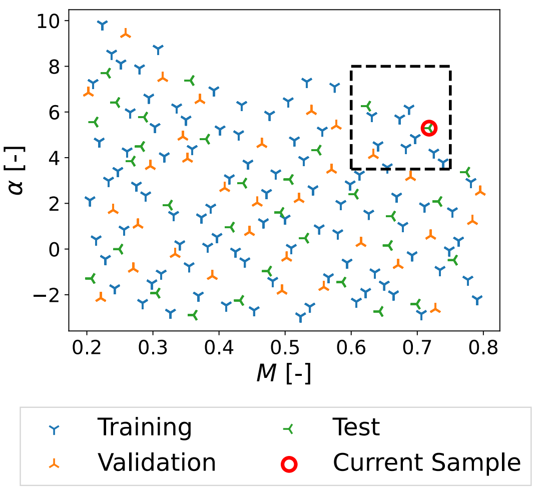


(e) Location of sample in M-α plane

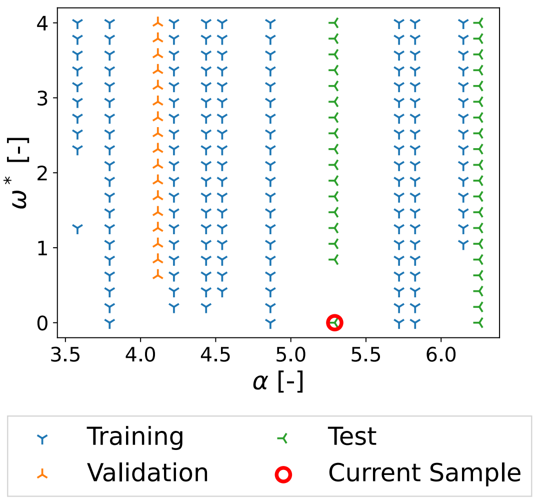
Figure 6.4: Distribution of complex pressure increment of Interpolation and GNS Local model compared to LFD at $M = 0.72$, $\alpha = 5.29$.

Sample at $\mathbf{M = 0.72}$ and $\boldsymbol{\alpha} = \mathbf{5.29}$

The sample in figure 6.4 (and fig. A.11) shows the predictions at $M = 0.72$ and $\alpha = 5.29$. This sample lies at the edge of the M-α sampling space, as visible in subfigure 6.4e. Thus, the data seen during training in this area is limited. As the plots in the figure show the Interpolation model is not able to give good predictions. In particular, the Interpolation model does not capture the peaks. However, from a practical standpoint, these peaks are the most interesting points since they correspond to the shock location on the airfoil and describe how the shock is affected by the gust excitation. The GNS Local model gives good, but improvable predictions at all frequencies except $\omega^* = 0.0$. This reduced frequency defines a boundary of the sampling space. However, this is also the case for $\omega^* = 4.0$ which does not lead to a bad prediction. Since the simulations at high M-α simulations frequently did not converge, the available training and validation data close to the sample is looked at in figure 6.5. The plots in the figure confirm the assumption that data around the sample is sparse. Especially simulations at low reduced frequencies failed more often.



(a) Definition of region that is looked at in figure 6.5b

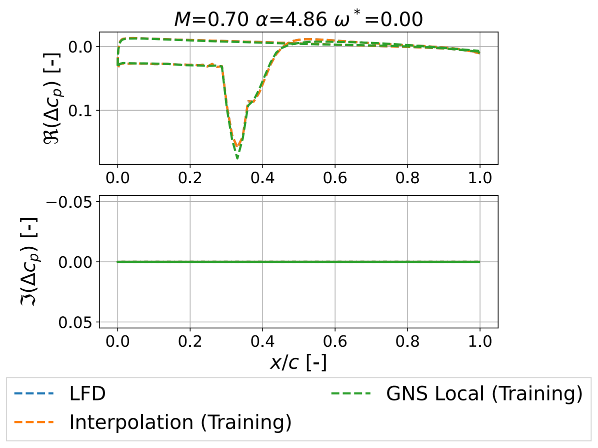


(b) Samples from the region defined in figure 6.5a represented in the α-ω^* plane

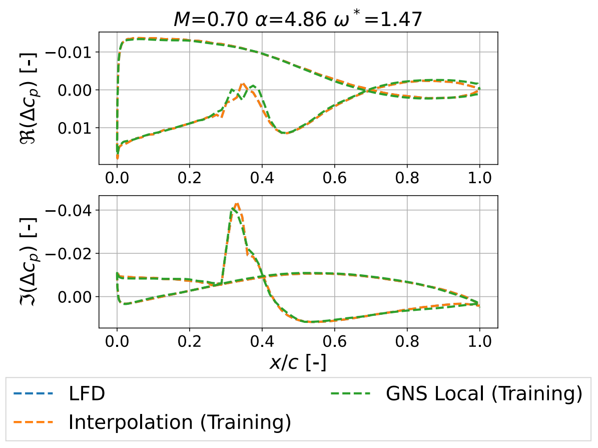
Figure 6.5: Sampling in a defined area around $M = 0.72$ and $\alpha = 5.29$

Training Sample at $\mathbf{M = 0.70}$ and $\boldsymbol{\alpha} = \mathbf{4.86}$

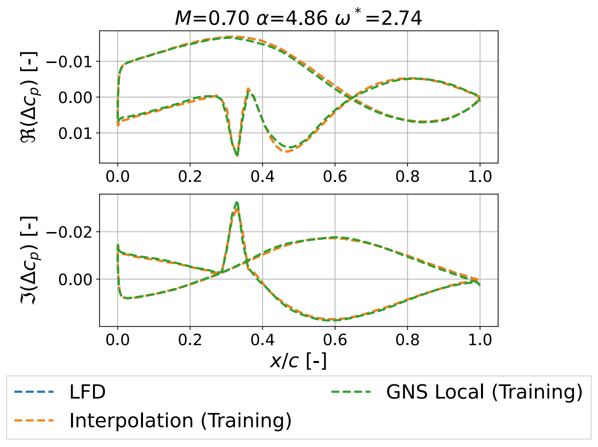
To gain more insights into the behavior in this region a training sample at $M = 0.70$ and $\alpha = 4.86$ is investigated. Figure 6.6 looks at the same frequencies as previously. As the plots indicate the Interpolation model almost perfectly predicts the target values. Considering the inaccurate predictions of the Interpolation model on the previously investigated test sample one can hypothesize that the model is overfitted to the training data and therefore not able to generalize [23]. The predictions of the GNS Local model on the training sample do have small deviations from the LFD solution, but can also be considered reasonably accurate for all reduced frequencies.



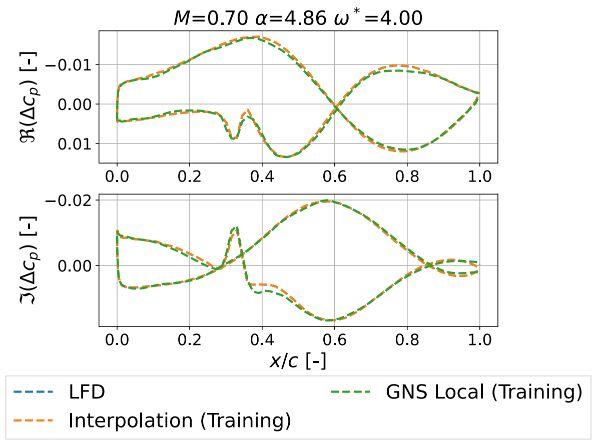
(a) $\omega^* = 0.0$



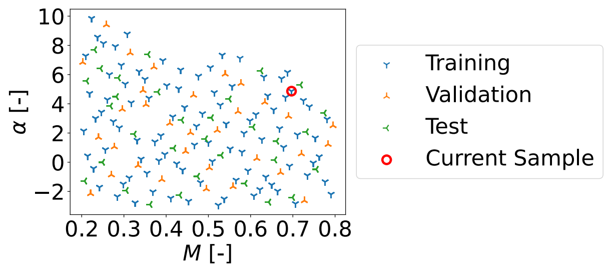
(b) $\omega^* = 1.47$



(c) $\omega^* = 2.74$



(d) $\omega^* = 4.00$

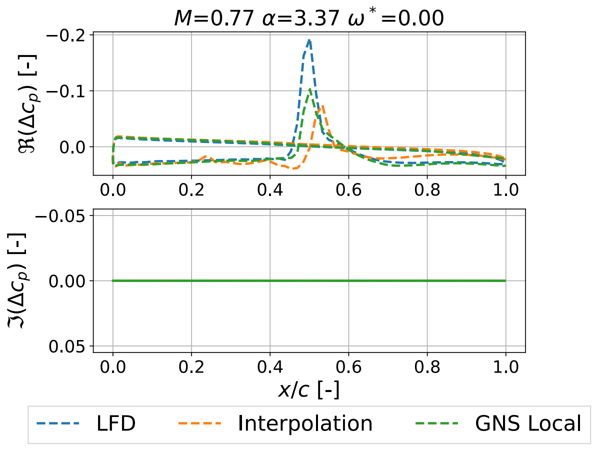


(e) Location of sample in M-α plane

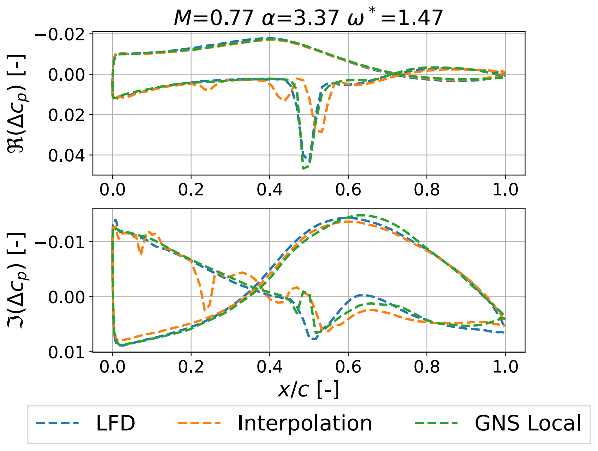
Figure 6.6: Distribution of complex pressure increment of Interpolation and GNS Local model compared to LFD at $M = 0.70$, $\alpha = 4.86$ (Training dataset).

Sample at $\mathbf{M = 0.77}$ and $\boldsymbol{\alpha = 3.37}$

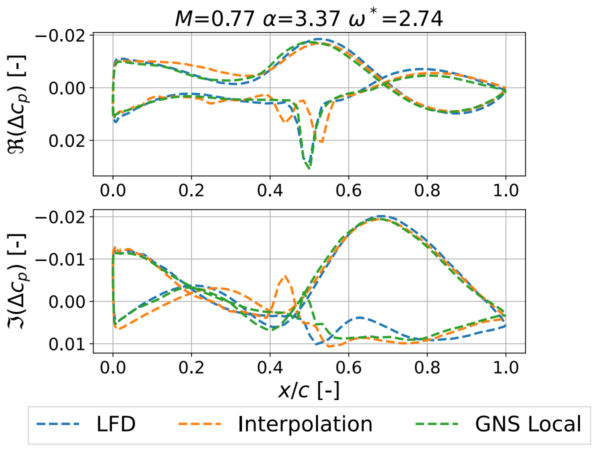
After observing that the GNS Local model can predict training data and higher reduced frequency test data at a high Mach number by examining the previous two samples, it is of interest to investigate whether the model is capable of predicting the peak for a low frequency sample as well. Figure 6.7 (and fig. A.13) refers to a sample at $M = 0.77$ and $\alpha = 3.37$. The surrounding samples are presented in the appendix in figure A.8. In this case, more training and validation data at lower reduced frequencies is available. Since this sample is at a lower Mach number than the previous one, this is consistent with the observation that non-convergence is a more relevant issue for LFD simulations at high M and high α. The GNS model is giving good predictions with some deviations to the LFD solutions at $\omega^* = 1.47$, $\omega^* = 2.74$ and $\omega^* = 4.00$. For $\omega^* = 0.0$ the model predicts the location of the peak, but deviates regarding the magnitude of the peak. This supports the hypothesis that more training data at low frequencies improves the prediction quality at $\omega^* = 0.0$. However, this sample also lies on the border of the M-α sampling space leading to an overall small amount of available training data in this region.



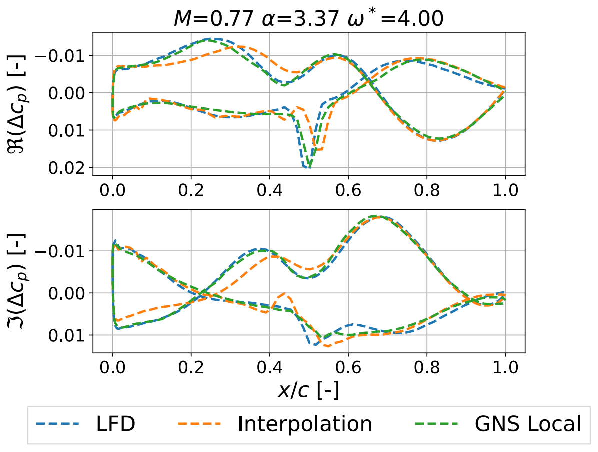
(a) $\omega^* = 0.0$



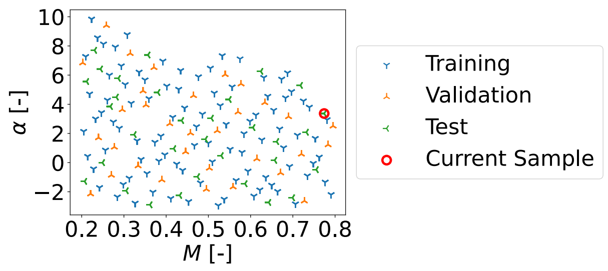
(b) $\omega^* = 1.47$



(c) $\omega^* = 2.74$



(d) $\omega^* = 4.00$



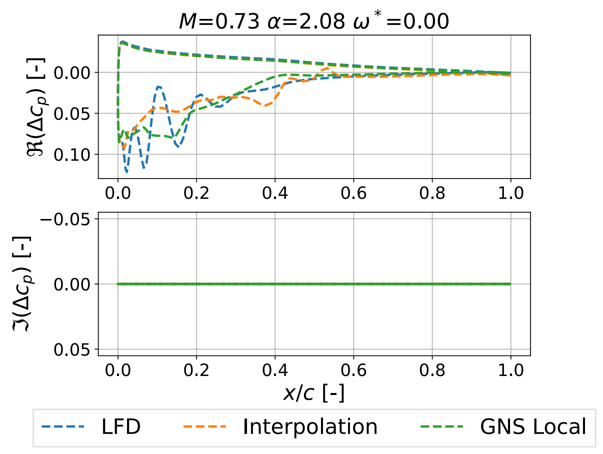
(e) Location of sample in M-α plane

Figure 6.7: Distribution of complex pressure increment of Interpolation and GNS Local model compared to LFD at $M = 0.77$, $\alpha = 3.37$.

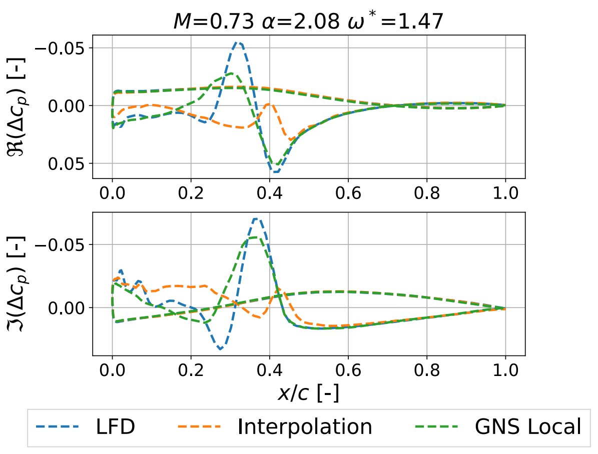
For non-sinusoidal excitations, e.g. with a 1-cos gust, the excitation signal can be transformed into frequency domain using a discrete Fourier transform that results in a series of frequency components. Typically, most of the original signal can be explained by a small number of low frequencies. Thus, especially the prediction of low frequencies should be accurate. The results so far suggest that the use of the models in this test case might be of limited benefit. Nevertheless, with enough training data available potential to also capture complex flow characteristics as the peaks can be seen.

Sample at M = 0.73 and α = 2.08

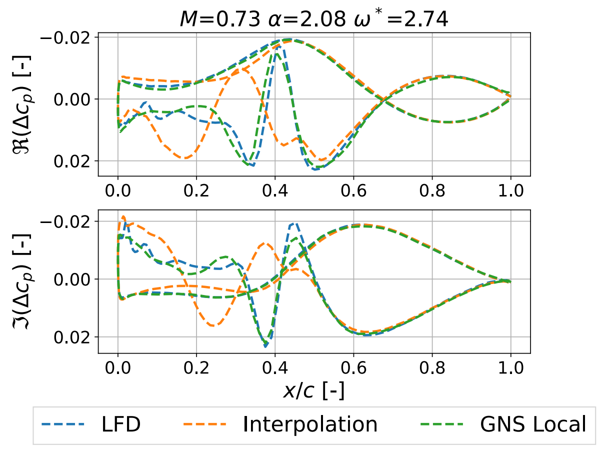
The previous high Mach number samples all have a peak corresponding to a shock in the steady solution. Figure 6.8 (and fig. A.12) shows a different behaviour. The LFD solution exhibits oscillations for all frequencies. The Interpolation model gives poor predictions for all frequencies. This again shows that it is not capable of capturing the underlying aerodynamic effects in this region.



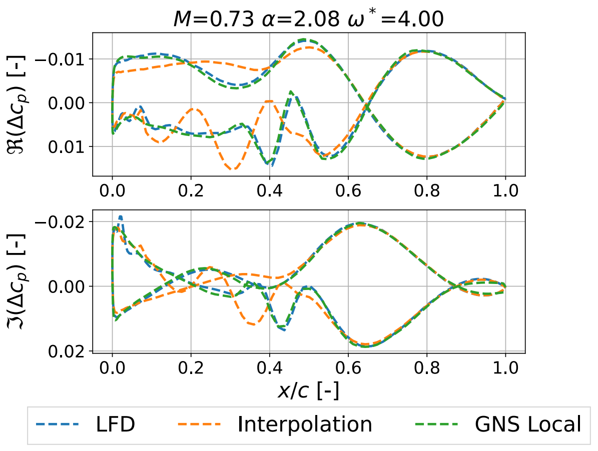
(a) $\omega^* = 0.0$



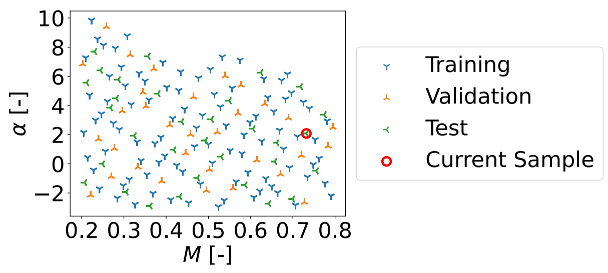
(b) $\omega^* = 1.47$



(c) $\omega^* = 2.74$



(d) $\omega^* = 4.00$



(e) Location of sample in M-α plane

Figure 6.8: Distribution of complex pressure increment of Interpolation and GNS Local model compared to LFD at $M = 0.73$, $\alpha = 2.08$.

The prediction accuracy of the GNS Local model decreases for lower reduced frequencies. In particular, for $\omega^* = 0$, the GNS model is not able to predict the oscillations. Figure A.9 in the appendix shows the surrounding samples. In the considered area only two validation samples at $\omega^* = 0$ are missing. The training samples are complete. However, this amount of data is not enough for the model to capture the underlying effects at low reduced frequencies. Similar results as for the GNS Local approach can be seen for the other DL models in the appendix. None of the other models performs visibly better than the GNS Local approach.

6.3 Size of the Dataset

In a practical setting, especially when investigating three-dimensional test cases, less data might be available due to limited computational resources. Thus, the influence of the dataset size on the results is of great interest. In this section, the influence of the amount of data seen in training is looked at.

Method

All model models are evaluated on the same test dataset as previously. For training and validation purposes differently sized subsets of the dataset are created. The reduction is defined by a percentage s. Then, a reduction in the number of samples is done in the steady sampling plane. Consequently, the resulting datasets look at fewer combinations of M and α, but still at 20 reduced frequencies at each steady sample. The original steady sampling containing N steady training samples is created by a Halton sequence. The reduced training data is created by selecting the first $s \cdot N$ samples from the original training data. The same is done for the validation data. No changes are applied to the test data. Table 6.2 shows the resulting subsets. In figure A.14 the datasets are visualized. It must be noted that especially the 5% reduction results in a very sparse training dataset and does only contain one validation sample (see subfig. A.14a).

Share of original dataset	**Training** samples	**Validation** samples	**Test** samples	**Sum** of samples
100 %	1875	634	635	3144
80 %	1482	498	635	2615
60 %	1103	378	635	2116
40 %	738	239	635	1612
20 %	359	119	635	1113
10 %	164	60	635	859
5 %	80	20	635	735

Table 6.2: Number of samples used in the training, validation, and test datasets that are used to investigate the size of the dataset

Each model is trained 10 times on each of the resulting datasets with random initialization. The hyperparameters are chosen according to the results of HPO. The models are trained for 2000 epochs each. Afterwards, the model adaptation with the lowest validation error during the training is used.

Results
The GNS Local model did not converge for all trials when trained over 2000 epochs (see fig. A.16). This might be a case of underfitting: the model is not able to capture the underlying pattern. Consequently, the training was repeated with 4000 epochs, those results are looked at.

Contrary to the observation that the prediction variance is decreased if the models are trained on larger datasets, the GNS Local model shows increased variance when trained on 80% or 100% of the dataset (see figs. A.16 and A.17). In comparison to the error of the 60% case, the 80% and 100% cases not only show an increase in variance but also in the mean value of the MAE. Investigating the validation error during training over the number of training epochs one can see that not all GNS Local models did converge during 2000 training epochs. Repeating the training for 4000 epochs leads to a decreased variance and improvement in accuracy. Still, some variance can be seen (see fig. A.15), nevertheless the model is used for further evaluation since this satisfies the purposes of this investigation.

Figure 6.9 shows the averaged MAE of 10 models over the size of the dataset. For small datasets, the error grows considerably. The largest decay in accuracy can be found for the GCN models. This might be caused by the large number of weights in this model (see table 6.3) that are not properly updated during training. As figure A.15 shows, the accuracy of the DL models has an increased variance for smaller datasets. The Interpolation model is deterministic and therefore has no variance. It must be noted that the HPO was done using the full dataset. It is reasonable that the accuracy of the models improves if they were optimized towards the smaller datasets. However, since HPO is very costly it is not done in this work. Depending on the needed accuracy and the available resources the results of this section can give a hint of which models are more suitable.

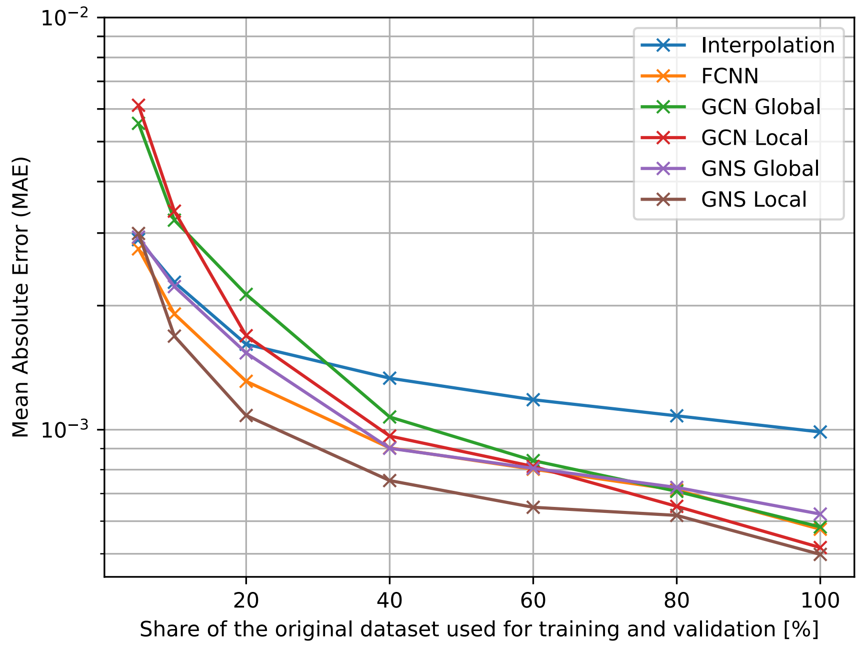


Figure 6.9: MAE over the size of training data for averaged over 10 trials for each model

6.4 Unseen Reduced Frequencies

The previous studies used the same set of reduced frequencies during training and testing. The goal of this section is to evaluate how the models perform on unseen frequencies, as the goal is to build a surrogate for the LFD solver which can solve for any frequency.

6.4.1 Creation of Reduced Dataset

Based on the dataset that is described in chapter 4 new training, validation and testing data are created as subsets. New training and validation data is created by selecting the same steady samples as in previous studies, but only using every second reduced frequency starting at the first one. Three test datasets are created: testing 1 consists of seen global flow parameters and unseen reduced frequencies. Dataset testing 2 consists of unseen global flow parameters and seen reduced frequencies. Testing 3 consists of unseen global flow parameters and unseen reduced frequencies. Figure 6.10 gives a schematic representation of the datasets.

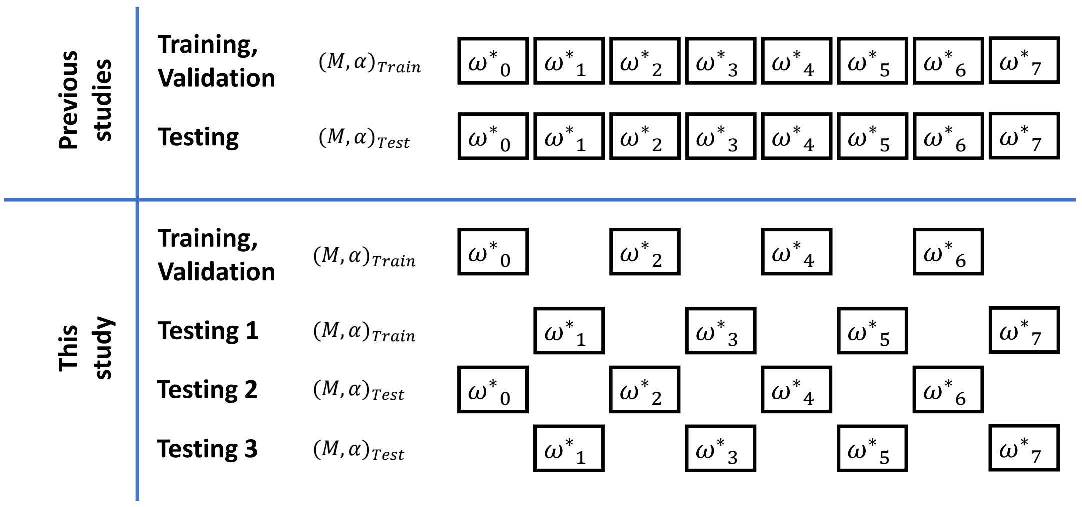

Previous studies
Training, Validation
$(M,\alpha)_{Train}$
ω^*_0 ω^*_1 ω^*_2 ω^*_3 ω^*_4 ω^*_5 ω^*_6 ω^*_7
Testing
$(M,\alpha)_{Test}$
ω^*_0 ω^*_1 ω^*_2 ω^*_3 ω^*_4 ω^*_5 ω^*_6 ω^*_7
This study
Training, Validation
$(M,\alpha)_{Train}$
ω^*_0 ω^*_2 ω^*_4 ω^*_6
Testing 1
$(M,\alpha)_{Train}$
ω^*_1 ω^*_3 ω^*_5 ω^*_7
Testing 2
$(M,\alpha)_{Test}$
ω^*_0 ω^*_2 ω^*_4 ω^*_6
Testing 3
$(M,\alpha)_{Test}$
ω^*_1 ω^*_3 ω^*_5 ω^*_7

Figure 6.10: Schematic representation of the datasets used in the unseen frequencies investigations

6.4.2 Results

The models are trained on the new reduced training dataset with the hyperparameters as determined during the HPO. Subsequently, the models are evaluated on the different test datasets. Examining the resulting prediction MAE in figure 6.11 (and the MSE and relative Error in the appendix A.18 and A.19) the same trend can be seen for all six models: evaluating the models at a seen M and α yields the smallest prediction error. Evaluating the models at unseen Mach and angle of attack leads to slightly higher errors at unseen reduced frequencies than at seen reduced frequencies. The difference in MAE between the latter two cases is smaller compared to the difference in MAE between the former two cases. Thus, it is more important that the model has seen a steady sample during training than a reduced frequency. This knowledge in combination with the finding from section 6.2 that lower frequency samples in the training data seem to be important can be used to define a new sampling strategy focusing on a high number of steady samples in combination with a refinement regarding lower reduced frequencies.

Figure 6.12 shows the prediction MAE of the Interpolation and GNS Local model on the three reduced test datasets. It compares the performance of the models trained on the full dataset to the models trained on the reduced datasets.

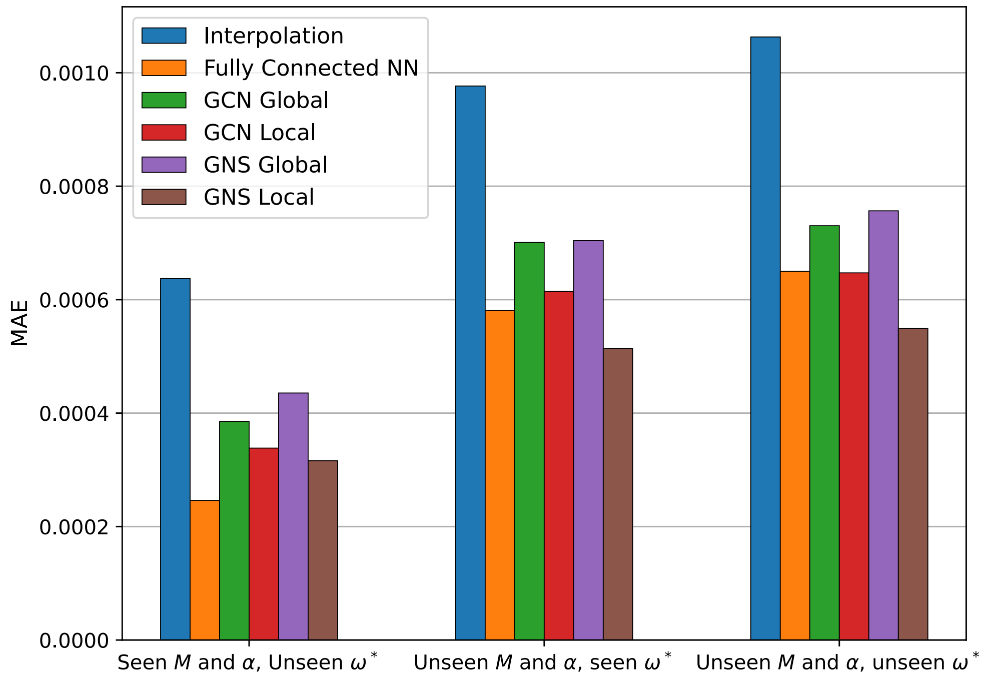
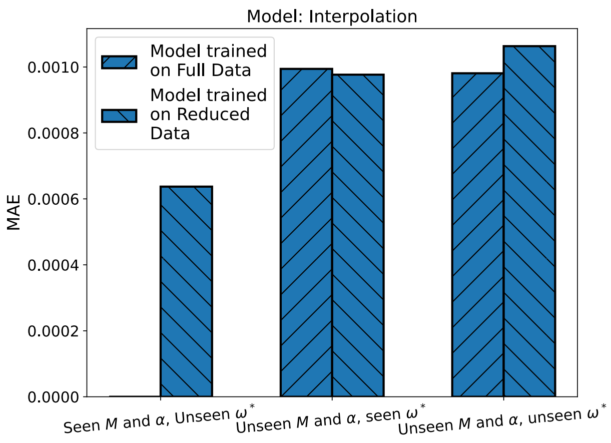
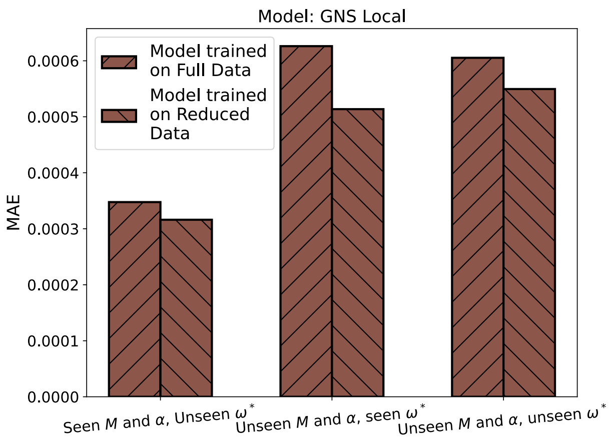


Figure 6.11: MAE for prediction of models in the unseen frequency investigation

The Interpolation model (see subfig. 6.12a) trained on the full dataset gives almost perfect predictions on the first test case, since these points are support points of the model. On unseen M, α and seen ω^* the model trained on reduced data performs slightly better, perhaps the additional sampling points in the model trained on the full dataset introduce noise into the model. On unseen M, α and ω^* the model trained on the full dataset performs better as it has seen these frequencies during training.



(a) Interpolation



(b) GNS Local

Figure 6.12: Comparison of Interpolation and GNS Local model trained on the entire dataset and trained on the reduced dataset. Note that labels on the abscissa are only valid for the models trained on the reduced dataset since the model trained on the full dataset has seen all ω^* during training.

The GNS Local model trained on the full data performs worse than the counterpart trained on the reduced data on all three test datasets. This could indicate that too many data points in training introduce noise into the model which leads to a deterioration in prediction accuracy. This can be supported by the results of the data size investigation, where the variance in the results of the GNS Local model increased for large datasets.

6.5 High Mach Number Models

The Interpolation model has proven suitable at low Mach numbers only. Thus, the question arises as to whether the models can be improved when trained in a specific region of the sample space. Especially interesting are regions with non-linear aerodynamic effects. These effects can be found at high Mach numbers.

6.5.1 Creation of Reduced Dataset

A subset from the original dataset is created in a way that only samples with $M \geq 0.6$ are regarded since this range produces higher errors when the models are trained on the entire dataset (see fig. A.20). Using the hyperparameter settings as previously each data-driven approach is trained on the data of the reduced sampling. Subsequently, the models are evaluated on the test samples from the reduced sampling. Furthermore, the models that have been trained on the entire dataset are evaluated on the same test samples.

6.5.2 Results

Figure 6.13 shows the prediction errors on the reduced test set. The results show an ambiguous tendency: the FCNN and GNS Local models give better predictions when trained on $M \geq 0.6$ data only, while the other models perform better when trained on the entire dataset. Similar tendencies can be seen using the other metrics (see figs. A.21 and A.22).

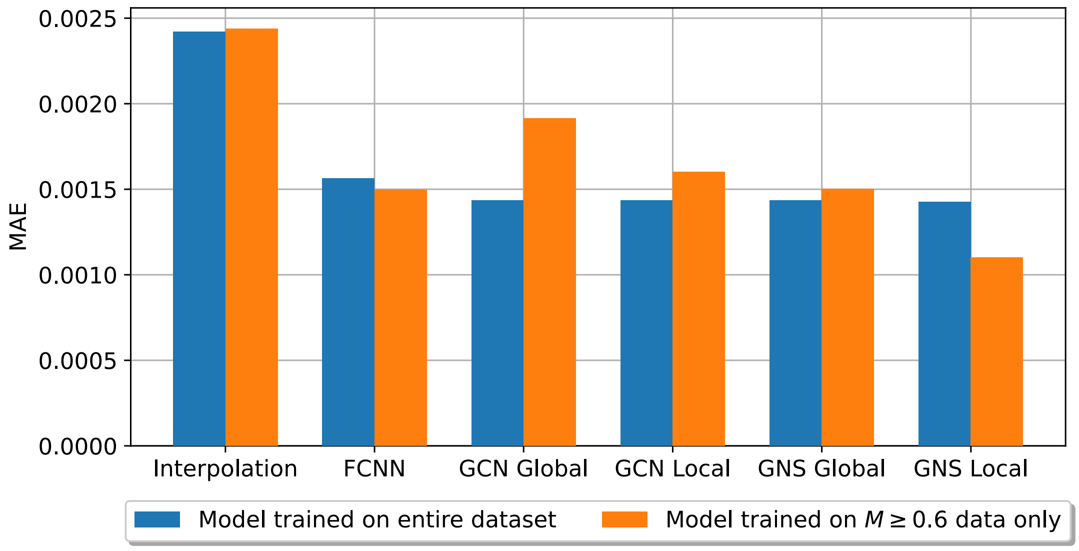


Figure 6.13: Models trained on full dataset and models trained on samples $M \geq 0.6$ evaluated against each other using the MAE.

The GNS Local model shows the strongest improvement when trained on the $M \geq 0.6$

data. The prediction accuracy of the GCN Global model has the most deterioration in this case. During the training on the entire dataset, the models have seen more samples than during training on $M \geq 0.6$ data, this must be noted when examining the results. The improvements and deterioration of the different models are visible, but not significant. Consequently, one can conclude that training the models just on high Mach number samples did not lead to significant improvements in the performance of the models and capturing non-linear behaviour.

6.6 Computational Effort

After analyzing the models' abilities to give accurate predictions, the computational effort must be evaluated since the motivation for a data-driven surrogate modelling is to enable faster predictions. The computational effort is evaluated regarding the training time, the size of the models, and the evaluation time.

	Training Time		**Number of Parameters**		**Evaluation Time**	
	t [s]	Relative	Absolute	Relative	t [s]	Relative
LFD					489.35 ± 4.890	1
Interpolation	4.33 ± 0.195	1	375 200	1	0.135 ± 0.0034	$2.76 \cdot 10^{-4}$
FCNN	13216.4 ± 2077.9	3054.4	1 844 226	4.92	0.124 ± 0.3160	$2.53 \cdot 10^{-4}$
GCN Global	6947.4 ± 483.7	1605.6	8 421 384	22.45	0.057 ± 0.0400	$1.16 \cdot 10^{-4}$
GCN Local	7979.6 ± 755.1	1844.1	10 524 680	28.05	0.062 ± 0.0003	$1.27 \cdot 10^{-4}$
GNS Global	5749.3 ± 23.6	1328.7	629 298	1.68	0.454 ± 0.0233	$9.28 \cdot 10^{-4}$
GNS Local	3712.3 ± 79.3	857.9	2 176 050	5.80	0.033 ± 0.0018	$6.74 \cdot 10^{-5}$

Table 6.3: Comparison of computational effort. The relative training time is normalized using the Interpolation training time. The relative numbers of parameters are normalized using the number of parameters of the Interpolation model. The evaluation time is normalized using the LFD evaluation time.

6.6.1 Training Time

In contrast to the LFD solver, the data-driven approaches must be trained before usage. All time comparisons are done using DLRs high-performance computing cluster *CARA*. Graphic processing units (GPUs) are an established choice for the handling of data-driven models. CARA is equipped with NVIDIA A100 GPUs. It must be noted, that results might be affected by the current load on the computing cluster.

To account for randomness each model is trained 10 times for 2000 epochs. GNS Local is trained for 4000 epochs. Note that some models might not need all epochs of training and converge faster. Table 6.3 shows the mean absolute training time and the standard deviation. By dividing the mean training times with the Interpolation model's mean training time the relative training times can be computed.

The Interpolation model is trained roughly three orders of magnitude faster than the DL approaches. The FCNN model takes the most training time and has the highest standard deviation as well. GNS Local is the fastest DL approach and even faster than the GNS Global approach despite though GNS Local uses more weights than GNS Global. Different implementations for the two models are used and can provide an explanation for this.

6.6.2 Number of Parameters

The considered data-driven models have several parameters (weights) that are adjusted during training and must be stored. Therefore, their number can be seen as a representation of the required storage for a model. The absolute numbers and the numbers normalized by the number of parameters of the Interpolation model can be seen in table 6.3.

The Interpolation model uses the least number of weights, but it must be noted that the weights are complex numbers and therefore need more storage than real numbers. Both of the GCN models use by far the largest amount of parameters. Another trend that can be seen is that the GNS Local and GCN Local models both have more parameters than their counterparts that use only global flow features.

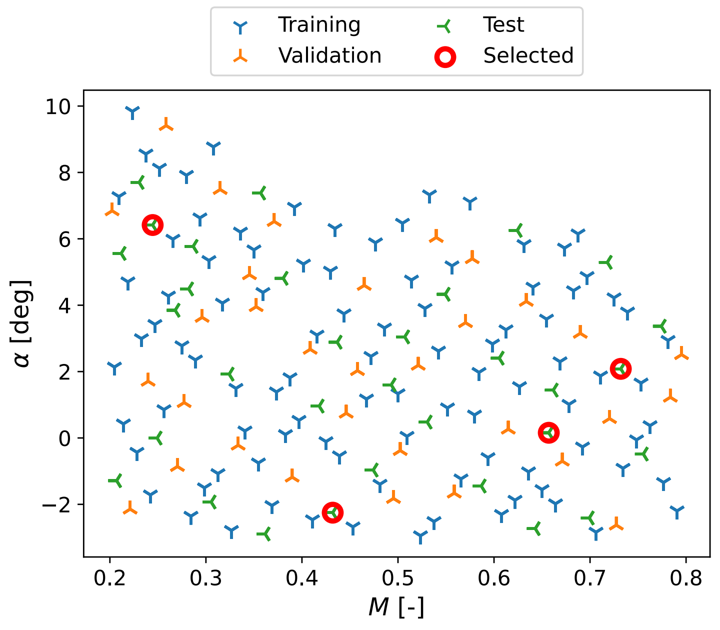


Figure 6.14: Samples selected for comparison of computational effort

6.6.3 Evaluation Time

Regarding the evaluation time, a comparison of the data-driven approaches to LFD can be done. Comparing both on the same device would be desirable. However, LFD computations are usually performed using CPUs, while GPUs are used for data-driven models. Thus, the comparison is done using the CARAs CPU capability (AMD EPYC 7061) for LFD and GPU capability for the remaining models.

Four samples are randomly drawn from the M-α plane. The four samples in figure 6.14 lead to converged simulations at all 20 reduced frequencies. Table 6.3 lists the mean time and the standard deviation that each approach needs to compute the results for 20 frequencies at the four given M-α combinations. The mean and the standard deviation are calculated by doing each computation 10 times. The relative values are computed by dividing the mean times by the mean time of the LFD solver.

All six data-driven models lead to time improvements by four to five orders of magnitude. The results might be dependent on the current load on the used devices. In this case, the GNS Local approach gives the fastest predictions. The fast results of the GNS Local models can again be explained by the different implementations when compared to the GNS Global model. Furthermore, the high standard deviation of the FCNN model is noticeable.

The most time costly step in the LFD computation is the computation of the Jacobian matrix $\partial\mathbf{R}/\partial\mathbf{w}$ (see eq. 2.3). To save computational time this should only be done once per steady case. Due to the implementation of the LFD solver, it was computed for each reduced frequency. Consequently, the LFD simulation can be sped up. However, one can still expect the data-driven approaches to be orders of magnitude faster since building the Jacobian matrix only once per steady case would only lead to a maximal reduction to a 20-th of the stated times.

Chapter 7

Conclusions and Outlook

7.1 Conclusions

As stated in the introduction the research goal is to investigate the applicability of different types of graph neural networks (GNNs) for the prediction of aerodynamic quantities in the frequency domain by comparing it to other data-driven methods and the LFD solver. The specific problem in this paper is to predict the complex-valued increment to the pressure coefficient at an airfoil surface given a Mach number, an angle of attack and a gust excitation which is characterized by a reduced frequency.

A test case using the two-dimensional NACA 64A010 airfoil is designed in a way to resemble a realistic use case of the proposed surrogate model by creating a DoE consisting of a quasi-random sampling of the Mach number and angle of attack combined with multiple reduced frequencies at each steady sample. During this paper, a limitation towards the surface solution and a single airfoil is done. Furthermore, in the course of feature selection pressure is selected as the only significant local flow feature. As the results show the sampling strategy is a good first reference point. However, due to unconverged simulations at high M and α, and low ω^*, the resulting dataset has sparse regions which probably influence the prediction capabilities.

Based on four different data-driven methodologies six surrogate models that can substitute the LFD solver are developed in this paper: Interpolation, Fully-Connected Neural Network (FCNN), Graph Convolutional Network (GCN) Global, GCN Local, Graph Network Simulator (GNS) Global, and GNS Local. The models are adapted to predict complex-valued targets by considering a different number of features, e.g. differentiating between global and local flow features.

The models' hyperparameters are optimized throughout this work towards a minimal prediction error on the stated test case. Notable results from the hyperparameter optimization (HPO) are that hyperparameters influencing the model training are more important than hyperparameters defining the model topology in this case. Furthermore, HPO leads to differently-sized models. The GCN models use roughly 5 to 28 times more parameters than the other models. Also, models using local flow parameters consist of more hyperparameters than their counterparts with global flow features only.

To evaluate the results, the prediction accuracy of the different models is evaluated on four metrics with a focus on the mean absolute error (MAE). The GNNs using local flow features score best on two metrics each. Analyzing the prediction errors regarding their distribution depending on the global flow features shows a clear trend: all models give almost perfect predictions for samples at lower Mach numbers. Depending on the model higher errors with differing magnitudes are seen at higher Mach numbers since non-linear flow characteristics can be found in this region. Investigating specific samples gives further insights: the selected linear interpolation approach is not suited for the non-linear flow characteristics. The deep learning models show more potential in this region as they give good predictions for several Δc_p-distributions, predominantly at higher reduced frequencies. However, especially at lower reduced frequencies, they do not lead to satisfactory results, even though for some samples the shock location is predicted correctly.

Determining the influence of the size of the training data on the prediction results shows a strongly deteriorating accuracy for all models especially the GCN approach which might be due to the high number of parameters that these models need to adapt during training. The GNS Local approach leads to constantly best MAE-scores, but needs a higher number of training epochs.

Investigating the prediction capability on unseen frequencies again leads to the best results for the GNS Local model. Furthermore, this investigation shows that with the given test case all models give better predictions when they interpolate on unseen frequencies than when they have to interpolate regarding unseen steady samples.

Another investigation shows that there is no general benefit regarding the prediction results for high Mach number test cases when a model is trained on high Mach number samples only.

Regarding the computational effort, all data-driven approaches lead to evaluation time savings of approximately four orders of magnitude. The differences between evaluation times of the data-driven approaches are negligible. The DL approaches need about three orders of magnitude more training time than the Interpolation model. The FCNN approach needs the longest training time whereas the GNS Local approach is the fastest.

Based on the described results some takeaway points can be formulated:

- If a surrogate model is only needed for **low Mach numbers** the **Interpolation** model is a suitable choice, since it is the simplest approach with the shortest training time. Furthermore, it is capable of directly predicting complex values and does this equally fast as the other approaches.
- For surrogate model problems that also to incorporate **high Mach number** samples all deep learning approaches show varying degrees of potential. The **GNS Local** model proves to be the most promising approach in this work based on scores on the metrics, the comparably low deterioration of accuracy when trained on small datasets, and the capability to capture non-linear behaviour.
- Using **local flow features** leads to slightly lower errors than using global flow features only.
- **All data-driven** approaches enable **significant time savings** regarding the evaluation.

Regarding the research objective, progress has been made by applying the GNN methods to a first frequency domain case. GNNs, especially the GNS model are a promising approach for predicting aerodynamic quantities in the frequency domain. Potential limitations of the models have been identified. Nevertheless, the results are not satisfactory yet, hence further investigation is needed. In the following section ideas for further work are given.

7.2 Outlook

Based on the results of this work, various ways to move forward can be defined. First, choices for other target quantities are explained. Afterwards, possibilities to improve the prediction accuracy and ideas for new test cases are stated. Finally, further methods that could be used to tackle frequency-domain aerodynamic problems are introduced.

Usage of Other Target Quantities
So far only complex-valued Δc_p-distributions at the airfoil's surface are predicted. This value gives a first insight into the capabilities of the different approaches. To build a surrogate for the LFD solver it is necessary to examine how far the current results can be transferred to the entire vector of conservative variables. Furthermore, the computation of global forces, e.g. the coefficient of lift or the coefficient of drag, could be implemented as post-processing. The resulting values could be evaluated as well. Even though the values at the airfoil surface define the forces acting on the airfoil, it might also be interesting to investigate whether using the volume solution as a target quantity delivers additional information to the surrogate models and therefore can improve the outcome. Currently, only predictions based on harmonic excitations using one reduced frequency are evaluated. Real-world use cases apply a spectrum of frequencies, e.g. to obtain the results of the excitation due to 1-cos gusts. Consequently, applying a frequency spectrum to the surrogate prediction results and evaluating the outcome can be a benefit.

Improvement of Prediction Accuracy
There are a variety of options to improve the prediction accuracy of the models. The gradient-loss method proposed in [52] outperforms other approaches by stronger penalising the loss in flow regions with larger gradients. Regarding the application of a spectrum of frequencies to the solution as described in the paragraph above, more sophisticated, tailored loss functions might lead to an improvement. One option could be to define a loss function that gives greater importance to lower reduced frequencies. For all models a selection regarding the hyperparameters that were used for HPO is made, more choices can lead to improvements. Regarding the DL models especially different types of activation functions, e.g. sigmoid or tanh, could be used. Next to altering the models, changes in the DoE might also be beneficial: as described in the results more M-α samples in the high Mach region and more samples at lower reduced frequencies are promising approaches.

New Test Cases
Examining one specific test case as done in this paper gives a first insight into the capabilities of the investigated methods. Considering other airfoils than the NACA 64A010, as well as the use of other meshes, e.g. fine versus coarse or fully structured versus unstructured mesh, gives deeper insights into the strengths and weaknesses of the models. The next step towards industrial usage is the application of 3D test cases, such as an isolated wing up to a full aircraft. Test cases using different types of excitation can deliver insights into the capability to generalize to other problems.

Further Methods
This paper took a selection of methods into account. GCN and GNS are two specific types of GNNs. Investigating modifications such as other message-passing structures might improve the models. Finally, complex-valued neural networks are a current subject

of research, with the first Python implementations as developed in [45] and [46]. Further development and testing of such networks and their conjunction with GNNs could be of great interest for frequency domain problems.

Bibliography

[1] European Union Aviation Safety Agency (EASA), "Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes (CS–25), Amendment 27," 2021.

[2] E. Albano and W. P. Rodden, "A Doublet Lattice Method for Calculating Lift Distribution on Oscillating Surfaces in Subsonic Flow," *AIAA Journal*, vol. 7, no. 2, pp. 279–285, 1969. DOI: `10.2514/3.5086`.

[3] L. Reimer, M. Ritter, R. Heinrich and W. Krüger, "CFD-based Gust Load Analysis for a Free-flying Flexible Passenger Aircraft in Comparison to a DLM-based Approach," ser. 22nd AIAA Computational Fluid Dynamics Conference, AIAA Paper 2015–2455, 2015. DOI: `10.2514/6.2015-2455`.

[4] R. Thormann and M. Widhalm, "Linear-Frequency-Domain Predictions of Dynamic-Response Data for Viscous Transonic Flows," *AIAA Journal*, vol. 51, no. 11, pp. 2540–2557, 2013. DOI: `10.2514/1.J051896`.

[5] P. Bekemeyer, R. Thormann and S. Timme, "Frequency-Domain Gust Response Simulation Using Computational Fluid Dynamics," *AIAA Journal*, vol. 55, no. 7, pp. 2174–2185, 2017. DOI: `10.2514/1.J055373`.

[6] S. L. Brunton, B. R. Noack and P. Koumoutsakos, "Machine Learning for Fluid Mechanics," *Annual Review of Fluid Mechanics*, vol. 52, no. 1, pp. 477–508, 2020. DOI: `10.1146/annurev-fluid-010719-060214`.

[7] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. DOI: `10.1109/TNN.2008.2005605`.

[8] L. Wu, P. Cui, J. Pei and L. Zhao, *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer Singapore, 2022, p. 725.

[9] F. Vetrano, F. Mastroddi and R. Ohayon, "POD approach for unsteady aerodynamic model updating," *CEAS Aeronautical Journal*, vol. 6, pp. 121–136, Mar. 2014. DOI: `10.1007/s13272-014-0133-0`.

[10] K. Hall, E. Dowell and J. Thomas, "Proper Orthogonal Decomposition Technique for Transonic Unsteady Aerodynamic Flows," *Aiaa Journal - AIAA J*, vol. 38, pp. 1853–1862, Oct. 2000. DOI: `10.2514/2.867`.

[11] M. Widhalm and P. Bekemeyer, "Nonlinear low-dimensional model order reduction with subspace interpolation for gust applications with the linear frequency domain approach," Jun. 2023. DOI: `10.2514/6.2023-3947`.

[12] H. Güner, D. Thomas, G. Dimitriadis and V. Terrapon, "Unsteady aerodynamic modeling methodology based on dynamic mode interpolation for transonic flutter

calculations," *Journal of Fluids and Structures*, vol. 84, pp. 218–232, 2019, ISSN: 0889-9746. DOI: `https://doi.org/10.1016/j.jfluidstructs.2018.11.002`.

[13] W. Faller, S. Schrek, A. I. of Aeronautics and Astronautics, *Unsteady Fluid Mechanics Applications of Neural Networks* (AIAA-95/0529). AIAA, 1995. [Online]. Available: `https://books.google.de/books?id=b3f1xwEACAAJ`.

[14] F. Marques and J. Anderson, "Identification and prediction of unsteady transonic aerodynamic loads by multi-layer functionals," *Journal of Fluids and Structures*, vol. 15, no. 1, pp. 83–106, 2001, ISSN: 0889-9746. DOI: `https://doi.org/10.1006/jfls.2000.0321`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0889974600903219`.

[15] A. Mannarino and P. Mantegazza, "Nonlinear aeroelastic reduced order modeling by recurrent neural networks," *Journal of Fluids and Structures*, vol. 48, pp. 103–121, 2014, ISSN: 0889-9746. DOI: `https://doi.org/10.1016/j.jfluidstructs.2014.02.016`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0889974614000383`.

[16] M. Winter, "Nonlinear aerodynamic reduced-order modeling using neuro-fuzzy approaches," Ph.D. dissertation, Apr. 2021. DOI: `10.13140/RG.2.2.29909.09440`.

[17] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," *arXiv e-prints*, arXiv:1704.01212, arXiv:1704.01212, Apr. 2017. DOI: `10.48550/arXiv.1704.01212`. arXiv: `1704.01212 [cs.LG]`.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[19] X. Bresson and T. Laurent, "Residual Gated Graph ConvNets," *arXiv e-prints*, arXiv:1711.07553, arXiv:1711.07553, Nov. 2017. DOI: `10.48550/arXiv.1711.07553`. arXiv: `1711.07553 [cs.LG]`.

[20] T. Wu *et al.*, "Learning large-scale subsurface simulations with a hybrid graph network simulator," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4184–4194.

[21] D. Hines Chaves and P. Bekemeyer, "Graph neural networks for the prediction of aircraft surface pressure distributions," *Aerospace Science and Technology*, vol. 137, p. 108 268, Mar. 2023. DOI: `10.1016/j.ast.2023.108268`.

[22] Deutsches Zentrum für Luft- und Raumfahrt e.V - Institute of Aerodynamics and Flow Technology, "TAU-Code User Guide - Release 2021.1.0," 2021.

[23] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning.* MIT Press, 2016, `http://www.deeplearningbook.org`.

[24] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958, ISSN: 0033-295X. DOI: `10.1037/h0042519`.

[25] K. Hornik, M. Stinchcombe and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989, ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8.

[26] R. Diestel, *Graphentheorie* (Springer-Lehrbuch Masterclass). Springer Berlin Heidelberg, 2006, ISBN: 9783540213918. [Online]. Available: https://books.google.de/books?id=pfTTwAEACAAJ.

[27] P. Bekemeyer *et al.*, “Data-Driven Aerodynamic Modeling Using the DLR SMARTy Toolbox,” in *AIAA AVIATION 2022 Forum*, AIAA 2022-3899. DOI: 10.2514/6.2022-3899. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2022-3899.

[28] R. L. Hardy, “Multiquadric equations of topography and other irregular surfaces,” *Journal of Geophysical Research (1896-1977)*, vol. 76, no. 8, pp. 1905–1915, 1971. DOI: https://doi.org/10.1029/JB076i008p01905.

[29] M. D. Buhmann, “Radial basis functions,” *Acta Numerica*, vol. 9, pp. 1–38, 2000. DOI: 10.1017/S0962492900000015.

[30] S. Y. Chung, S. Venkatramanan, H. E. Elzain, S. Selvam and M. Prasanna, “Chapter 4 - supplement of missing data in groundwater-level variations of peak type using geostatistical methods,” in *GIS and Geostatistical Techniques for Groundwater Science*, S. Venkatramanan, M. V. Prasanna and S. Y. Chung, Eds., Elsevier, 2019, pp. 33–41, ISBN: 978-0-12-815413-7. DOI: https://doi.org/10.1016/B978-0-12-815413-7.00004-3.

[31] A. Bertram, “Data-driven variable-fidelity reduced order modeling for efficient vehicle shape optimization,” en, Ph.D. dissertation, Nov. 2018. DOI: 10.24355/dbbs.084-201811231243-0.

[32] J. Sacks, W. J. Welch, T. J. Mitchell and H. P. Wynn, “Design and Analysis of Computer Experiments,” *Statistical Science*, vol. 4, no. 4, pp. 409–423, 1989. DOI: 10.1214/ss/1177012413.

[33] A. Forrester, A. Sobester and A. Keane, *Engineering Design Via Surrogate Modelling: A Practical Guide.* Jul. 2008, ISBN: 978-0-470-06068-1. DOI: 10.1002/9780470770801.

[34] M. Minsky and S. Papert, *Perceptrons.* Cambridge, MA: MIT Press, 1969.

[35] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas and H. S. Seung, “Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit,” *Nature*, vol. 405, no. 6789, pp. 947–951, Jun. 2000. DOI: 10.1038/35016072.

[36] D. E. Rumelhart, G. E. Hinton and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[37] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez and P. W. Battaglia, “Learning mesh-based simulation with graph networks,” *arXiv preprint arXiv:2010.03409*, 2020.

[38] A. Paszke *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.

cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[39] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[40] L. K. Loftin, "Theoretical and Experimental Data for a Number of NACA 6a-Series Airfoil Sections," National Advisory Committee for Aeronautics. Langley Aeronautical Lab, 1947.

[41] S. S. Davis, "NACA 64A010 (NASA Ames Model) Oscillatory Pitching," *Compendium of Unsteady Aerodynamic Measurements, AGARD R-702*, vol. 55, no. 7, pp. 2-1–2-22, 1982. DOI: 10.2514/1.J055373.

[42] J. H. Halton, "Algorithm 247: Radical-Inverse Quasi-Random Point Sequence," *Commun. ACM*, vol. 7, no. 12, pp. 701–702, Dec. 1964, ISSN: 0001-0782. DOI: 10.1145/355588.365104. [Online]. Available: https://doi.org/10.1145/355588.365104.

[43] P. SPALART and S. ALLMARAS, "A one-equation turbulence model for aerodynamic flows," in *30th Aerospace Sciences Meeting and Exhibit*. DOI: 10.2514/6.1992-439.

[44] M. A. Day, "The no-slip condition of fluid dynamics," *Erkenntnis (1975-)*, vol. 33, no. 3, pp. 285–296, 1990, ISSN: 01650106, 15728420. [Online]. Available: http://www.jstor.org/stable/20012308 (visited on 10/01/2024).

[45] J. Bassey, L. Qian and X. Li, "A survey of complex-valued neural networks," *arXiv preprint arXiv:2101.12249*, 2021.

[46] J. A. Barrachina, C. Ren, G. Vieillard, C. Morisseau and J.-P. Ovarlez, "Theory and implementation of complex-valued neural networks," *arXiv preprint arXiv:2302.08286*, 2023.

[47] T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[48] J. Bergstra, R. Bardenet, Y. Bengio and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira and K. Weinberger, Eds., vol. 24, Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[49] F. Hutter, H. Hoos and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, Bejing, China: PMLR, Jun. 2014, pp. 754–762. [Online]. Available: https://proceedings.mlr.press/v32/hutter14.html.

[50] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[51] S. Wassing, S. Langer and P. Bekemeyer, "Physics-informed neural networks for parametric compressible Euler equations," *Computers & Fluids*, vol. 270, p. 106 164, 2024, ISSN: 0045-7930. DOI: `https://doi.org/10.1016/j.compfluid.2023.106164`.

[52] D. Massegur Sampietro and A. Da Ronch, "Gradient-guided graph convolutional multi-mesh frameworks for aircraft aerodynamics modelling," in *AIAA SCITECH 2024 Forum*, 2024, p. 0456.

[53] W. W. Vaughan, "Guide to reference and standard atmospheric models," *American Institute of Aeronautics and Astronautics*, vol. AIAA G-003C-2010, Jan. 2010.

[54] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer series in statistics). Springer, 2009, ISBN: 9780387848846. [Online]. Available: `https://books.google.de/books?id=eBSgoAEACAAJ`.

List of Figures

2.1 Sketch of 1-cos gust with airfoil and parameters [5] 4
2.2 Schematic visualization of the task for a surrogate model 6

3.1 Exemplary attributed graph with node v_i that is connected to two neighbours. Each node has three features, each edge has two features. 8

4.1 Sampling of M and α as input variables for steady-state simulations 14
4.2 Mesh used in the study 15
4.3 Converged and not converged samples in the M-α-plane 16
4.4 c_l vs α for different Mach numbers 16
4.5 Final set of samples in the M-α-plane 17
4.6 Train-Validation-Test-Split in the M-α-plane 19

6.1 Maximum MAE of the Interpolation and the GNS Local model evaluated on the test data shown in the M-α plane. 25
6.2 MAE for the GNS Local model using all test samples in a specified range of Mach numbers shown in the α-ω^* plane. 25
6.3 Distribution of complex pressure increment of Interpolation and GNS Local model compared to LFD at $M = 0.44$, $\alpha = 2.88$. 26
6.4 Distribution of complex pressure increment of Interpolation and GNS Local model compared to LFD at $M = 0.72$, $\alpha = 5.29$. 27
6.5 Sampling in a defined area around $M = 0.72$ and $\alpha = 5.29$ 28
6.6 Distribution of complex pressure increment of Interpolation and GNS Local model compared to LFD at $M = 0.70$, $\alpha = 4.86$ (Training dataset). 29
6.7 Distribution of complex pressure increment of Interpolation and GNS Local model compared to LFD at $M = 0.77$, $\alpha = 3.37$. 30
6.8 Distribution of complex pressure increment of Interpolation and GNS Local model compared to LFD at $M = 0.73$, $\alpha = 2.08$. 31
6.9 MAE over the size of training data for averaged over 10 trials for each model 33
6.10 Schematic representation of the datasets used in the unseen frequencies investigations 34
6.11 MAE for prediction of models in the unseen frequency investigation 35
6.12 Comparison of Interpolation and GNS Local model trained on the entire dataset and trained on the reduced dataset. Note that labels on the abscissa are only valid for the models trained on the reduced dataset since the model trained on the full dataset has seen all ω^* during training. 35
6.13 Models trained on full dataset and models trained on samples $M \geq 0.6$ evaluated against each other using the MAE. 36
6.14 Samples selected for comparison of computational effort 38

A.1 Pairwise plotted distribution of each all possible features that are non-zero at the airfoil surface 53
A.2 HPO results of the Interpolation model 56
A.3 HPO results of the FCNN model 56
A.4 HPO results of the GCN Global model 57
A.5 HPO results of the GCN Local model 57
A.6 HPO results of the GNS Global model 57
A.7 HPO results of the GNS Local model 58
A.8 Sampling in a defined area around $M = 0.72$ and $\alpha = 5.29$ 59
A.9 Sampling in a defined area around $M = 0.72$ and $\alpha = 5.29$ 59

A.10 Distribution of complex pressure increment of FCNN, GCN Global, GCN Local and GNS Global model compared to LFD at $M = 0.44$, $\alpha = 2.88$. . . 60
A.11 Distribution of complex pressure increment of FCNN, GCN Global, GCN Local and GNS Global model compared to LFD at $M = 0.72$, $\alpha = 5.29$. . . 61
A.12 Distribution of complex pressure increment of FCNN, GCN Global, GCN Local and GNS Global model compared to LFD at $M = 0.73$, $\alpha = 2.08$. . . 62
A.13 Distribution of complex pressure increment of FCNN, GCN Global, GCN Local and GNS Global model compared to LFD at $M = 0.77$, $\alpha = 3.37$. . . 63
A.14 Training, validation, and test data of reduced datasets in the M-α plane . 64
A.15 Distribution of MAE over the size of training data for 10 trials for each model. 65
A.16 MAE over the size of training data for averaged over 10 trials for each model (First Try) 66
A.17 MAE over the size of training data for the GNS Local model. 66
A.18 MSE for prediction of models in the unseen frequency investigation 67
A.19 Relative Error for prediction of models in the unseen frequency investigation 67
A.20 M-α sample space for the investigation of high Mach number models. . . . 68
A.21 Models trained on full dataset and models trained on samples $M \geq 0.6$ evaluated against each other using the MSE. 68
A.22 Models trained on full dataset and models trained on samples $M \geq 0.6$ evaluated against each other using the relative Error. 69

List of Tables

3.1 Overview of the features and targets used for the different models using the generic variables introduced in chapter 2. The differentiation between vectors and values with the suffix i gives information on whether the approach works for all nodes at once or node-wise. 12
3.2 Advantages and disadvantages of the different models 13

5.1 Overview of the features and targets used for the different models. The differentiation between vectors and values with the suffix i gives information on whether the approach works for all nodes at once or node-wise. 20

6.1 Models after HPO evaluated on different metrics. The best score for each metric is highlighted. 24
6.2 Number of samples used in the training, validation, and test datasets that are used to investigate the size of the dataset 32
6.3 Comparison of computational effort. The relative training time is normalized using the Interpolation training time. The relative numbers of parameters are normalized using the number of parameters of the Interpolation model. The evaluation time is normalized using the LFD evaluation time. 37

A.1 RBF types implemented in SMARTy 52
A.2 Gaussian process regression kernels implemented in SMARTy 52
A.3 Hyperparameters of the Interpolation model 54
A.4 Hyperparameters of the FCNN model 54
A.5 Hyperparameters of the GCN Global model 55
A.6 Hyperparameters of the GCN Local model 55
A.7 Hyperparameters of the GNS Global model 55
A.8 Hyperparameters of the GNS Local model 56

Chapter A

Appendix

A.1 Data-Driven Methods

Name	Definition
Linear	$\varphi(r) = r$
Square	$\varphi(r) = r^2$
Cubic	$\varphi(r) = r^3$
Thin Plate Spline (TPS)	$\varphi(r) = r^2 ln(r)$

Table A.1: RBF types implemented in SMARTy

Kernel
Gaussian
GaussianExponential
CubicSpline

Table A.2: Gaussian process regression kernels implemented in SMARTy

A.2 Test Case

A.2.1 Reynolds Number

Using the density ρ, velocity u, Reynolds length L, and dynamic viscosity μ, the Reynolds Re number is defined as

$$Re = \frac{\rho u L}{\mu}. \tag{A.1}$$

In the context of CFD the Reynolds length is determined by the grid size of the mesh. As the same mesh is used for all samples L is constant. The dynamic viscosity is a property of the air and is constant for a fixed temperature. It is fixed in this case. Since the velocity is dependent on the Mach number, it is variable for different samples. The density is the only parameter that is left in the equation and consequently must be variable too. In the context of air travel this means the simulations are conducted at different altitudes as the density is a function of the altitude [53].

A.2.2 Features

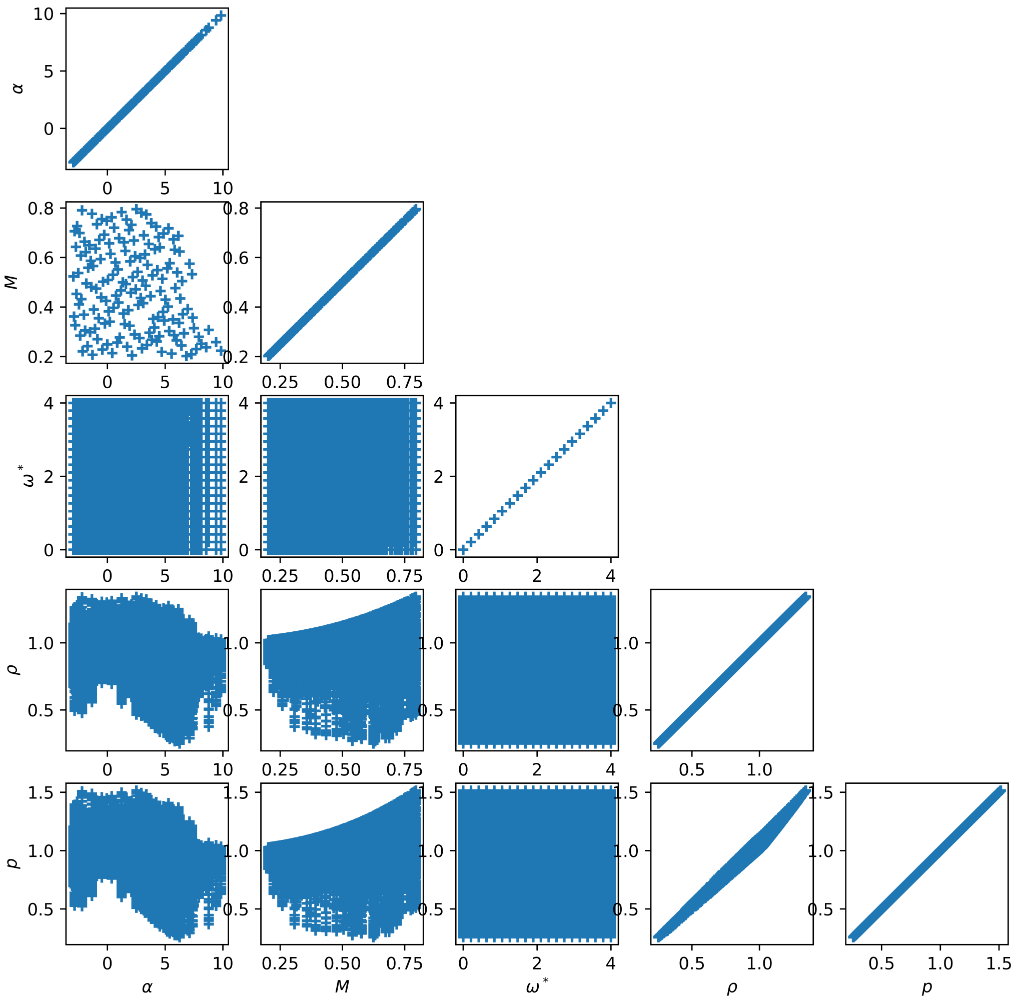


Figure A.1: Pairwise plotted distribution of each all possible features that are non-zero at the airfoil surface

A.3 Model Selection

A.3.1 Scaling

The performance of many ML approaches can be affected by the range of the values in a dataset. Scaling of the data can lead to improvements in a model's performance. Assuming a Gaussian distribution of the data standardization is done by transforming the data to have zero as the mean and a standard deviation of one. A second approach to scaling is *MinMax*-scaling which linearly transforms the data to a new range. Assuming the values x_i with $i = 1, .., n$ to be stored in vector $\mathbf{x} \in \mathbb{R}^n$ MinMax-scaling is done as follows:

$$x_{i,scaled} = \frac{x_i - \min \mathbf{x}}{\max \mathbf{x} - \min \mathbf{x}} \tag{A.2}$$

Scaling can be done for both, features and targets. Equation A.2 is implemented in SMARTy and is used throughout this work [27, 54].

A.3.2 Hyperparameters

Hyperparameter	**Type**	**Range**	**Optimal Value**
`kernel`	categorical	[TPS, Cubic, Square, Linear, Kriging, CubicSpline, Gaussian, GaussianExponential]	Linear
`augmentation`	integer	[-1, 2]	0
`regularization`	categorical	[0, 0.0001, 0.001, 0.01, 0.1, 1]	0
`scale`	categorical	[True, False]	True

Table A.3: Hyperparameters of the Interpolation model

Hyperparameter	**Type**	**Range**	**Optimal Value**
`numHiddenLayers`	integer	[5, 12]	8
`numNeuronsPerLayer`	integer	[64, 128, 256, 512]	512
`learningRate`	float from log-domain	[1e-5, 1e-2]	0.001083
`gamma`	float	[0.1, 1]	0.640675
`milestoneFactor`	integer from log-domain	[10, 1000]	11

Table A.4: Hyperparameters of the FCNN model

Hyperparameter	**Type**	**Range**	**Optimal Value**
`numHiddenLayers`	integer	[5, 12]	9
`numNeuronsPerLayer`	integer	[64, 128, 256, 512]	512
`learningRate`	float from log-domain	[1e-5, 1e-2]	0.000226
`gamma`	float	[0.1, 1]	0.116663
`milestoneFactor`	integer from log-domain	[10, 1000]	140

Table A.5: Hyperparameters of the GCN Global model

Hyperparameter	**Type**	**Range**	**Optimal Value**
`numHiddenLayers`	integer	[5, 12]	11
`numNeuronsPerLayer`	integer	[64, 128, 256, 512]	512
`learningRate`	float from log-domain	[1e-5, 1e-2]	0.000443
`gamma`	float	[0.1, 1]	0.363582
`milestoneFactor`	integer from log-domain	[10, 1000]	132

Table A.6: Hyperparameters of the GCN Local model

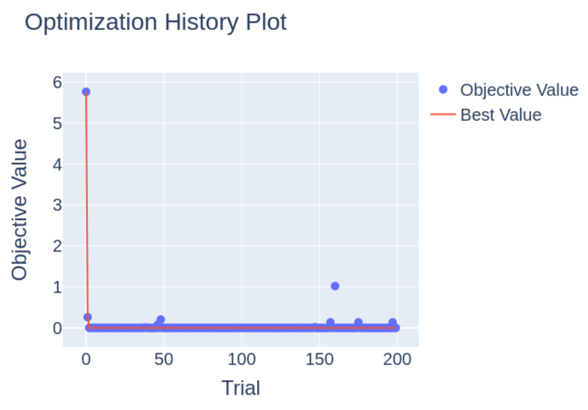
Hyperparameter	**Type**	**Range**	**Optimal Value**
`numHiddenLayersEncoder`	integer	[5, 12]	8
`numNeurons`	categorical	[64, 128, 256, 512]	256
`numLayersProcessor`	integer	[1, 4]	1
`learningRate`	float from log-domain	[1e-5, 1e-2]	0.000654
`gamma`	float	[0.1, 1]	0.495361
`milestoneFactor`	integer from log-domain	[10, 1000]	57

Table A.7: Hyperparameters of the GNS Global model

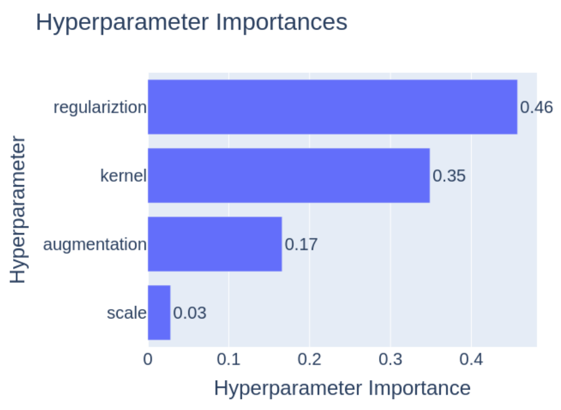
Hyperparameter	Type	Range	Optimal Value
`numHiddenLayersEncoder`	integer	[5, 12]	7
`numNeurons`	categorical	[64, 128, 256, 512]	512
`numLayersProcessor`	integer	[1, 4]	1
`learningRate`	float from log-domain	[1e-5, 1e-2]	0.000914
`gamma`	float	[0.1, 1]	0.468220
`milestoneFactor`	integer from log-domain	[10, 1000]	82

Table A.8: Hyperparameters of the GNS Local model

A.3.3 HPO Results

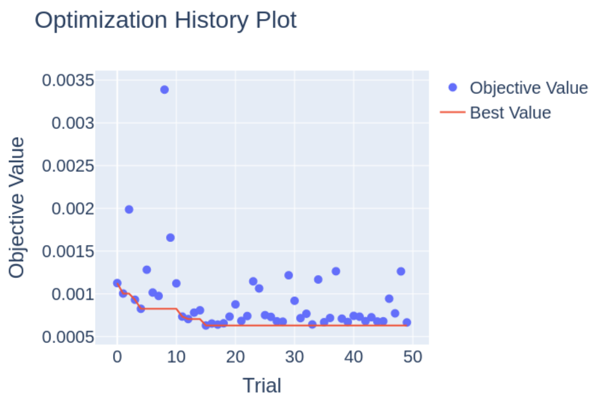


(a) Optimization history

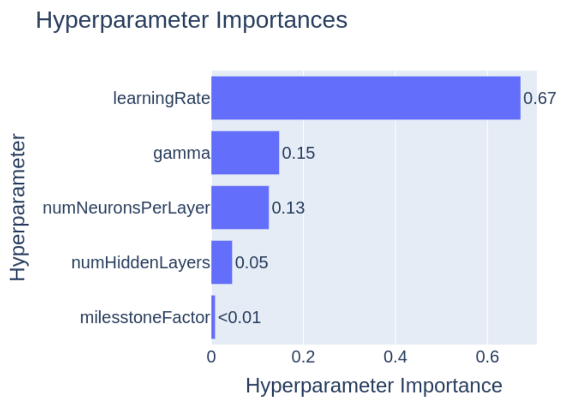


(b) Parameter importances

Figure A.2: HPO results of the Interpolation model

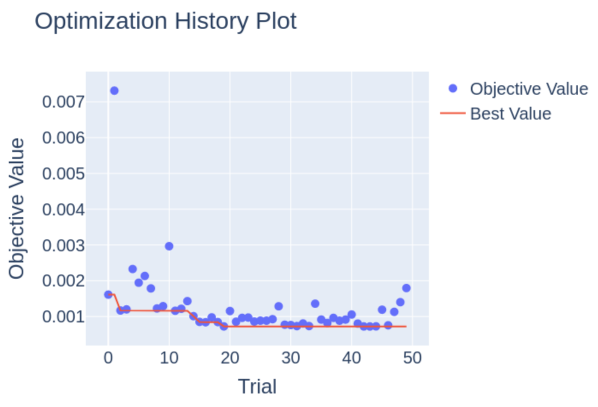


(a) Optimization history

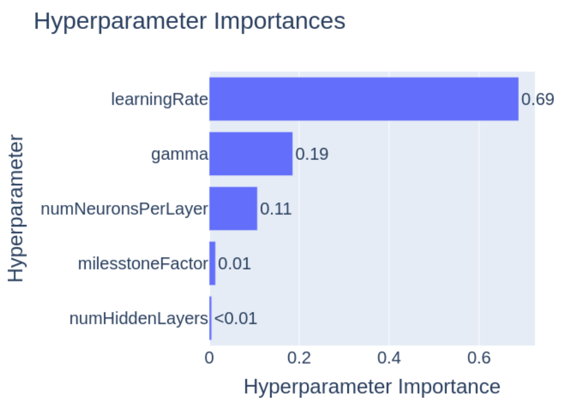


(b) Parameter importances

Figure A.3: HPO results of the FCNN model

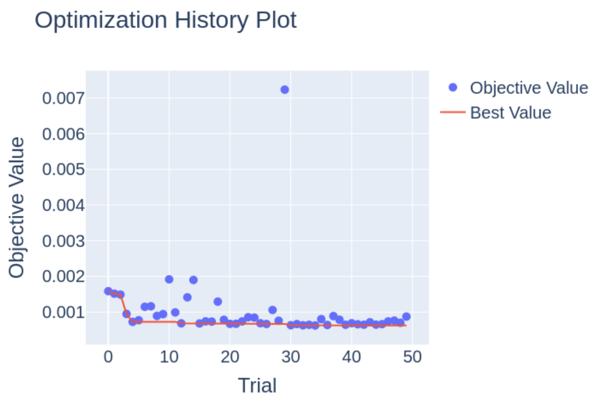


(a) Optimization history

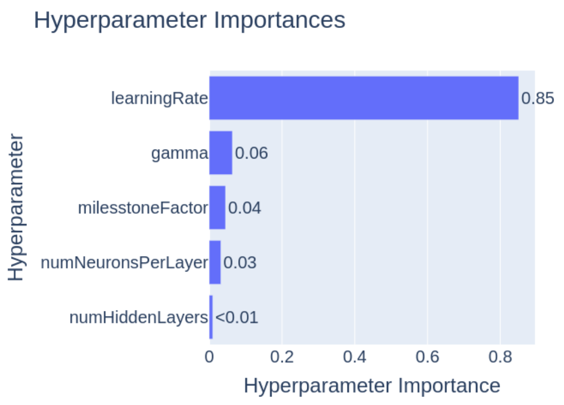


(b) Parameter importances

Figure A.4: HPO results of the GCN Global model

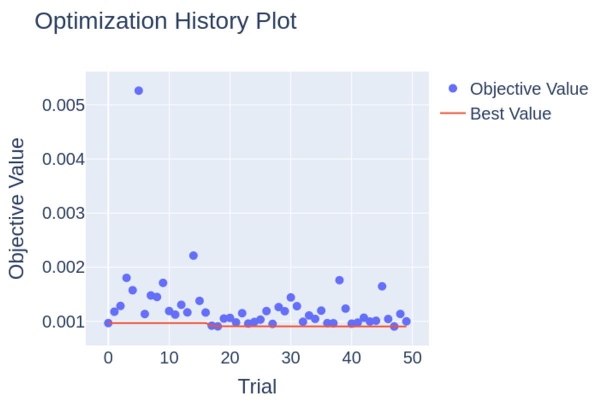


(a) Optimization history

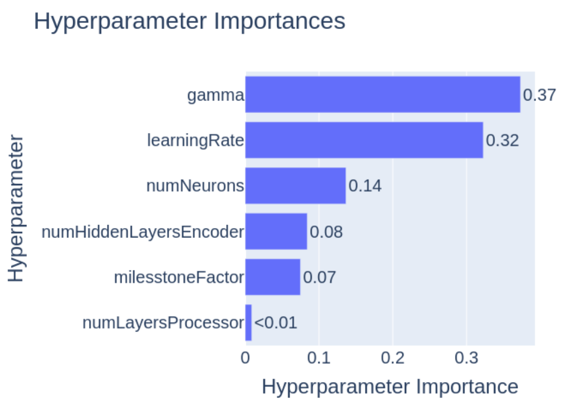


(b) Parameter importances

Figure A.5: HPO results of the GCN Local model

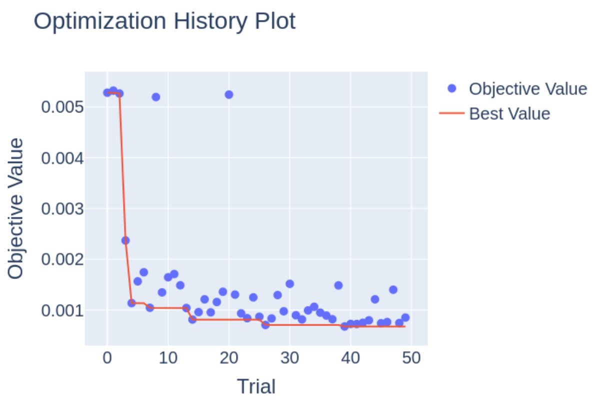


(a) Optimization history

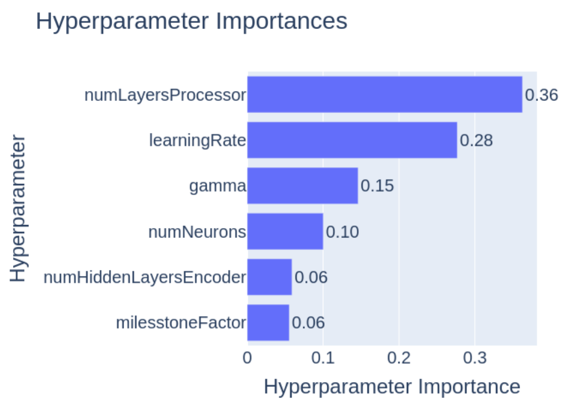


(b) Parameter importances

Figure A.6: HPO results of the GNS Global model



(a) Optimization history

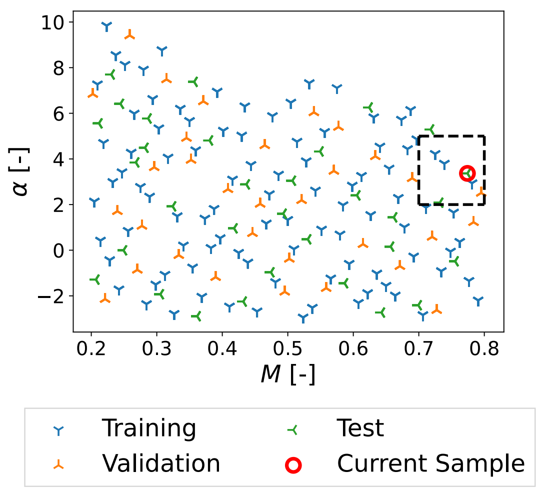


(b) Parameter importances

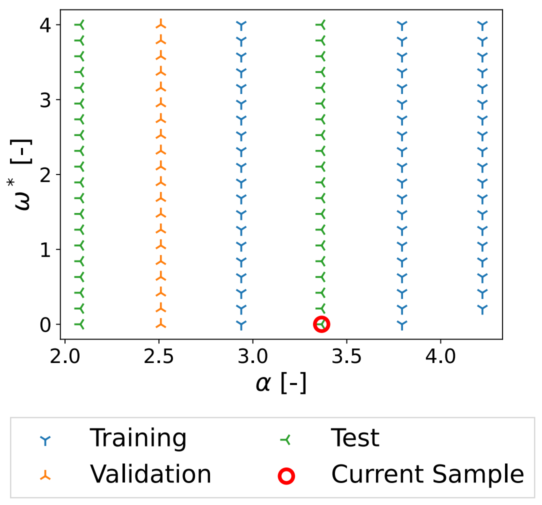
Figure A.7: HPO results of the GNS Local model

A.4 Results

A.4.1 Models after HPO

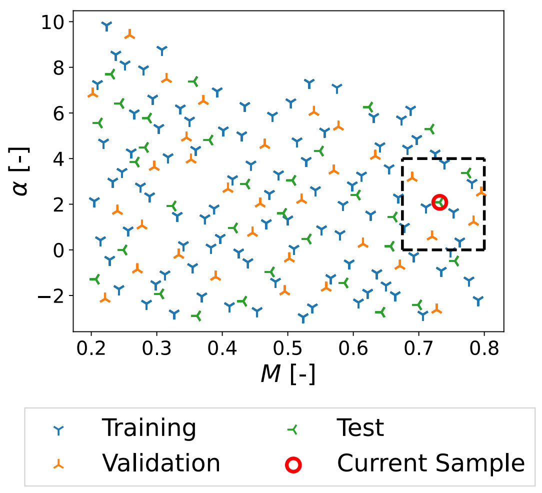


(a) Definition of region that is looked at in figure A.8b

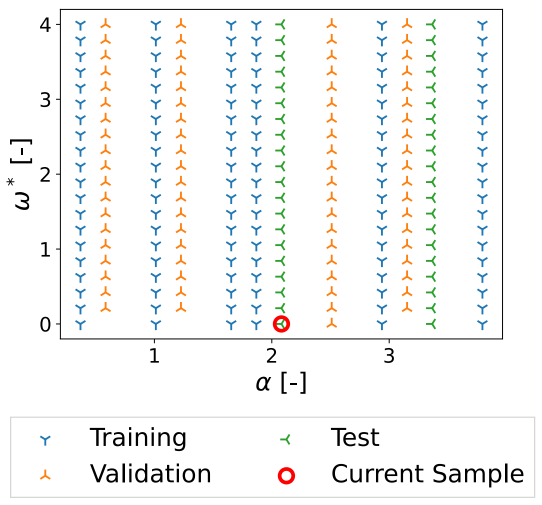


(b) Samples from the region defined in figure A.8a represented in the α-ω^* plane

Figure A.8: Sampling in a defined area around $M = 0.72$ and $\alpha = 5.29$



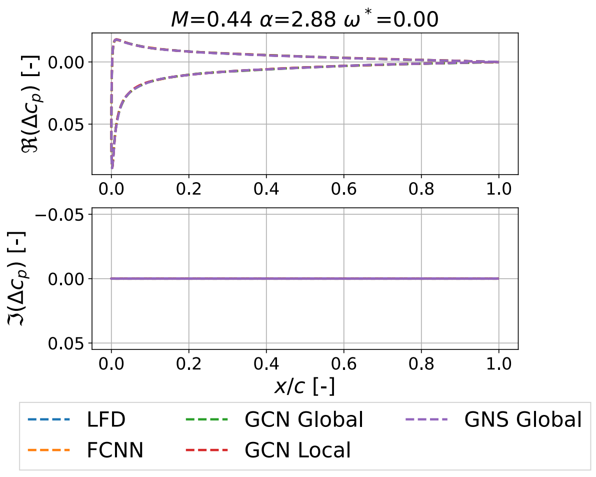
(a) Definition of region that is looked at in figure A.9b



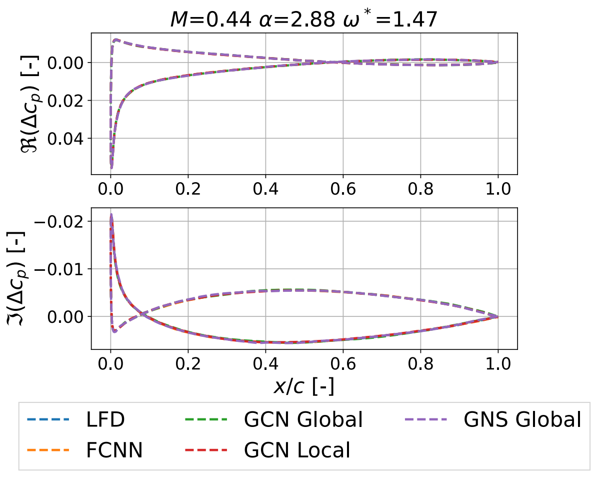
(b) Samples from the region defined in figure A.9a represented in the α-ω^* plane

Figure A.9: Sampling in a defined area around $M = 0.72$ and $\alpha = 5.29$

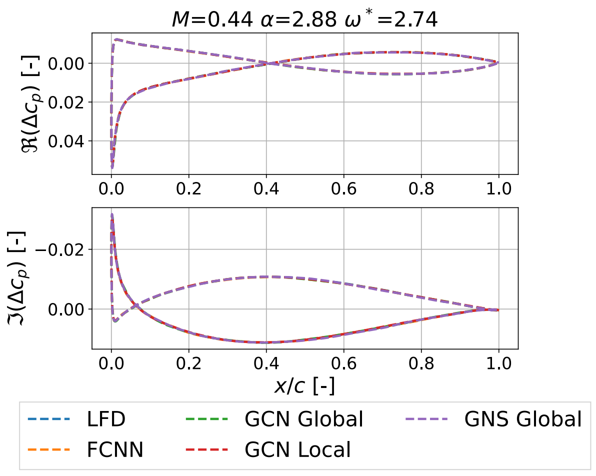
A.4.2 Complementary Figures for all Models



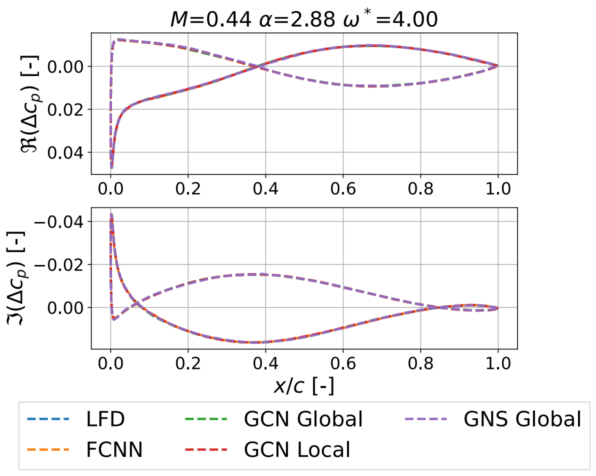
(a) $\omega^* = 0.0$



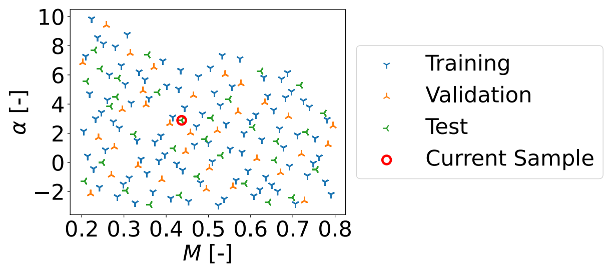
(b) $\omega^* = 1.47$



(c) $\omega^* = 2.74$

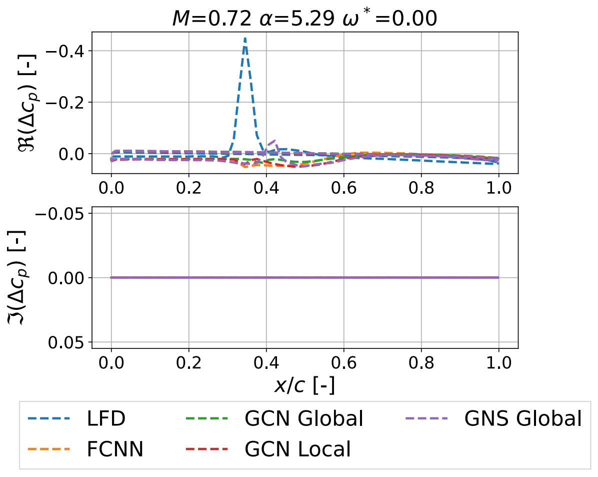


(d) $\omega^* = 4.00$

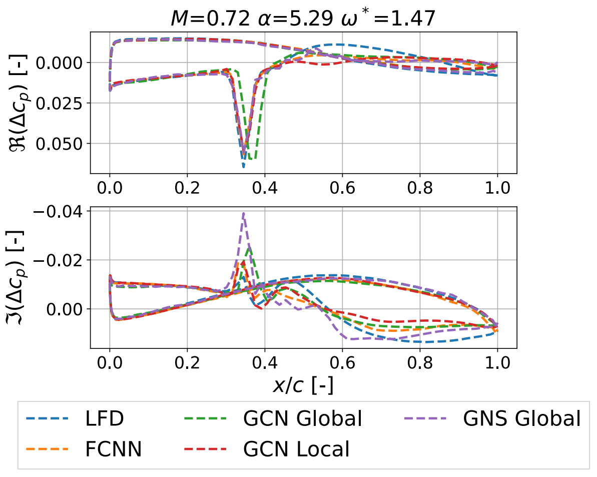


(e) Location of sample in M-α plane

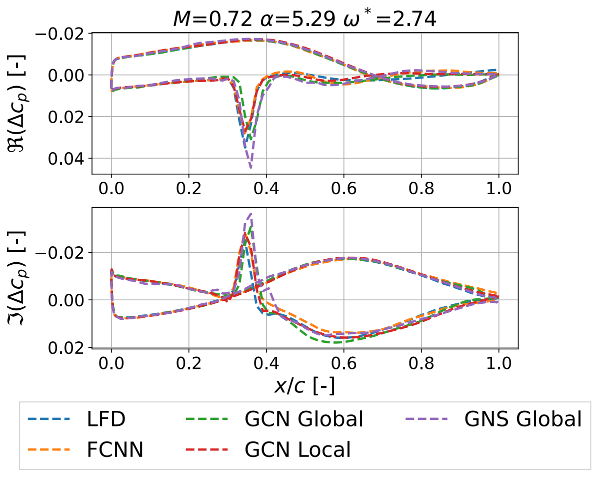
Figure A.10: Distribution of complex pressure increment of FCNN, GCN Global, GCN Local and GNS Global model compared to LFD at $M = 0.44$, $\alpha = 2.88$.



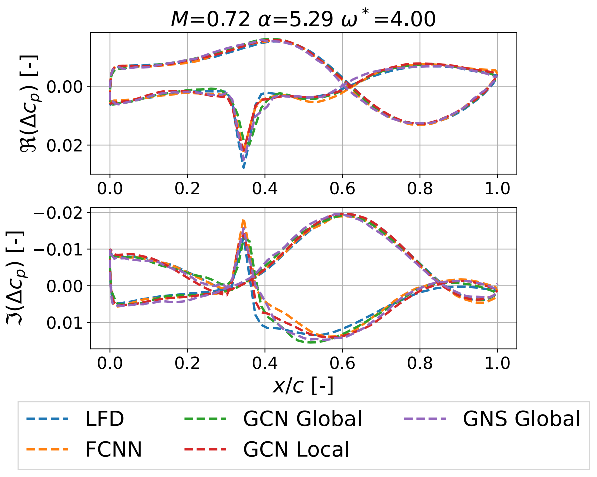
(a) $\omega^* = 0.0$



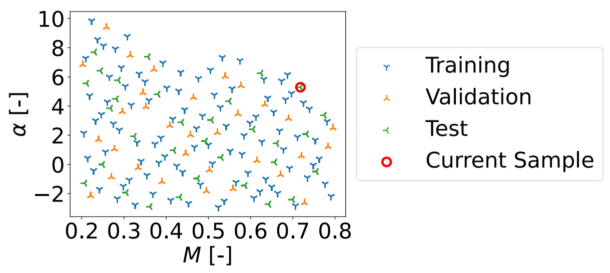
(b) $\omega^* = 1.47$



(c) $\omega^* = 2.74$

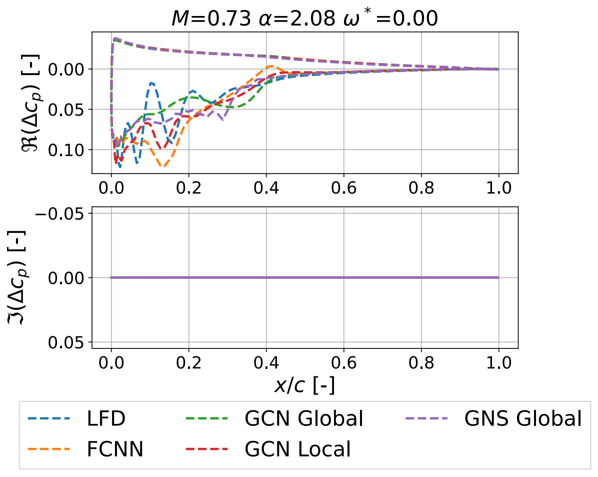


(d) $\omega^* = 4.00$

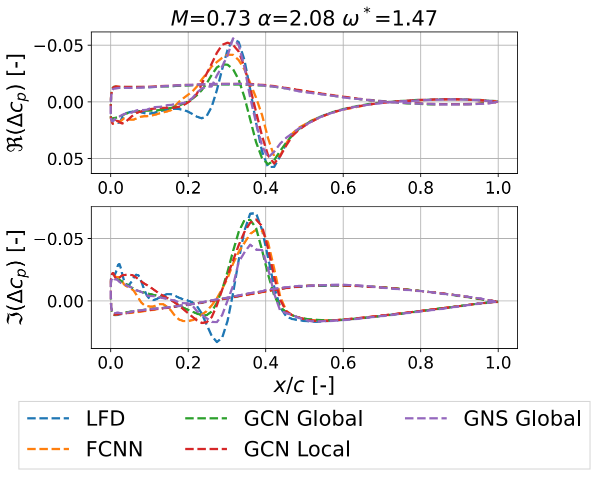


(e) Location of sample in M-α plane

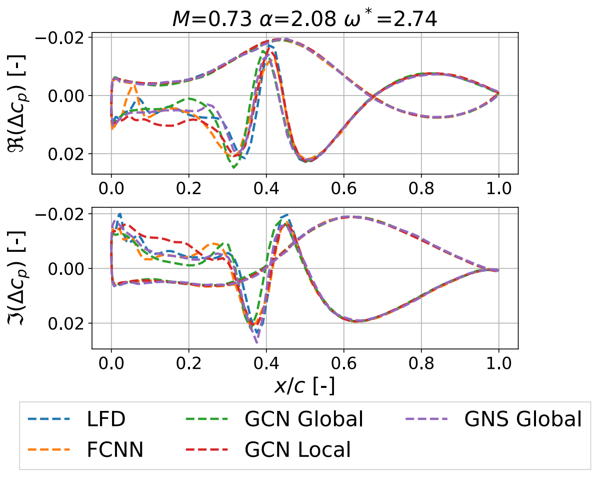
Figure A.11: Distribution of complex pressure increment of FCNN, GCN Global, GCN Local and GNS Global model compared to LFD at $M = 0.72$, $\alpha = 5.29$.



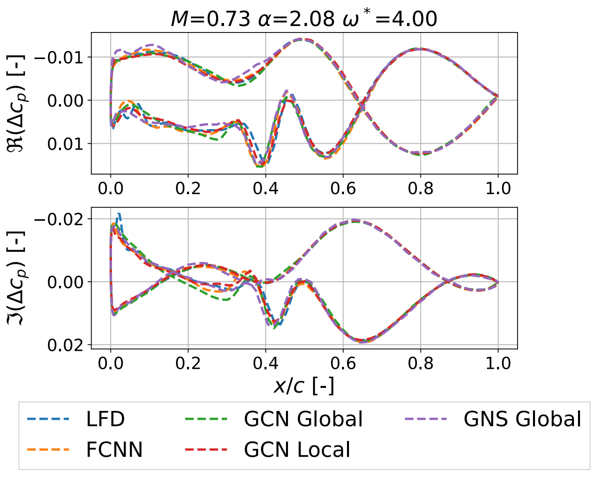
(a) $\omega^* = 0.0$



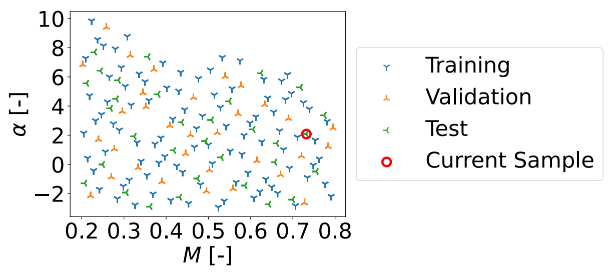
(b) $\omega^* = 1.47$



(c) $\omega^* = 2.74$

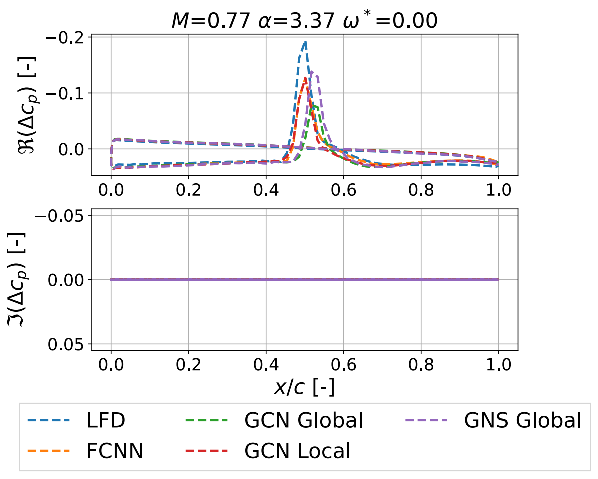


(d) $\omega^* = 4.00$

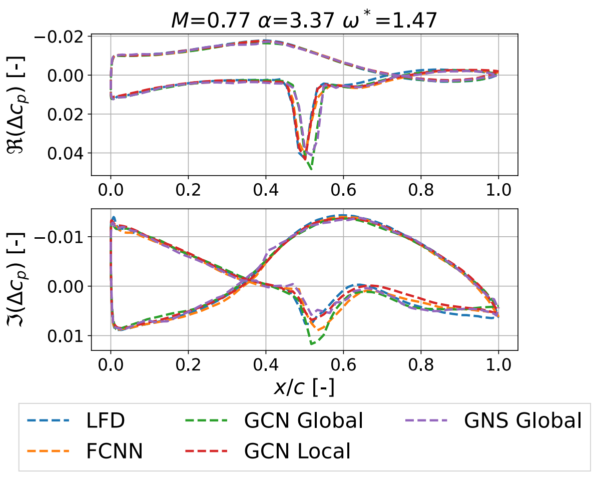


(e) Location of sample in M-α plane

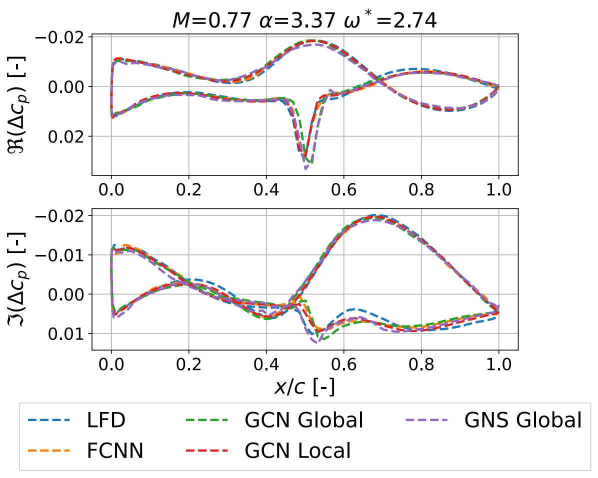
Figure A.12: Distribution of complex pressure increment of FCNN, GCN Global, GCN Local and GNS Global model compared to LFD at $M = 0.73$, $\alpha = 2.08$.



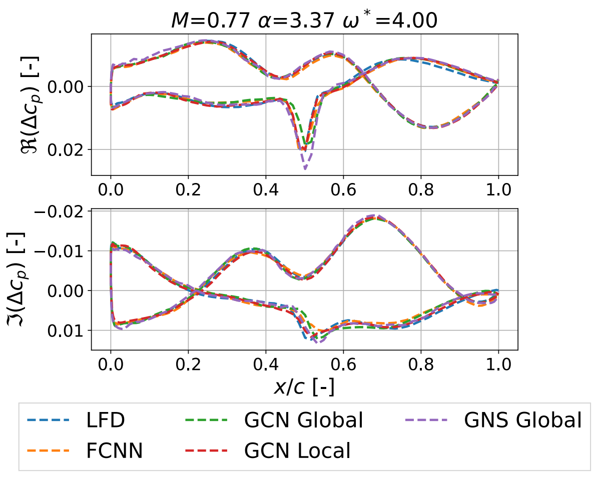
(a) $\omega^* = 0.0$



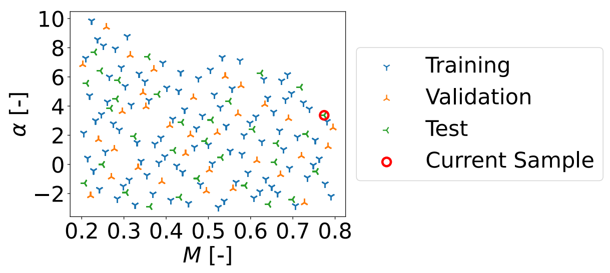
(b) $\omega^* = 1.47$



(c) $\omega^* = 2.74$



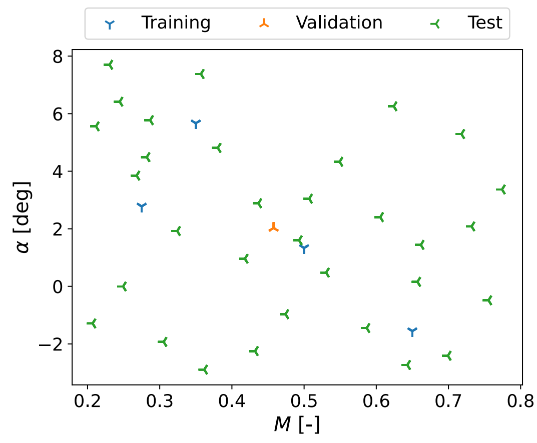
(d) $\omega^* = 4.00$



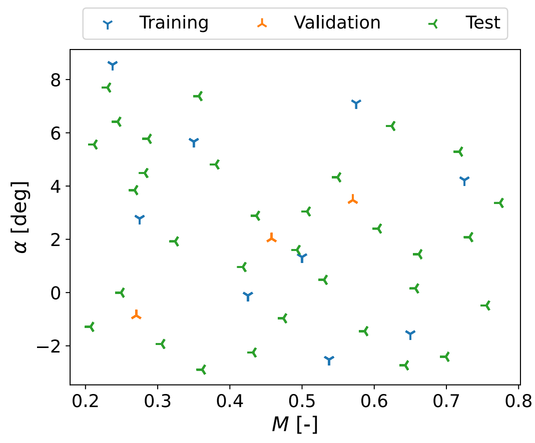
(e) Location of sample in M-α plane

Figure A.13: Distribution of complex pressure increment of FCNN, GCN Global, GCN Local and GNS Global model compared to LFD at $M = 0.77$, $\alpha = 3.37$.

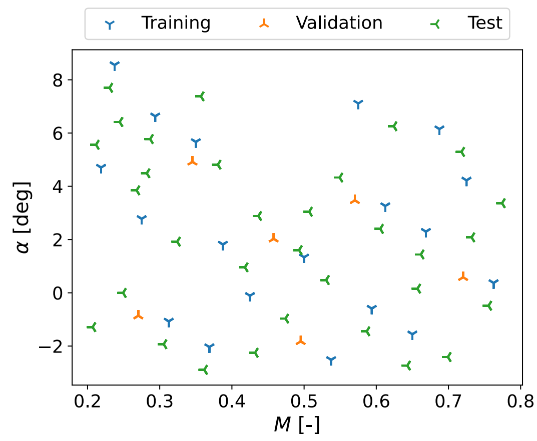
A.4.3 Size of Dataset



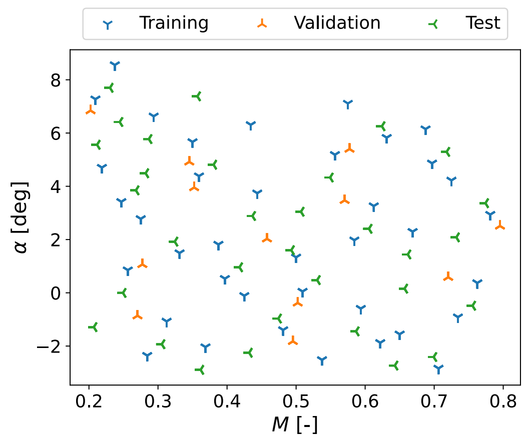
(a) 5%



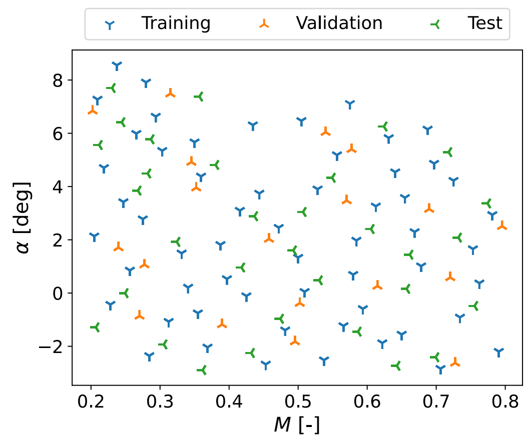
(b) 10%



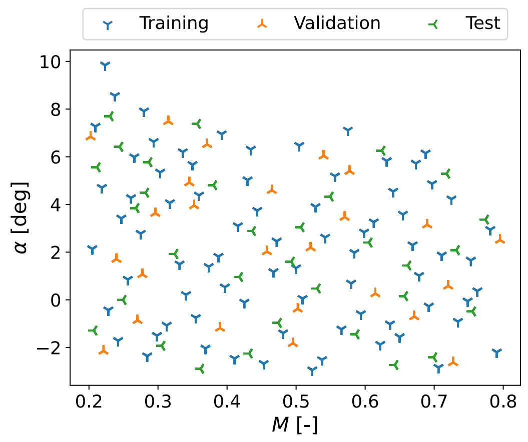
(c) 20%



(d) 40%



(e) 60%



(f) 80%

Figure A.14: Training, validation, and test data of reduced datasets in the M-α plane

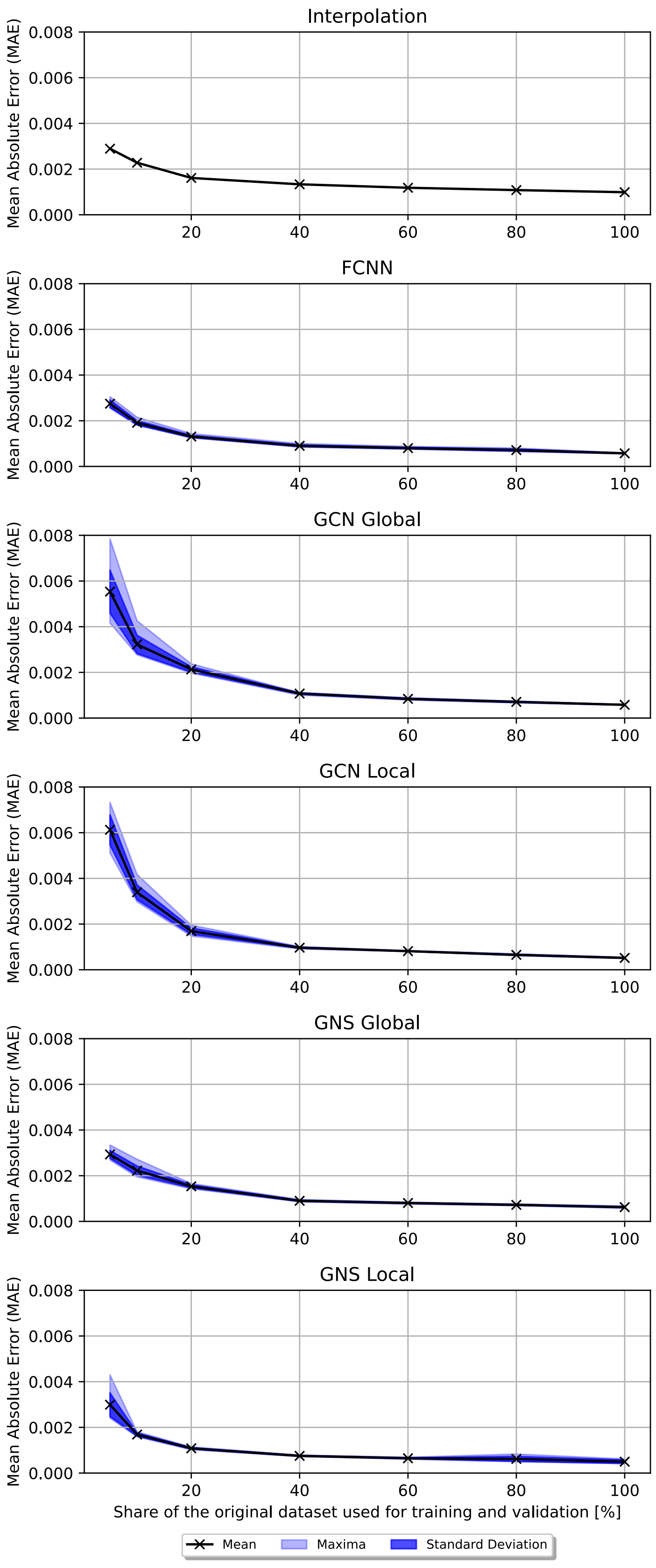


Figure A.15: Distribution of MAE over the size of training data for 10 trials for each model.

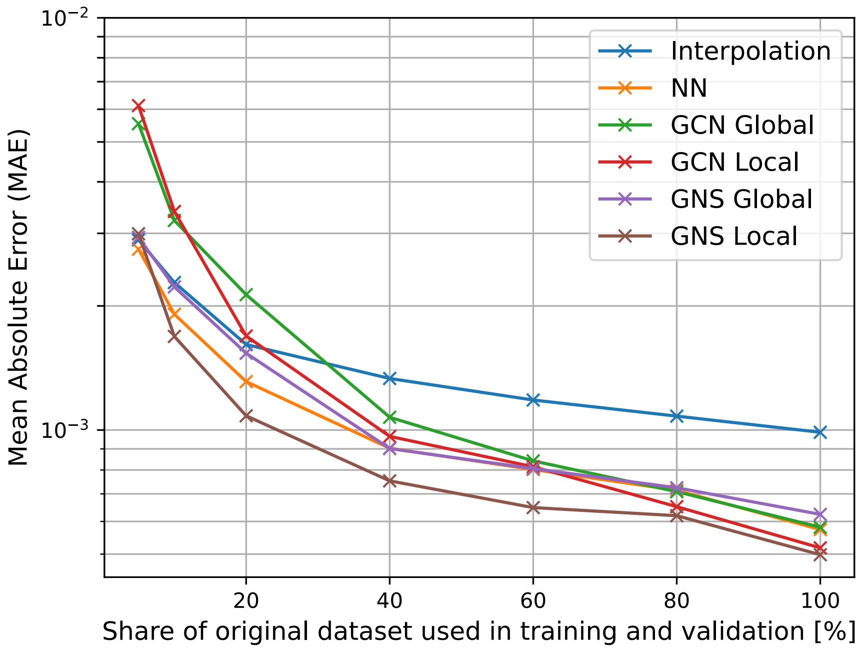


Figure A.16: MAE over the size of training data for averaged over 10 trials for each model (First Try)

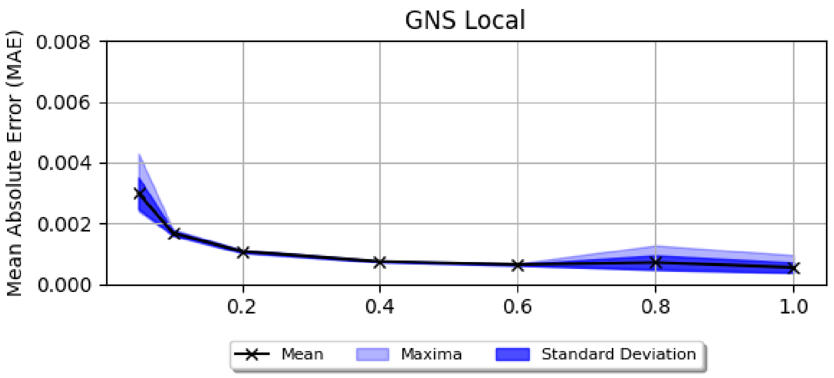


Figure A.17: MAE over the size of training data for the GNS Local model.

A.4.4 Unseen Frequencies

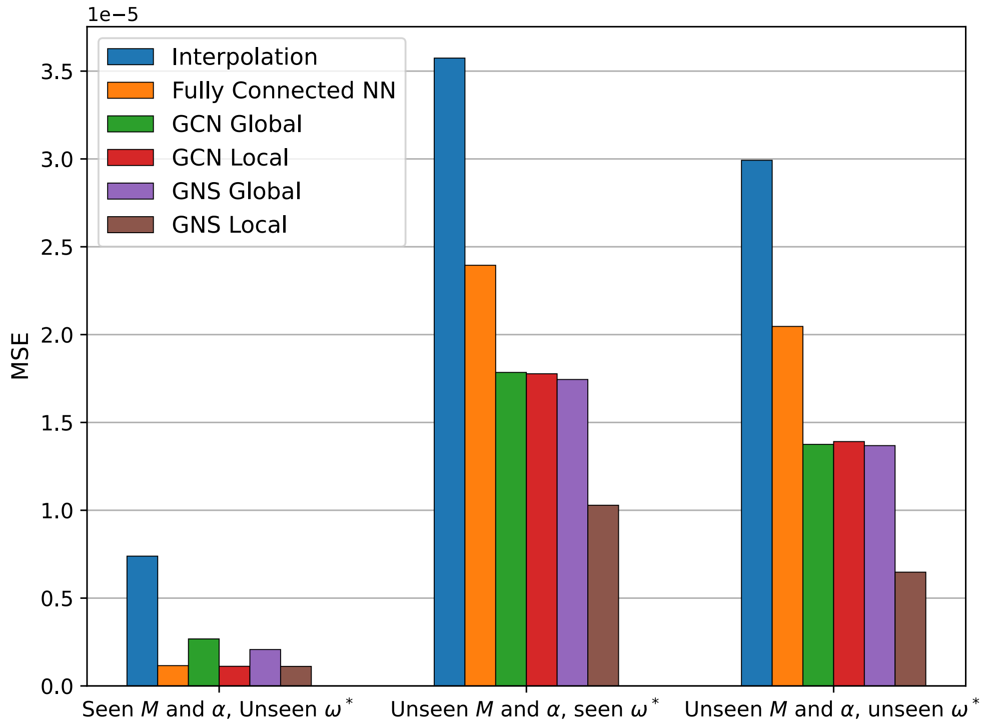


Figure A.18: MSE for prediction of models in the unseen frequency investigation

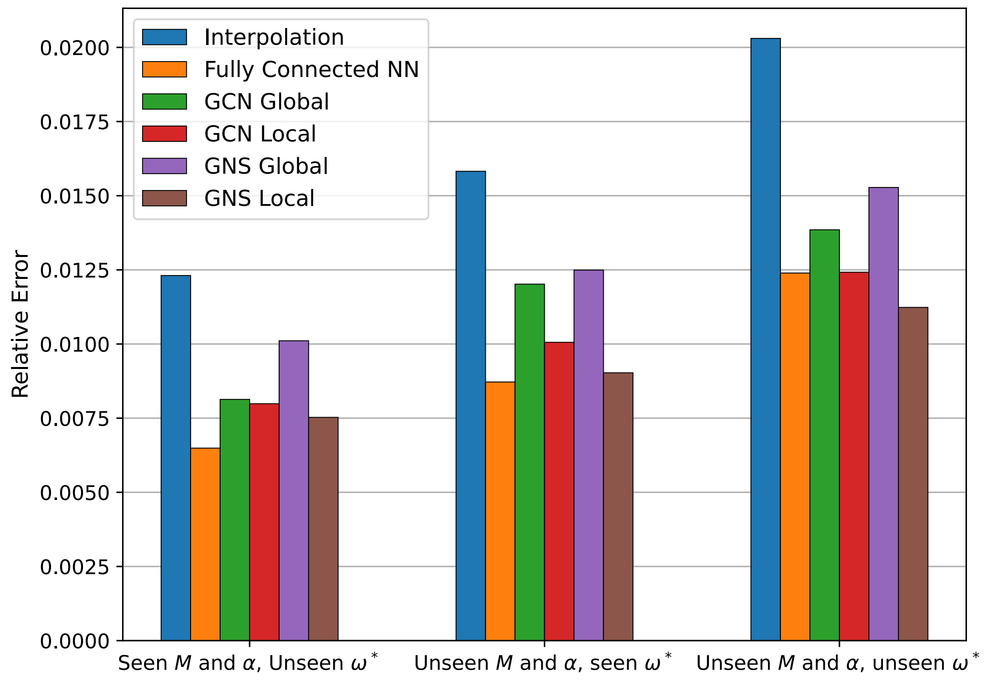


Figure A.19: Relative Error for prediction of models in the unseen frequency investigation

A.4.5 High Mach Number Models

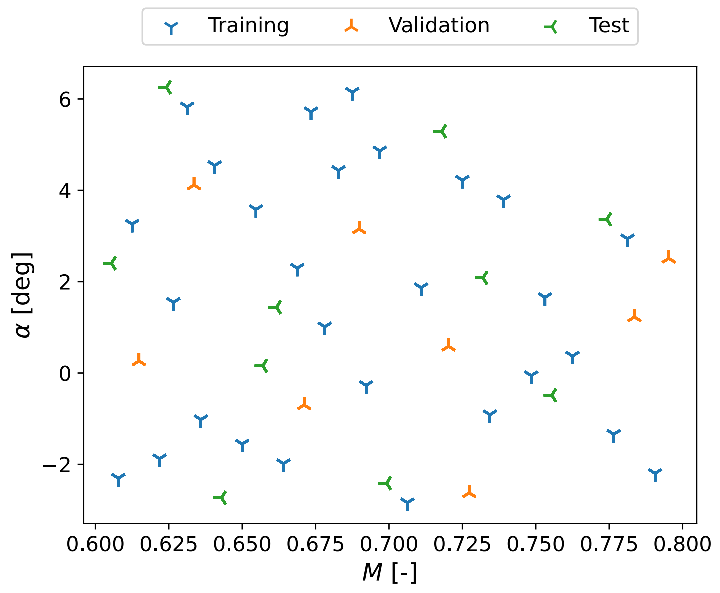


Figure A.20: M-α sample space for the investigation of high Mach number models.

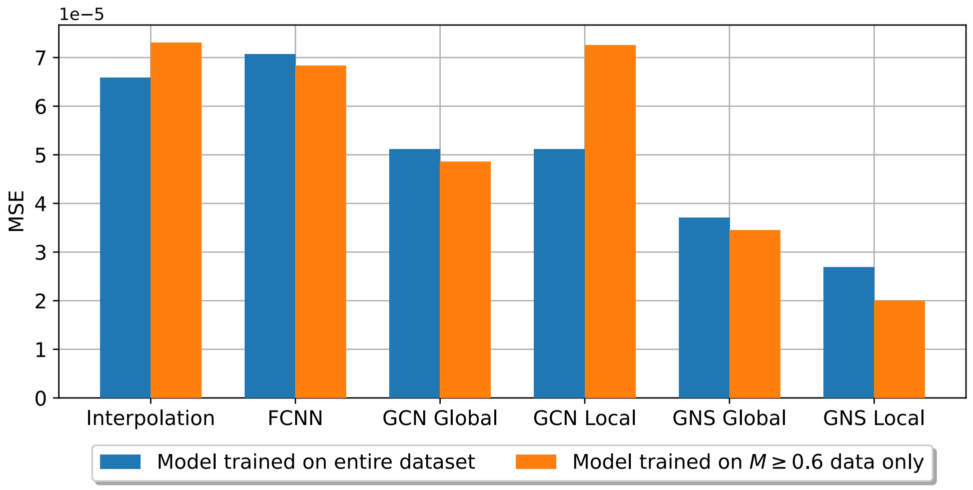


Figure A.21: Models trained on full dataset and models trained on samples $M \geq 0.6$ evaluated against each other using the MSE.

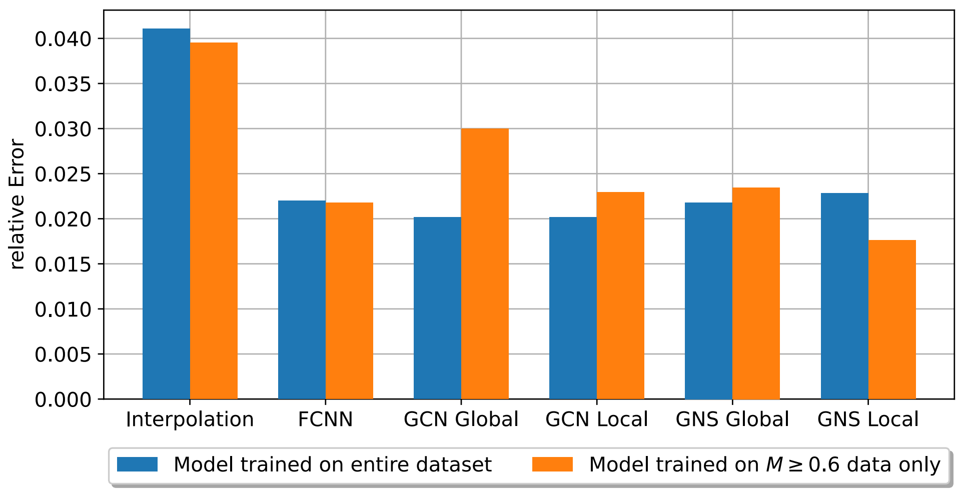


Figure A.22: Models trained on full dataset and models trained on samples $M \geq 0.6$ evaluated against each other using the relative Error.