TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechanical Engineering

von KARMAN INSTITUTE
FOR FLUID DYNAMICS

Inferring wall pressure spectral model using neural networks

Master thesis

Study programme: M2301 – Mechanical Engineering
Study branch: 3901T003 – Applied Mechanics

Author: **Bc. Jan Bayer**
Supervisors: Prof. Ing. Karel Fraňa, Ph.D.
Prof. Dr. Ir. Miguel Alfonso Mendez
Ir. Joachim Dominique

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Zadání diplomové práce

Inferring wall pressure spectral model using neural networks

Jméno a příjmení: **Bc. Jan Bayer**
Osobní číslo: S19000211
Studijní program: M2301 Strojní inženýrství
Studijní obor: Aplikovaná mechanika – mechanika tekutin a termodynamika
Zadávající katedra: Katedra energetických zařízení
Akademický rok: **2020/2021**

Zásady pro vypracování:

1. Prepare a database of wall pressure spectra and their respective boundary layer parameters (from experiments or CFD)
2. Select proper boundary layer parameters for the model
3. Use a neural network to predict wall pressure spectra
4. Use a time series neural network to predict wall pressure spectra

Rozsah grafických prací: 15
Rozsah pracovní zprávy: 70
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Angličtina

Seznam odborné literatury:

[1] HIRSCH, Ch. *Numerical computation of internal and external flows*. Volume 1, Fundamentals of computational fluid dynamics. 2nd ed. Oxford: Elsevier, 2007. ISBN 978-0-7506-6594-0.
[2] CHUNG, T. J. *Computational fluid dynamics*. Cambridge: Cambridge University Press, 2002. ISBN 0-521-59416-2.
[3] VERSTEEG, H. K. a W. MALALASEKERA. *An introduction to computational fluid dynamics: the finite volume method*. 2nd ed. Harlow: Pearson Prentice Hall, 2007. ISBN 978-0-13-127498-3.

Vedoucí práce: prof. Ing. Karel Fraňa, Ph.D.
Katedra energetických zařízení

Datum zadání práce: 1. listopadu 2020
Předpokládaný termín odevzdání: 30. dubna 2022

L.S.

prof. Dr. Ing. Petr Lenfeld
děkan

doc. Ing. Petra Dančová, Ph.D.
vedoucí katedry

V Liberci dne 1. listopadu 2020

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

4. května 2021 Bc. Jan Bayer

Inferring wall pressure spectral model using neural networks

Abstrakt

Tato diplomová práce se zabývá modelováním spektra tlakových fluktuací u stěny v turbulentní mezní vrstvě založeným na datech. Současné semi-empirické modely predikují spektrum tlakových fluktuací u stěny z čistě lokálních veličin mezní vrstvy (přístup One-to-One). Historie proudění je v těchto modelech reprezentována pouze některými parametry. Nový přístup navrhuje použít k predikci spektra tlakových fluktuací u stěny v jednom bodě více bodů proti proudu tekutiny (přístup Many-to-One). Neuronové sítě jako modely založené na datech jsou navrženy pro model jak s přístupem One-to-One, tak i Many-to-One. Použitá databáze se skládá z jednoho experimentálního a tří numerických (proudění okolo controlled-diffusion profilu) souborů dat. Relevantní parametry mezní vrstvy jsou extrahovány z databáze a zpracovány do bezrozměrného vztahu. Nejprve se natrénují dva One-to-One modely s dopřednou neuronovou sítí. Oba dobře predikují spektra z databáze. Jeden z modelů je natrénován pouze na souboru dat s controlled-diffusion profily, aby byla k dispozici srovnávací architektura. Za druhé jsou natrénovány dvě Many-to-One dopředné neuronové sítě se dvěma a deseti pozicemi v mezní vrstvě. Model se dvěma pozicemi vylepšuje model One-to-One. Desetipoziční model takové zlepšení nepřináší. Oba tyto modely však predikují spektra poměrně dobře. Zatřetí, Many-to-One model konvoluční neuronové sítě je natrénován se vstupy s proměnným počtem pozic. Model však na proměnný počet pozic nereaguje dobře. Predikce modelu tedy nebyla dobrá. Kromě aplikace neuronových sítí je teoreticky zkoumáno použití modelů Many-to-One. Závěrem lze k novému přístupu říci, že model Many-to-One se dvěma pozicemi sice přináší určité zlepšení, ale není tak výrazné. Potvrzuje se, že model Many-to-One využívá hodnoty, které jsou proti proudu tekutiny a že modely Many-to-One mohou predikovat spektra tlakových fluktuací u stěny.

Klíčová slova:

model pro spektra tlakových fluktuací u stěny, turbulentní mezní vrstva, strojové učení, neuronová síť

Inferring wall pressure spectral model using neural networks

Abstract

This master thesis deals with a data-driven approach to the turbulent boundary layer wall pressure spectral modelling. Current wall pressure spectra semi-empirical models predict the spectra from purely local boundary layer quantities (a One-to-One approach). The flow history is only represented by some parameters in these models. A novel approach suggests using multiple upstream points to predict the wall pressure spectra at one point (a Many-to-One approach). Neural networks as data-driven models are proposed to build a model with a One-to-One and Many-to-One approach. A database of one experimental and three numerical (controlled-diffusion airfoil) datasets is used. Relevant boundary layer parameters are extracted from the database and processed into a dimensionless relation. Firstly, two One-to-One models with a feedforward neural network are trained. Both of them predict the spectra from the dataset reasonably well. One of the models is trained only on the airfoil datasets to have a comparison architecture. Secondly, two Many-to-One feedforward neural networks are trained with two and ten positions in the inputs. The two positional model improves the One-to-One model. The ten positional model does not bring such improvement. Both of these models predict the spectra reasonably well. Thirdly, a Many-to-One convolution neural network model is trained with variable-length inputs. However, the model did not react well on the variable-length inputs. The prediction of the model was not great. In addition to these models, the use of the Many-to-One models is investigated in theory. To conclude the novel approach, the two positional Many-to-One model does bring some improvement, but it is not that significant. However, the investigation confirms that the upstream values are used in the Many-to-One models and that the Many-to-One models can predict the wall pressure spectra.

Keywords:

wall pressure spectral model, turbulent boundary layer, machine learning, neural network

Acknowledgements

I would like to thank Ir. Joachim Dominique for providing his time in guiding my work and for the opportunity and such a challenging topic. The almost every week session was really enriching, and it provided me with valuable knowledge.

I would also like to thank Prof. Dr. Ir. Miguel Alfonso Mendez for the supervision of the work and fruitful discussions, and his optimism and openness.

Thanks should also be given to Prof. Ing. Karel Fraňa, Ph.D. for the supervision at the home university.

This master thesis was done in cooperation with the von Karman Institute for Fluid Dynamics (VKI).

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
AOA	Angle of Attack
APG	Adverse Pressure Gradient
CD	Controlled-Diffusion (airfoil)
CFD	Computational Fluid Dynamics
CNN	Convolution Neural Network
CNNM	Convolution Neural Network Model
DNC	Direct Noise Computation
DNS	Direct Numerical Simulation
FNN	Feedforward Neural Network
FNNM	Feedforward Neural Network Model
FPG	Favourable Pressure Gradient
LES	Large Eddy Simulation
MLP	Multi-Layer Perceptron
MtO	Many-to-One
OtO	One-to-One
RANS	Reynolds-Averaged Navier–Stokes eq.
ReLU	Rectified Linear Unit
SELU	Scaled Exponential Linear Unit
WPF	Wall Pressure Fluctuations
WPS	Wall Pressure Spectrum/Spectra
ZPG	Zero Pressure Gradient

Physics Symbols:

B	[1]	logarithmic law of the wall coefficient
c_0	$[\mathrm{m\,s^{-1}}]$	speed of sound
C_f	[1]	coefficient of friction
$\mathrm{d}p/\mathrm{d}x$	$[\mathrm{Pa\,m^{-1}}]$	static pressure gradient
f	[Hz]	frequency
H	[1]	shape factor, δ^*/θ
$\mathbf{k}$	$[\mathrm{rad\,m^{-1}}]$	wavenumber vector
M	[1]	Mach number
$p, p', \langle p \rangle$	[Pa]	pressure, pressure fluctuation, mean pressure

$P(\mathbf{k}, \omega)$	[s]	wavenumber frequency spectrum
p_x^+	[1]	dimensionless pressure, $(\nu/\varrho u_\tau^3)\,(\mathrm{d}p/\mathrm{d}x)$
q	[Pa]	dynamic pressure
R	[J kg^{-1} K^{-1}]	specific gas constant
Re	[1]	Reynolds number
R_T	[1]	ratio of timescales
t	[s]	time
T	[K]	temperature
T_{ref}	[K]	referential temperature
$u, u', \langle u \rangle$	[m s^{-1}]	velocity, velocity fluctuation, mean velocity
u^+	[1]	dimensionless mean velocity, u/u_τ
U_e	[m s^{-1}]	external velocity
u_{ps}	[m s^{-1}]	pseudo-velocity
U_{ref}	[m s^{-1}]	referential velocity
u_τ	[m s^{-1}]	friction velocity, $\sqrt{\tau_w/\varrho}$
$\mathcal{W}$	[1]	wake function
x_c	[1]	relative chord position
x,y,z	[m]	coordinates
y^+	[1]	dimensionless distance from the wall, yu_τ/ν
y_c^+	[1]	dimensionless viscous sublayer thickness
β_c	[1]	Clauser's parameter, $(\theta/\tau_w)\,(\mathrm{d}p/\mathrm{d}x)$
γ	[1]	heat capacity ratio
δ	[m]	boundary layer thickness
δ^*	[m]	displacement thickness
Δ	[1]	Zagarola and Smits's parameter, δ/δ^*
η	[1]	dimensionless coordinate, y/δ
θ	[m]	momentum thickness
κ	[1]	von Kármán's constant
μ	[Pa s]	dynamic viscosity
ν	[m^2 s^{-1}]	kinematic viscosity
$\boldsymbol{\xi}$	[m]	spatial separation vector
Π	[1]	Coles's wake parameter
ϱ	[kg m^{-3}]	density
τ_w	[Pa]	wall shear stress
Φ_{pp}	[Pa^2 s]	wall pressure spectrum (power spectral density of WPF)
ω	[rad s^{-1}]	angular frequency
$\boldsymbol{\omega}_{vor}$	[Hz]	vorticity

Machine Learning Symbols:

b	bias
f	activation function
J	cost function

k_{width}	kernel width
L, l	layer
m	n. of examples
n	n. of input connections
n_{CNNM}	n. of train. param. in a CNNM
n_{FNN}	n. of train. param. in an FNN
n_k	n. of kernels
n_L	n. of layers
n_p	n. of input parameters (features)
n_u	n. of units
w	weight
x	input
y	output
$\hat{y}$	prediction

1 Introduction

Wall Pressure Fluctuations (WPF) beneath a turbulent boundary layer have been studied primarily for two reasons: to investigate the structure and physical mechanisms of the boundary layer turbulence which generates the pressure field and to find relevant properties of the field which occurs in a large number of practical engineering problems [1]. Such practical engineering problems can be following: 1) fatigue loading on structures (e.g. [2]); 2) generation of acoustic radiation into the flow as a result of fluid interaction with flexible structure (e.g. [3]); 3) generation of acoustic radiation into the flow as a result of turbulent flow over an extended rigid surface (aero-acoustics). The last problem is connected to this work.

Wall pressure fluctuations can be obtained by high fidelity simulations, e.g. Direct Numerical Simulations (DNS) or Large Eddy Simulations (LES). Also, they can be obtained by an experiment in a wind tunnel. Both these approaches are very time-consuming. Thus, the modelling of Wall Pressure Spectra (WPS) will be used.

One way to model wall pressure spectra is to use semi-empirical models. These models take local boundary layer quantities to scale wall pressure spectra. However, finding the right scaling is not a trivial task. The modern semi-empirical models are based on the Chase-Howe model [4] and the Goody model [5]. The later models account for adverse, and zero pressure gradient flows and also use broader datasets – e.g. Rozenberg *et al.* [6]. These models are physics-based.

An alternative is to use a data-driven model. Such a data-driven model can be built with a neural network. Previous work on the neural network modelling of wall pressure spectra was done by Jan Van den Berghe [7], where he proved that neural networks could be used to predict the wall pressure spectra. Both the semi-empirical and the neural network model used local quantities to predict the wall pressure spectra.

Since the flow is influenced by its history, the wall pressure spectra could also be influenced by the flow history. Therefore the proposed question for this work is: *Can we use the values upstream the point where we would like to predict the wall pressure spectra, and is it going to improve the accuracy of such prediction compared to the existing models?* To answer this question, a feedforward neural network and a convolution neural network will be used to build a wall pressure spectral model. Four datasets [8, 9, 10, 11] will be used to build the models.

In the following chapter, the basics of the wall pressure fluctuations, the wall pressure spectra, and machine learning will be described. In chapter 3, the database of boundary layer parameters is prepared. In chapter 4, the results of the neural network models together with a theoretical investigation of the history effects is presented. Chapter 5 concludes the work and proposes new work.

2 Literature Review and Methods

2.1 Wall Pressure Fluctuations

Pressure fluctuations in an incompressible flow are governed by the Poisson equation with two source terms [12, 13]

$$\frac{1}{\varrho}\nabla^2 p' = -2\frac{\partial\langle u_i\rangle}{\partial x_j}\frac{\partial u_j'}{\partial x_i} - \frac{\partial^2}{\partial x_i \partial x_j}\left(u_i'u_j' - \langle u_i'u_j'\rangle\right), \tag{2.1}$$

where p' is the fluctuating pressure, ϱ is the constant fluid density, $\langle u_i\rangle$ and u_i' are the mean and fluctuating components of the flow velocity. The brackets $\langle\ \rangle$ denote ensemble average. The first source term characterize so-called *rapid pressure fluctuations* and the second term characterize *slow pressure fluctuations* [13]. The rapid pressure term, also called turbulence–mean flow interaction term [14], responds immediately to a change in the mean velocity gradients and, in some cases, has a dominant effect [13]. The slow term, also called turbulence–turbulence interaction term [14], is not directly driven by the mean velocity gradient and the interaction between different components is non-linear.

To find the integral solution of Eq. (2.1), one must obtain the velocity in the flow domain at every point. Moreover, there is no single scaling for the boundary layer itself. Also, the velocity fluctuations are convected at different speeds according to the mean velocity profile in the boundary layer. Therefore, a universal scaling with local length and velocity scales might be cumbersome [1].

See Appendix A for the derivation of Eq. (2.1).

2.1.1 Wall Pressure Spectra

Considering wall pressure fluctuations as $p' = p'(x, z, t)$ and flow in the x direction over (x, z) plane, space-time correlation can be defined as [1]

$$R\left(\xi_x, \xi_z, \tau\right) = \langle p'\left(x, z, t\right) p'\left(x + \xi_x, z + \xi_z, t + \tau\right)\rangle, \tag{2.2}$$

where $\xi = (\xi_x, 0, \xi_z)$ is a two dimensional spatial separation vector.

Reducing the correlation function leads to intermediate correlation functions like $R\left(\xi_x, 0, \tau\right)$, $R\left(0, \xi_z, \tau\right)$ in the space-time domain and $R\left(0, 0, \tau\right)$ in the time domain. The Fourier transform of $R\left(0, 0, \tau\right)$ in the time domain gives the single-point *wall pressure spectrum* (Wiener-Khinchin theorem) [1]

$$\Phi_{pp}\left(\omega\right) = \frac{1}{2\pi}\int_{-\infty}^{\infty} R\left(0, 0, \tau\right) e^{-i\omega\tau}\, d\tau. \tag{2.3}$$

Scaling

Self-similarity is a core principle in fluid mechanics. Proper scaling of experimental data to collapse it points onto one curve or surface can result in a better understanding of the investigated phenomenon and find a universal approximation of the phenomenon that does not change for a wide variety of conditions.

One characteristic of turbulent boundary layers is a large variety of length, velocity and pressure scales. Moreover, velocity in all parts of the turbulent boundary layer influences the wall pressure fluctuations. Therefore, there is no single universal scaling that would collapse the wall pressure spectra onto a single curve at all frequencies [1, 5].

Traditionally, the wall pressure spectrum has been determined from empirical curves plotted with dimensionless (scaled) spectral density and dimensionless (scaled) frequency [15]. Proper scaling has been discussed in the works of several authors, e.g. Bull [1], Farabee and Casarella [16], Smol'yakov [17] or Goody [5]. The result of this discussion is to use different scales for different frequency regions to collapse the data (Fig. 2.1). The resultant four regions are summed up in the Tab. 2.1. In this table, ω is used for the frequency, δ is the boundary layer thickness, δ^* is the displacement thickness, τ_w is the wall shear stress, $u_\tau = \sqrt{\tau_w/\varrho}$ is the friction velocity, ν is the kinematic viscosity and $q = 1/2\varrho U_e^2$ is the local dynamic pressure, where U_e is the free stream velocity. From the low-frequency region to the high-frequency region, the scaling varies from the outer-layer variable scaling to the inner-layer variable scaling. For the universal (overlap) region, both of these scalings can be used.

Table 2.1: Four frequency regions – scaling and properties [15].

Region	Scaled frequency range	Pressure scale	Time scale	Properties
Low-frequency	$\frac{\omega\delta}{u_\tau} \leq 5$	q or τ_w	δ^*/U_e	Varies with ω^2
Mid-frequency	$5 \leq \frac{\omega\delta}{u_\tau} \leq 100$	τ_w	δ/u_τ	Peak at $\frac{\omega\delta}{u_\tau} \approx 50$
Universal (overlap)	$100 \leq \frac{\omega\delta}{u_\tau} \leq 0.3\left(\frac{u_\tau\delta}{\nu}\right)$	τ_w	δ/u_τ	Varies with $\omega^{-0.7}$ to $\omega^{-1.1}$
High-frequency	$0.3 \leq \frac{\omega\nu}{u_\tau^2}$	τ_w	ν/u_τ^2	Varies with ω^{-1} to ω^{-5}

2.1.2 Modelling of Wall Pressure Spectra

To obtain the wall pressure spectra, one has to either perform an experiment or a high fidelity simulation (DNS/LES).

Experimental works were performed mainly in acoustic wind tunnel facilities. In one group of the experiments (e.g. [18, 16]), the authors measured wall pressure fluctuations on a flat wall with flush mounted pinhole microphones. In the second group (e.g. [19, 20]), the authors conducted experiments on the flow around an airfoil, where they measured wall pressure fluctuations on the airfoil and also the emitted far field noise.

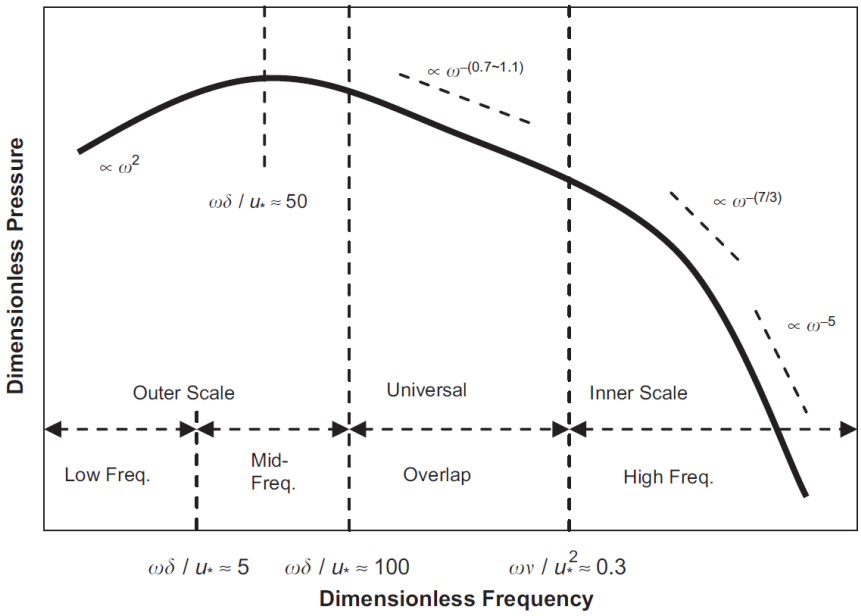


Figure 2.1: General characteristics of turbulent boundary layer wall pressure spectra. Different notation is used here, u_* is u_τ. Picture taken from [15].

Numerical computations were performed using large eddy simulation, direct numerical simulation and also a Lattice-Boltzmann method (e.g. [21, 22, 10]). In all of these works, the wall pressure fluctuations and the far field noise were investigated. Mostly, the numerical results were also compared to experimental results to assure the correctness of the computation. Compared to a purely experimental approach, one of the advantages of numerical computation is that it produces a broader set of information about the turbulent flow field and its statistics.

Both of these approaches are expensive. Another method is to model the wall pressure spectra using either a semi-empirical model or a model based on the solution of the Poisson equation (Eq. (2.1)). With the latter approach, two kinds of models can be obtained: 1) Model that outputs a spectral solution (wavenumber frequency pressure spectrum); 2) Model that outputs space-time solution (pressure fluctuation correlations) [23]. One example of a former model is from Kraichnan [24], where he expressed the wavenumber frequency spectrum as a double integral over the wall-normal coordinates [23]. Panton and Linebarger [25] followed the Kraichnan approach with more realistic flow conditions, which resulted in a five-dimensional integration. For such integration, they used Monte Carlo method. More recent work from Slama *et al.* [23] focused on computing the pressure correlations using a new model called Kriging-based elliptic extended anisotropic model (KEEAM), where they used DNS and RANS data as inputs. TNO-Blake family of models also exists. These models use an approximative solution of the Poisson equation (Eq. (2.1)) to predict the wall pressure spectra (e.g. [26]).

The semi-empirical models are based on an algebraic expression that uses scaling of the wall pressure spectra to collapse the spectra' curves. For such scaling, the

models use local quantities. The first attempts in the research on semi-empirical models date back some 60 years. The earliest model published might be the one from Maestrello [3]. Unfortunately, these early models are usually single-scale and covering only some narrow frequency range [15]. This work is entirely devoted to the semi-empirical modelling approach, and some of the most recent models will be described in detail in the following sections.

Chase-Howe Model

Howe [4] based his model on the Chase's [27] expression for the wavenumber frequency spectrum $P(\mathbf{k}, \omega)$. He suggested the following expression for the single point wall pressure spectrum

$$\frac{\Phi_{pp}(\omega) U_e}{\tau_w^2 \delta^*} = \frac{2 (\omega \delta^* / U_e)^2}{\left[(\omega \delta^* / U_e)^2 + 0.0144\right]^{3/2}}, \tag{2.4}$$

which is the so-called *Chase-Howe model*. The 2 in the numerator is not in the original model in the work of Howe [4], since in the original work he described a double-sided spectrum. As stated by Goody [5], the model does not account the ω^{-5} decay at high frequencies (Tab. 2.1).

Goody Model

Goody [5] proposed a model based on the Chase-Howe model. He valued the attractive functional form of the Chase-Howe model and he also saw it as a compromise between pure curve fit and a complex expression for the wavenumber frequency spectrum $P(\mathbf{k}, \omega)$. In the derivation of the Goody model, following assumptions about the model were made: 1) A term was added to the denominator so that the spectrum varies as ω^{-5} when $\omega \to \infty$; 2) Exponents were modified to better agree with the spectrum at middle frequencies; 3) Multiplicative constant was added to raise the spectral levels at all frequencies; 4) The Reynolds number trends were reflected. The model was fitted on experimental, 2D, zero pressure gradient boundary layer data. The final equation of the model is

$$\frac{\Phi_{pp}(\omega) U_e}{\tau_w^2 \delta} = \frac{C_2 (\omega \delta / U_e)^2}{\left[(\omega \delta / U_e)^{0.75} + C_1\right]^{3.7} + \left[C_3 (\omega \delta / U_e)\right]^7}, \tag{2.5}$$

where $C_1 = 0.5$, $C_2 = 3.0$ and $C_3 = 1.1 R_T^{-0.57}$. The R_T is the *ratio of timescales*, which characterizes the Reynolds number effect on the spectrum.

The ratio of timescales R_T is defined as

$$R_T = \frac{\delta / U_e}{\nu / u_\tau^2}. \tag{2.6}$$

Rozenberg Model

Rozenberg *et al.* [6] used the Goody model and modified it to account both zero pressure gradient and adverse pressure gradient data. To achieve this, they introduced new parameters and new data into the model. The first parameter is Coles's wake parameter Π, the second parameter is Zagarola and Smits's [28] parameter $\Delta = \delta/\delta^*$ and the third parameter is Clauser's [29] equilibrium parameter $\beta_c = (\theta/\tau_w)(\mathrm{d}p/\mathrm{d}x)$. They also used δ^* instead of δ in the time scale as the displacement thickness δ^* is more accurate. The maximum shear stress $\tau_{max} = \max\left[\mu\left(\mathrm{d}u/\mathrm{d}y\right)\right]$ is used instead of the wall shear stress τ_w. The final form of the Rozenberg model is

$$\frac{\Phi_{pp}\left(\omega\right)U_e}{\tau_{max}^2\delta^*} = \frac{\left[2.82\Delta^2\left(6.13\Delta^{-0.75}+F_1\right)^{A_1}\right]\left[4.2\left(\Pi/\Delta\right)+1\right]\left(\omega\delta^*/U_e\right)^2}{\left[4.76\left(\omega\delta^*/U_e\right)^{0.75}+F_1\right]^{A_1}+\left[C_3'\left(\omega\delta^*/U_e\right)\right]^{A_2}}, \tag{2.7}$$

where $A_1 = 3.7 + 1.5\beta_c$, $A_2 = \min\left(3; 19/\sqrt{R_T}\right) + 7$, $C_3' = 8.8R_T^{-0.57}$ and $F_1 = 4.76\left(1.4/\Delta\right)^{0.75}\left(0.375A_1 - 1\right)$.

The Rozenberg model should provide the same results as the Goody model for zero pressure gradient flows, but according to Lee's work [30] it is not true. Moreover, the Rozenberg model underpredicts the spectra at high frequencies compared to the Goody model in the case of zero pressure gradient flows. However, there is a possibility to correct the model using $A_2 = 7$ to fit the Goody model in the case of zero pressure gradient flows.

In this work, an implementation of the Rozenberg model with τ_w instead of τ_{max} is used.

Kamruzzaman Model

Kamruzzaman *et al.* [31] compared the Chase-Howe, Goody and Rozenberg model with experimental data from two NACA 0012 airfoil test cases. They noted that Chase-Howe and Goody model hugely underestimate the experimental data. The Rozenberg model, compared to the experimental data, predicted the peak of the spectra reasonably well for lower Reynolds number cases, but had an offset in the low and high frequencies. With increasing Reynolds number, the difference was even bigger. Thus, Kamruzzaman *et al.* wanted to overcome these limitations of the Rozenberg model. Their proposed model was fitted to a broaden set of non-zero and zero pressure gradient data (airfoil and flat plate flows). The final form of this model is

$$\frac{\Phi_{pp}\left(\omega\right)U_e}{\tau_w^2\delta^*} = \frac{B_2\left(\omega\delta^*/U_e\right)^2}{\left[\left(\omega\delta^*/U_e\right)^p+B_1\right]+\left[B_3\left(\omega\delta^*/U_e\right)\right]^7}, \tag{2.8}$$

where $B_1 = 0.27$, $B_2 = 0.45[1.75\left(\Pi^2\beta_c^2\right)^m + 15]$, $m = 0.5\left(H/1.31\right)^{0.3}$, $B_3 = \left(1.15R_T\right)^{-r}$, $r = 2/7$, $p = 1.637$ and $q = 2.47$. The parameter H is the shape factor $H = \delta^*/\theta$. R_T (Eq. (2.6)) in this model is defined with δ^* instead of δ.

Hu Model

Hu and Herr [32] proposed a new model from the Goody model. They used experimental non-zero and zero pressure gradient data from flows over a flat plate. The main difference between the Hu model and the other models is, firstly, instead of $U_e/\tau_w^2\delta$ (or δ^*), $u_\tau/q^2\theta$ is used. Secondly, instead of the ω^2 increase at low frequencies, only ω is used. And thirdly, instead of Clauser's parameter β_c, the shape factor H is used to account for the influence of a zero and non-zero pressure gradient on the mean velocity profile shape. The authors noted that the local pressure gradient directly impacts β_c. Therefore β_c does not account for the history of the flow as good as H. The final form of the Hu model is

$$\frac{\Phi_{pp}(\omega)\,u_\tau}{q^2\theta} = \frac{a\,(\omega\theta/U_e)^{1.0}}{\left[(\omega\theta/U_e)^{1.5h^{1.6}} + d\right]^{1.13/h^{0.6}} + \left[f\,(\omega\theta/U_e)\right]^{6.0}}, \tag{2.9}$$

where $a = (81.004d + 2.154)\times 10^{-7}$, $d = 10^{-5.8\times 10^{-5} Re_\theta H - 0.35}$, $h = 1.169\ln(H) + 0.642$, $f = 7.645 Re_\tau^{-0.411}$, $Re_\theta = U_e\theta/\nu$ and $Re_\tau = u_\tau\delta/\nu$.

Lee Model

Lee [30] wanted to improve the previous models for a better prediction in all of the observed cases. He proposed new parameters to firstly improve the transition from the overlap frequency region to the high frequency region and to improve the roll-off at high frequencies. Secondly, he introduced a correction of the amplitude at low and middle frequencies for zero and low pressure gradient flows. And thirdly, he introduced a correction of the spectrum magnitude for high β_c cases. The final form of his model is

$$\frac{\Phi_{pp}(\omega)\,U_e}{\tau_w^2\delta^*} = \frac{\max\left(a; (0.25\beta_c - 0.52)\,a\right)(\omega\delta^*/U_e)^2}{\left[4.76\,(\omega\delta^*/U_e)^{0.75} + d^*\right]^e + \left[8.8 R_T^{-0.57}\,(\omega\delta^*/U_e)\right]^{h^*}}, \tag{2.10}$$

where $a = \left[2.82\Delta^2\,(6.13\Delta^{-0.75} + d)^e\right]\left[4.2\,(\Pi/\Delta) + 1\right]$, $d = 4.76\,(1.4/\Delta)^{0.75}\,[0.375e - 1]$, $e = 3.7 + 1.5\beta_c$, $d^* = \max(1.0; 1.5d)$ if $(\beta_c < 0.5)$ and $h^* = \min(3; (0.139 + 3.1043\beta_c)) + 7$.

All the previous models have a common functional form for the wall pressure spectra [30]

$$\Phi_{pp}(\omega)\,SS = \frac{a\,(\omega FS)^b}{\left[i\,(\omega FS)^c + d\right]^e + \left[(f R_T^g)\,(\omega FS)\right]^h}. \tag{2.11}$$

In this *general form*, a to i are parameters, R_T is the ratio of timescales (Eq. (2.6)), SS is the spectral scaling and FS is the frequency scaling. Lee [30] summed up in his article the already used forms for the parameters a to i and for the scaling SS and FS. Summary of the parameters and the scalings extended with Lee's model is in the Tab. 2.2. The simplest case of the parameters occurs when the parameters are constants. In fact, that is the case of the Goody model.

Table 2.2: The overview of the coefficients for the general semi-empirical wall pressure spectra model.

	a	b	c	d
Goody	3.0	2.0	0.75	0.5
Rozenberg	$\left[2.82\Delta^2\left(6.13\Delta^{-0.75}+d\right)^e\right]\left[4.2\left(\Pi/\Delta\right)+1\right]$	2.0	0.75	$4.76\left(1.4/\Delta\right)^{0.75}\left(0.375e-1\right)$
Kamruzzaman	$0.45\left[1.75\left(\Pi^2\beta_c^m\right)^2+15\right]$; $m=0.5\left(H/1.31\right)^{0.3}$	2.0	1.637	0.27
Hu	$\left(81.004d+2.154\right)10^{-7}$	1.0	$1.5h^{1.6}$	$10^{-5.8\times10^{-5}Re_\theta H-0.35}$
Lee	$\max\left(a;\left(0.25\beta_c-0.52\right)a\right)$; $a=\left[2.82\Delta^2\left(6.13\Delta^{-0.75}+d\right)^e\right]\left[4.2\left(\Pi/\Delta\right)+1\right]$	2.0	0.75	$4.76\left(1.4/\Delta\right)^{0.75}\left(0.375e-1\right)$, or $\max\left(1.0;1.5d\right)$ if $\left(\beta_c<0.5\right)$

	e	f	g	h
Goody	3.7	1.1	-0.570	7.0
Rozenberg	$3.7+1.5\beta_c$	8.8	-0.570	$\min\left(3;19/\sqrt{R_T}\right)+7$
Kamruzzaman	2.47	$1.15^{-2/7}$	$-2/7$	7.0
Hu	$1.13/h^{1.6}$	7.645	-0.411	6.0
Lee	$3.7+1.5\beta_c$	8.8	-0.570	$\min\left(3;\left(0.139+3.1043\beta_c\right)\right)+7$

	i	SS	FS
Goody	1.0	$U_e/\tau_w^2\delta$	δ/U_e
Rozenberg	4.76	$U_e/\tau_w^2\delta^*$	δ^*/U_e
Kamruzzaman	1.0	$U_e/\tau_w^2\delta^*$	δ^*/U_e
Hu	1.0	$u_\tau/Q^2\theta$	θ/U_e
Lee	4.76	$U_e/\tau_w^2\delta^*$	δ^*/U_e

To sum up the applicability of the previous models, the Goody and the Hu model work well for zero pressure gradient flows [30]. The Goody model does not work for adverse pressure gradients. The Rozenberg and the Kamruzzaman model work well for airfoil flows. However, they are not that accurate under some conditions. The Hu model is the only one that can match Hu and Herr's test dataset [32]. The Lee model improves the Rozenberg model to match a broader set of flow cases. The final applicability range of the Lee model should be flows with $1.4 \times 10^3 \leq Re_\theta \leq 2.34 \times 10^4$ for zero pressure gradients and with $0.5 \times 10^3 \leq Re_\theta \leq 1.65 \times 10^4$ for adverse pressure gradients. Still, there is no semi-empirical model which could fit all the datasets and all the flow conditions with some acceptable error.

The inputs into every semi-empirical model are always the frequency and combination of boundary layer parameters derived from the mean flow. The output is always the wall pressure spectrum at one point. Finding the right combination of the boundary layer parameters is a cognitively challenging task. Therefore, if there was an approach, or a method, which could find the combinations for us, we could quickly find different combinations in a reasonable time frame. Moreover, if there was a method that would search the space of the potential combinations and somehow choose the best one, it would significantly improve the search for the best semi-empirical model. Such a method might be neural network modelling or, generally, data-driven modelling.

Data-Driven Modelling

Data-driven modelling and mainly neural network modelling of wall pressure spectra is not a total novelty. Previous work on this topic at The von Karman Institute for Fluid Dynamics was done by Jan Van den Berghe [7]. He asked the question if neural networks are an interesting tool for building wall pressure spectra models. To answer such a question, he investigated the behaviour of neural networks. He proposed a global structure for his models, to use frequency ω and boundary layer parameters U_e, C_f and β_c as inputs and wall pressure spectra Φ_{pp} at one point as output. He applied this structure to two approaches: 1) Using the inputs directly – so-called *brute force approach*; 2) Connecting the neural network to the general wall pressure spectra model (Eq. (2.11)), thus using the output as the coefficients into the general expression and also restricting the pure neural network model to some physical boundaries – so-called *parametric approach*. He concluded that the neural network can predict the wall pressure spectra and that it will generalize well between the extremities of the database. Also, training the model on a larger database, especially using the parametric approach, will result in a more accurate prediction. He finally noted that it is still not clear how the network predicts with such accuracy.

Other works on the same topic are not known to the author up to this date.

One typical attribute of a boundary layer, or a flow generally, is that the flow at one point is heavily influenced by what happened upstream this point. It might be called a *history effect*. Since the wall pressure spectra are predicted with semi-empirical models using the quantities from the mean flow, there might be the same history effect. This results in the already asked question: *Can we use the values*

upstream the point where we would like to predict the wall pressure spectra, and is it going to improve the accuracy of such prediction compared to the existing models?

Besides the main question, a classical feedforward neural network with different inputs will be trained on a broader dataset than in the previous work. This approach, where the parameters at one point are used to predict the spectra at the same point, will be called a *One-to-One model*.

To answer the main question of this work, firstly, machine learning and neural networks have to be introduced and also the possibilities of how to use a neural network with many inputs and one output (a *Many-to-One model*) have to be investigated.

2.2 Machine Learning

This section describes the basics of machine learning and some relevant techniques of supervised learning.

Machine learning is a part of a bigger concept called *artificial intelligence (AI)*. Artificial intelligence is a general term that encapsulates all computer science techniques that would be similar to some "intelligent" behaviour or a process.

Machine learning describes the process where the system extracts patterns from a raw dataset by itself and then acquire knowledge from these patterns [33]. Machine learning can be divided into supervised, unsupervised and reinforcement learning. Only the first one will be used in this work. Supervised learning introduces the inputs and also the outputs (or labels) to the particular algorithm. On the contrary, unsupervised learning introduces only the inputs to the algorithm and lets the algorithm decide how the inputs should be adjusted.

In the following lines, the *feedforward neural network* (*FNN*) and the *convolution neural network* (*CNN*), as supervised learning techniques, are mentioned.

2.2.1 Feedforward Neural Network (FNN)

Artificial neural networks (ANN) (or simply *neural networks*), in general, were mathematically firstly described in the work of McCulloch and Pitts [34]. They proposed a computational model of the neurons and formulated how the neurons are working together. From this time, the ANN has gone through several periods of ups and downs. In the current time, the *deep learning* is one of the reasons why we are witnessing the renaissance of the ANN once again. Deep learning allows a model to learn a very abstract representation of data [35]. Thus, it is nowadays used in many applications such as image recognition, speech recognition, or large datasets analysis.

The *perceptron* (Fig. 2.2) is a basic structure of an ANN. The structure is similar to the actual neuron with its cell body, axon and dendrites. However, the inner processes are not that similar. The perceptron receives the signals from the inputs and multiply these inputs with their respective *weights*. All of these multiplications are summed up together with a *bias* (so-called *excitation*) and input into an *activation function*. This process can be described by an equation

$$y = f\left(\sum_{i=1}^{n} x_i w_i + b\right), \tag{2.12}$$

where f denotes the activation function, x_i are the inputs, w_i are the weights, b is the bias and n is the number of input connections. The weights and biases are called *trainable parameters*. What the training means will be described later.

The "original" perceptron [34] used a step function as the activation function

$$f(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0. \end{cases} \tag{2.13}$$

The step function is not going to be used in this work, since it does not provide the required features for this work.

The perceptrons, or so-called *units*, can be stacked in layers and also in rows and thus creating networks where the units are connected. In the case of a *feedforward neural network* (also called *multi-layer perceptrons* (*MLP*))[1], the connections are only between each layer and not within the layer. Moreover, there are no feedback connections to any unit. In graph theory, this is a special case of a *directed acyclic graph* [33].

Hornik [36] stated that a multilayer feedforward neural network could approximate any well-behaved continuous function if sufficiently many hidden units are available (*universal approximation theorem*). For this statement, he assumed the activation function to be continuous, bounded and non-constant. The statement thus makes the multilayer feedforward neural network a very promising architecture.

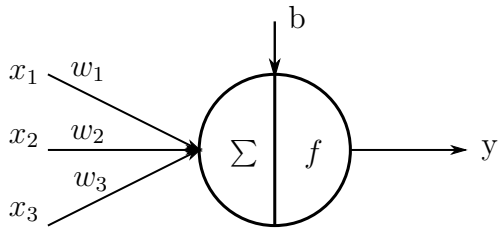


Figure 2.2: Perceptron.

An example of an FNN is the Fig. 2.3. The network consists of *input*, *output* and *hidden* layers. The circles denote the same units as in the previous lines (Fig. 2.2). However, the input layer is an exception because these circles are just used for the input values, and no real computation is done here. The example network, therefore, consists of 3 inputs, 5 units in the first hidden layer, 3 units in the second hidden layer and 1 output unit. All of those units are connected to the layer before and after the layer, where the units are. The choice of the number of inputs, units, hidden layers and outputs is purely optional and depends on the application.

The computations in the neural network can be divided into two processes: 1) *forward-propagation* and 2) *back-propagation*.

Forward-Propagation

Forward-propagation is a deterministic process, which provides the whole computation from the input layer to the output layer. It can also be called a *prediction*. The process for one layer can be written in a matrix form. Thus, e.g. for the second layer in the neural network in the Fig. 2.3 the relation is

$$\begin{aligned} \mathbf{z}^{(2)} &= \mathbf{W}^{(1)}\mathbf{x}^{(1)} + \mathbf{b}^{(1)}, \\ \mathbf{a}^{(2)} &= f\left(\mathbf{z}^{(2)}\right), \end{aligned} \tag{2.14}$$

where $\mathbf{W}$, $\mathbf{x}$ and $\mathbf{b}$ are the same weights, inputs and biases as in the Eq. (2.12), but for the whole layer. The vector $\mathbf{z}$ is the input into the activation function, and $\mathbf{a}$ is

[1] A feedforward neural network is a special case of an artificial neural network, but those terms are sometimes interchanged.

the output from the whole layer. The superscript describes the layer to which the variable belongs, which means that the weights and biases belong to the first layer and the output belongs to the second layer. The advantage of the matrix form is that it is possible to use various numerical techniques to solve the matrix equation. Putting all these computations for each layer together, one can finally obtain the output (predicted) value from the neural network.

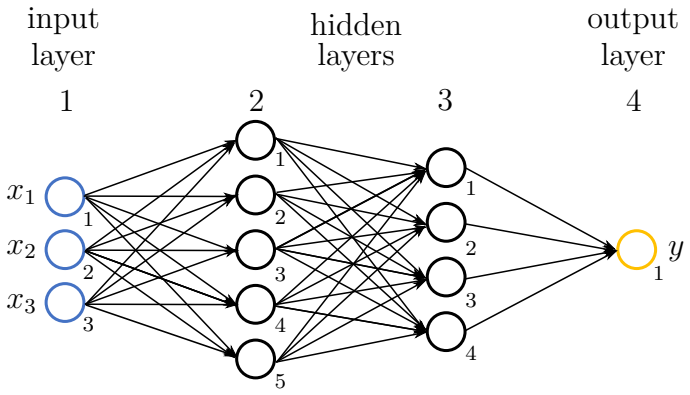


Figure 2.3: The example of a feedforward neural network with one input, two hidden and one output layer.

Back-Propagation

In the computation of forward-propagation, the weights w and biases b were used. This requires the knowledge of these values. The weights and biases are initialized to some value before the first forward pass. Usually, the initialization is random. This makes the process of obtaining and improving the weights and biases a non-deterministic process.

The weights and biases need to be optimized to improve the future prediction of the model. The process of calculation of the new weights and biases is called back-propagation.

Firstly, the computed output of the whole neural network needs to be compared with the desired values. A *cost function*[2] is used for such comparison. The output of the cost function is some error between the output (predicted) values and the desired values. The choice of the cost function depends on the task for which the neural network is used. In the case of a regression task, the cost function is usually the *mean squared error*

$$J = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 , \tag{2.15}$$

where J denotes the cost function, $\hat{y}$ is the output (or prediction), y is the desired value, and m is the number of examples. Sometimes, the mean absolute error is used instead as the cost function. The number of examples for computing the cost function can vary, and a set of such examples is called a *batch*. A *batch size* is one of the parameters that can be set for the neural network.

[2] A cost function is sometimes called a *loss function* or an *error function*.

After obtaining the loss with the cost function, the main part of back-propagation arises. The goal is to compute the gradients, which are then used in an optimization algorithm. In other words, it is necessary to compute the change of the cost function according to some particular weight or bias. The optimization algorithm then improves the weights and biases to better predict the desired values. An example of such an optimization algorithm is the *gradient descent*

$$w_{i+1} = w_i - \alpha \frac{\partial J}{\partial w_i}, \tag{2.16}$$

which optimizes, in here, some particular weight. The parameter α is the *learning rate*, which defines how big the optimization step should be.

The main relation used for back-propagation is the chain rule

$$\frac{\partial a}{\partial d} = \frac{\partial a}{\partial b}\frac{\partial b}{\partial c}\frac{\partial c}{\partial d}, \tag{2.17}$$

where a-d are some arbitrary variables. With this chain rule, one can "go backwards" the neural network and calculate the gradients. Again, the gradient calculation can be illustrated by an example. The sought gradient can be in the second layer, in the first unit in the Fig. 2.4. It is possible to write, with the help of the chain rule,

$$\frac{\partial J}{\partial w_{11}^{(2)}} = \frac{\partial J}{\partial a_1^{(4)}}\frac{\partial a_1^{(4)}}{\partial z_1^{(4)}}\frac{\partial z_1^{(4)}}{\partial a_1^{(3)}}\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}}\frac{\partial z_1^{(3)}}{\partial w_{11}^{(2)}}. \tag{2.18}$$

The notation here, $w_{ji}^{(L)}$, is following. The index i is the number of the input unit, j is the number of the output unit and (L) denotes the layer. The red dotted line in the Fig. 2.4 illustrates the back-propagation path described by the Eq. (2.18).

With the previous process, it possible to calculate all the weights and biases in the neural network.

The combination of forward-propagation and back-propagation might be called *learning* or *training*. One pass of forward- and back-propagation improves the weights and biases and also calculates the loss (or the cost) with the cost function. As it was said, there is a possibility to specify the batch size. The batch size specifies the number of examples that are input into the network before the gradients are updated. A small batch size results in quick but oscillating learning. A large batch size results in smooth but slow learning. LeCun recommended [37] to use a batch size smaller than 32.

Once forward- and back-propagation is done on the input dataset exactly once, the process is called an *epoch*. The neural networks are usually trained with many epochs.

In the following sections, some parts of the neural networks (relatable to this work) will be described in more detail.

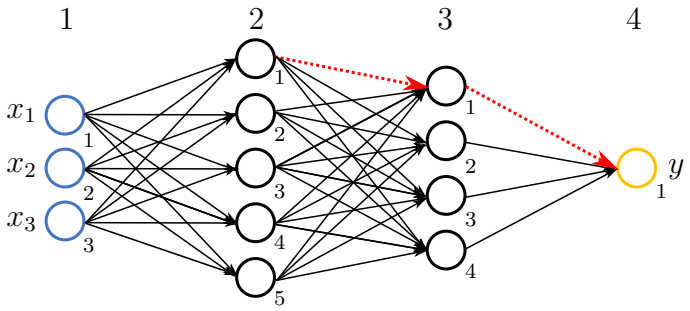


Figure 2.4: The example of a feedforward neural network. The red dotted line illustrates the back-propagation path described in this section.

Activation Functions

The activation function was already mentioned in the Eq. (2.12). Most hidden units are distinguished from each other by the choice of the activation function [33]. The activation function in most of the cases introduces the non-linearity into the output of the unit. Without the non-linearity, the neural network would put several linear combinations together and thus introduce another linear combination [37]. Also, the non-linear activation function is an important difference between machine learning and the biological neuron [38]. Brief overview [39] of the currently used activation functions and their derivatives is in the Fig. 2.5. From the top left corner, the following activation functions are mentioned:

- **Linear** – $f(x) = x$,
- **Logistic (sigmoid)** – $f(x) = \dfrac{1}{1 + e^{-x}}$,
- **Tanh** – $f(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$,
- **Rectified Linear Unit (ReLU)** [38] – $f(x) = \max(0; x)$,
- **Leaky ReLU** [40] – $f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.01x, & \text{if } x \leq 0 \end{cases}$,
- **Scaled Exponential Linear Unit (SELU)** [41] – $f(x) = \lambda \begin{cases} x, & \text{if } x > 0 \\ \alpha\left(e^{x} - 1\right), & \text{if } x \leq 0 \end{cases}$, $\alpha = 1.6733$, $\lambda = 1.0507$.

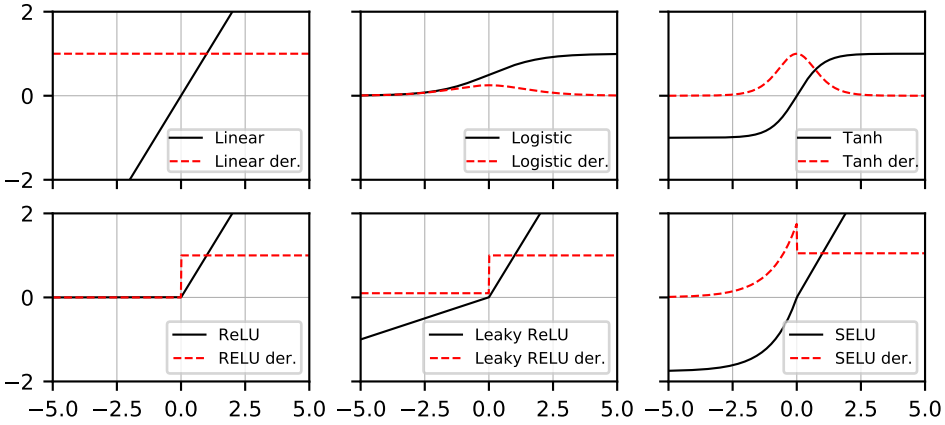


Figure 2.5: The activation functions and their derivatives. The Leaky ReLU is plotted with $0.1x$ when $x \leq 0$.

The linear activation function, by its definition, does not introduce non-linearity to the output. However, it might be used in the output layer for regression tasks. The logistic function and the hyperbolic tangent introduce the non-linearity to the output, but there might be problems with the *vanishing* or *exploding gradient*. These phenomena are the consequences of the shape of these functions. When the inputs to these functions are very large or very small, then the derivatives (e.g. $\partial a_1^{(3)}/\partial z_1^{(3)}$ in the Eq. (2.18)) will saturate extremely close to zero and the unit will "die". This "dead" unit will also affect the units in the previous layers since they also use the same gradient in back-propagation. The exploding gradient is mainly visible in the recurrent neural networks.

New activation functions were proposed to solve the vanishing/exploding gradient problem. The ReLU [38] partially solves the vanishing gradient problem because the function gradient is equal to one when the unit is active. However, when the unit is initialized as non-active, the gradient will always be zero during the optimization. However, the simplicity of the ReLU makes the computation of the function easier. The ReLU was improved by Mass *et al.* [40], who introduced the Leaky ReLU. This function has a non-zero gradient in the case of a non-active unit. Klambauer *et al.* [41] introduced the *self-normalizing neural network*, which is based on the SELU activation function. This function assures self-normalizing properties like variance stabilization, which avoids exploding and vanishing gradients [41]. The function also needs to be used together with a proper initialization.

Géron [37] recommends using activation functions in the following descending order: SELU, Leaky ReLU, ReLU, tanh, logistic.

Initialization

Before the weights and biases can be optimized, they first need to be initialized. The first most straightforward way would be to initialize the weights and biases with zeros. However, given units in one layer, this approach would make all the units

equal, and the back-propagation will also affect them equally. Therefore, the *random initialization* is used to break such symmetry in the layer [37].

Random initializers might have a uniform or a normal distribution [42]. Glorot and Bengio [43] proposed a way, how to maintain approximately equal activation variances and variances of back-propagation gradients as one moves up or down the network. Their initializer is called the *Glorot initializer* (or the *Xavier initializer*) and uses the previous and actual layer size as inputs. This initializer is also one of the tricks that led to the current success of deep learning [37].

Glorot and Bengio did not account for the ReLU in their derivation, and their initializer might cause problems in some deeper models. Therefore, there is another method proposed by He *et al.* [44] which works well together with the ReLU and its variants.

Klambauer *et al.* [41] proposed zero mean and unit variance for the initialization together with their SELU activation function to satisfy the self-normalizing neural network. The initializer also takes the previous layer size as an argument. This initializer was already proposed by LeCun [45]. Thus it is referenced as the *LeCun initializer*.

Optimization

Gradient descent, as an optimizer, was already mentioned in the Eq. (2.16). Several techniques can be used to improve default gradient descent [37]. The first one is adding a *momentum* to the optimization algorithm. The momentum takes the previous gradients into account and increases the actual gradient. The behaviour is similar to the physical momentum. The main advantage of this technique is that it accelerates optimization.

The second technique uses *Nesterov Accelerated Gradient*, which instead of computing the gradient of the cost function at the local position, it computes the gradient slightly ahead in the direction of the momentum. This technique, again, results in a faster reaching of the optimum and also reduces the oscillations.

The third technique used by *AdaGrad* and *RMSProp* [37] is to scale down the gradient vector along the steepest dimensions (the so-called adaptive gradient). In other words, this technique decays the learning rate faster for steep dimensions than for dimensions with gentler slopes [37].

In this work, the *Adam* [46] and the *Nadam* [47] optimizers are used. The Adam is a stochastic gradient-based optimizer (stochastic, as it randomly selects a subset of data for the computation). The name Adam stands for *adaptive moment estimation*. It combines the momentum technique and the RMSProp. The Nadam adds the Nesterov Accelerated Gradient technique to the Adam. Dozat [47] in his report compared different optimizers on some experiments, where one can see that the choice of an optimizer is highly data-dependent, and sometimes the Adam performs better than the Nadam and vice versa.

The optimization of the weights and biases might lead, for some optimizers more or less, to a local minimum. It happens in the case of a very irregular cost function [37]. However, it is assumed that the reached local minima are sufficiently near the

global minima. In the case of larger neural networks, the problem of poor local minima becomes gradually less important [48].

Train, Validation and Test Split

The key attribute of a neural network is the ability to generalize to new cases. Therefore, it is necessary to take out a subset of data to evaluate the ability to generalize. However, in practice, the whole default dataset is divided into three groups:

- **A training set** – This dataset is used for the training of the model.
- **A validation set** – This dataset is taken out prior to the training, and then the model is evaluated with this dataset. Also, the dataset is used to compare different models between each other.
- **A test set** – This dataset is taken out prior to the training and is used purely to evaluate the final model.

The main reason to use both a validation and a test set is that the validation set is used to compare of different models. Thus the validation dataset projects itself to the model, and the generalization verification would not be that strong.

To obtain the training, validation and test set, the whole default dataset is usually divided into 60 %, 20 % and 20 % sets (depending on the data amount). Also, each dataset should have a similar representation of the data as the default dataset.

Overfitting, Underfitting and Regularization

The neural network model might suffer from overfitting or underfitting. Overfitting occurs when the model is too complex, and thus the model fits the data perfectly. But the prediction of new values is bad. On the other hand, underfitting occurs when the model is very simple and the model does not represent the training data. It can be said that an overfitting model has a high variance and low bias, and an underfitting model has a low variance and high bias. An illustration of overfitting and underfitting is in the Fig. 2.6, where the ideal state is somewhere between. The ideal state is a compromise between a good representation of the training data and generalization.

Looking at the Fig. 2.7, the point where the fit is optimal can be observed as the minimum of the validation loss. To the left to this point, the model underfits and to the right, the model overfits.

Overfitting can be cured by simplifying the model, e.g., using less trainable parameters or using fewer features in the dataset. The cost function can include regularization in form of a "penalty" (e.g. ℓ_1 and ℓ_2 norm). Another possibility is to use a dropout layer or increase the amount of data.

To cure underfitting, it is necessary to increase model complexity, add more features or decrease regularization in the cost function.

Lastly, it is possible to stop the learning of the model when the validation loss starts to increase. Such a technique is called *early stopping*.

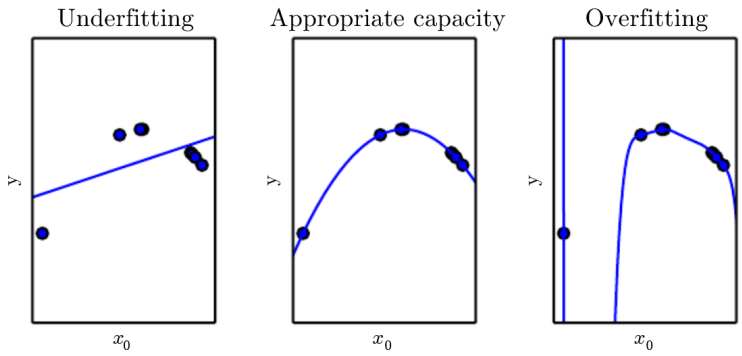


Figure 2.6: The example of a curve fit with different degrees of a polynomial. Picture taken from [33].

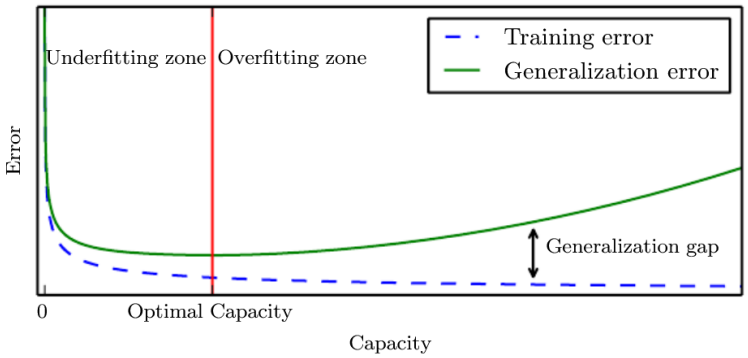


Figure 2.7: A training and validation loss (error) example. The capacity, in this case, might be interchanged with the number of epochs. Picture taken from [33].

2.2.2 Convolution Neural Network (CNN)

The proposed Many-to-One model in the subsection 2.1.2 is partially investigated with the *convolution neural network*. The CNN and the convolution layer have been used extensively in, e.g. image processing. The reason is that the CNN is inspired by the real processes in the visual cortex [37]. The work of Fukushima [49] was inspired by the visual cortex, and thus he formulated the current form of the CNN. Mainly 2D or 3D CNNs are used in image processing. However, only the 1D CNN will be mentioned in this work, as it is suitable for 1D signals.

The use of the 1D CNN was described, e.g. by Kiranyaz *et al.* [50], where they surveyed how the 1D CNN can be used in classification tasks. They concluded that the 1D CNNs are relatively easier to train than the 2D CNNs, and with such minimal computational complexity, the 1D CNNs still achieve state-of-the-art performance. Another example of the use of 1D CNNs is the work by Malek *et al.* [51]. They used

the 1D CNNs for regression tasks with application to spectroscopic signal regression. Both of the previous works emphasize the use of 1D CNNs as feature extractors.

The convolution layer is the basis of the CNN. The discrete form of the convolution is described in the Fig. 2.8. Here, the matrix $\mathbf{I}$ denotes the input matrix, where, e.g. the columns are timesteps or spatial positions and the rows are parameters for each timestep/position. This matrix is step-by-step element-wise multiplied with the *kernel* (or the *filter*) $\mathbf{K}$. Each step (also called a *stride*), those multiplications are summed up and stored in an output vector. The step length and kernel width are arbitrary.

In image processing, it is usual to create very deep models with many convolution layers. However, it is rare to use the convolution layers solely. *Pool layers* are used in many cases after those convolution layers in order to downsample the model [37]. The pool layers can also be used to fix the size of the output from the convolution layer. After the convolution layers (and pool layers), the feedforward neural network can be connected, as it helps to improve the abstraction and regression/classification ability of the model.

Similar rules and practices apply for training the convolution neural network as for the feedforward neural network. Thus it is not mentioned here again.

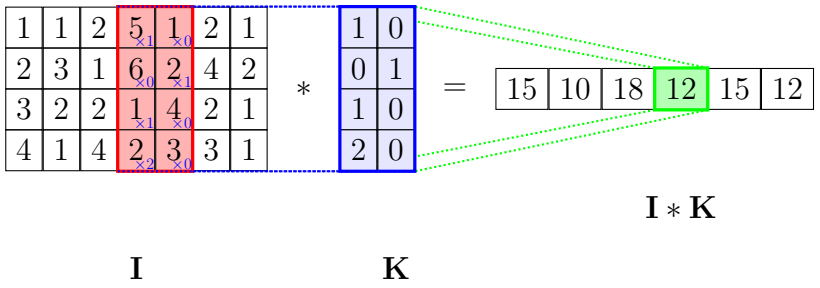


Figure 2.8: The illustration of convolution calculation.

Conclusion

Two neural network architectures were proposed and described in the previous sections. The first one is the FNN which is a great candidate for the One-to-One model. In that case, the boundary layer parameters can be used as inputs into the FNN, and the output can be the wall pressure spectrum for one frequency point.

The FNN is also a great candidate for the Many-to-One model. In that case, the model can use the boundary layer parameters from several boundary layer points next to each other. The model's output can be the wall pressure spectrum for the last point in the series of boundary layer points for one frequency point.

The CNN can also be used for the Many-to-One model. Here, the inputs and outputs can be the same as in the FNN Many-to-One model. The difference is that the CNN can use variable-length inputs. Connecting the CNN with the pool layer and the FNN results in a model that takes variable-length inputs, extracts the features via the CNN, subsamples the features to some fixed size via the pool layer and then performs the regression task via the FNN.

A challenging aspect in training the proposed models is the setup of the model's architecture, such as choosing the number of hidden layers, the number of hidden units, the activation function, the initialization or the number of kernels. In the machine learning field, these parameters are called *hyper-parameters*. The proposed models, in this work, will be trained with different hyper-parameter setups.

Nowadays, the industry-standard in machine learning is to prototype neural network models with the help of some machine learning library. In this work, mainly the Tensorflow [52] together with the Keras [53] will be used. The reason for the use of these libraries is that they are Python-based and open-source.

In the next chapter, the database preparation will be described together with the selection of boundary layer parameters and model inputs.

3 Database Preparation

In order to train neural network model, a database of relevant boundary layer parameters is needed. In this work, four different data sources were chosen:

- Salze *et al.* [8] – an experiment, in a wind tunnel test channel, on a flat plate
- Deuse and Sandberg [9] – LES, CD airfoil,
- Wu *et al.* [10] – DNS, CD airfoil,
- Christophe *et al.* [11] – LES, CD airfoil,

where all of them will be described in the following sections.

All datasets used here are already preprocessed from the raw signal outputs or data stored in a mesh. Therefore, only time-averaged and, in the case of airfoils, spanwise averaged data are available. Concretely, each dataset is divided into boundary layer points stored in a JSON format (JavaScript Object Notation). In each of these boundary layer points, the following values are specified: points on the y-coordinate (perpendicular to the surface), a velocity vector for each y point, a pressure gradient for each y point, a wall pressure spectrum and its frequency points and referential values (a referential temperature, pressure, length and velocity). Moreover, in the numerical datasets, static pressure and vorticity in each y point are specified.

3.1 Datasets

In this section, all datasets are described. Each dataset description also contains basic preprocessing of the raw boundary layer points. The *BLtools* Python package [54] is here used for the processing of each boundary layer point. To unify the notation, the x-coordinate in the following description is tangent to the boundary surface, and y-coordinate is perpendicular to the boundary surface.

Salze

Salze *et al.* [8] conducted an experiment where they investigated the wall pressure wavenumber frequency induced by a turbulent boundary layer in the presence of mean pressure gradient. The experiment was performed in an anechoic chamber in a subsonic wind tunnel with a test channel (Fig. 3.1). The wind tunnel was equipped with a proper acoustic treatment. Variable mean pressure gradient was accomplished

by changing the ceiling angle of the wind tunnel. The velocity profiles were measured by the hot-wire anemometry, and the wall pressure fluctuations were measured by a 1/8 inch microphone with a pinhole cap.

The Salze dataset yields 17 boundary layer points. Six points have a zero pressure gradient, seven points have an adverse pressure gradient, and four points have a favourable pressure gradient. All of the 17 points are used for the database.

The mean velocity profile data points are already stored in each of the boundary layer points. In this experimental dataset, the external velocity (the velocity at the edge of the boundary layer) U_e is defined as $U_e = u_{max}$. The boundary layer thickness δ describes a position where the mean velocity u asymptotically reaches the free-stream flow velocity U_∞. In this experimental dataset, the boundary layer thickness is defined as $\delta = y\,(0.99u_{max})$ (Fig. 3.2).

It is now possible to plot the velocity profiles with U_e and δ used as scaling (Fig. 3.3). In all of those 17 boundary layer points, the boundary layer stays attached. Therefore, there are no limitations in using these points for future model design. The Fig. 3.4 shows the wall pressure spectra for the same set of points as in the Fig. 3.3.

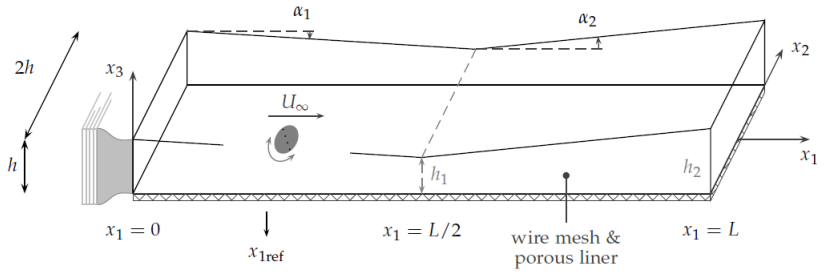


Figure 3.1: Salze – test channel. The height is $h = 250$ mm and the length is $L = 16h$. Parameters $h_1, h_2, \alpha_1, \alpha_2$ are changed according to the sought pressure gradient. Picture taken from [8].

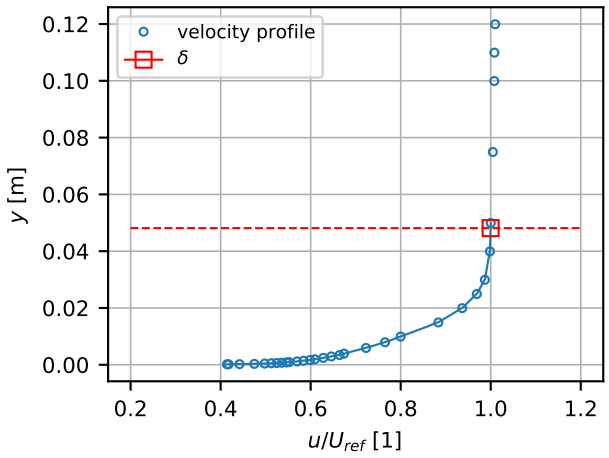


Figure 3.2: Salze, ZPG, $U_{ref} = 36\ \mathrm{m\,s^{-1}}$. Boundary layer thickness definition $0.99u_{max}$.

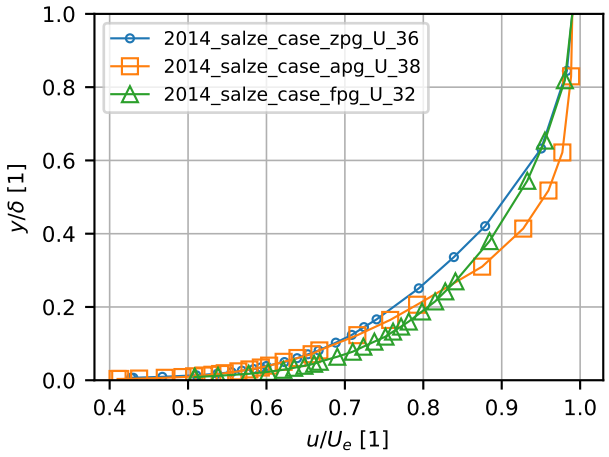


Figure 3.3: Salze – normalized velocity profiles for different pressure gradients.

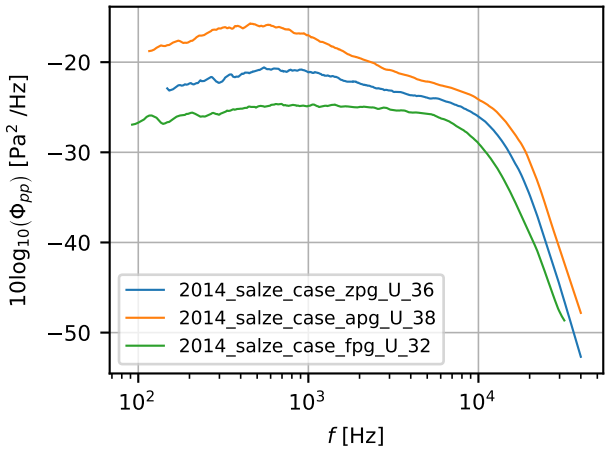


Figure 3.4: Salze – WPS for different pressure gradients.

Deuse

Deuse and Sandberg [9] investigated the self-noise of an isolated CD airfoil using direct noise computation (DNC) (Fig. 3.5). The DNC consists of numerical simulation of the full compressible turbulent flow, including the acoustic field. Specifically, they conducted four large eddy simulations and one direct numerical simulation using a high-order finite difference solver HiPSTAR. The LES was performed for a constant angle of attack 8°, a constant chord-based Reynolds number 10^5 and for four free-stream Mach numbers $[0.2; 0.3; 0.4; 0.5]$. The DNS was performed only for $M = 0.4$ to validate the LES results. In this work, only the results with free-stream Mach number $M = 0.2$ are considered.

Under the previous conditions, a separation bubble forms at the airfoil leading edge on the suction side (Fig. 3.6). Therefore, in this place, a laminar boundary layer separates, transits to a turbulent boundary layer and then reattaches. On the pressure side, the boundary layer is fully laminar until the trailing edge.

This dataset yields 20 boundary layer points in total. The points vary with the chord position, where the origin of the chord coordinate is at the trailing edge. The first point is positioned at the relative chord $x_c = 0.02$ and the last point is at $x_c = 0.95$.

The mean velocity profile datapoints are already stored in each of the boundary layer points as in Salze dataset. However, different definition of the external velocity U_e is used here. Since the flow is not constant outside of the boundary layer, using the u_{max} definition for the external velocity might lead to an overestimation of the resultant boundary layer thickness. A pseudo-velocity u_{ps} is defined instead. Firstly, the vorticity is

$$\boldsymbol{\omega}_{vor} = \nabla \times \mathbf{u} = \left(\frac{\partial u_z}{\partial y} - \frac{\partial u_y}{\partial z}\right)\mathbf{i} + \left(\frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x}\right)\mathbf{j} + \left(\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y}\right)\mathbf{k}. \tag{3.1}$$

The index *vor* is added here to distinguish the vorticity from the angular frequency. Then, considering only a 2D boundary layer and neglecting the term $\partial u_y/\partial x$, the pseudo-velocity u_{ps} can be determined as

$$u_{ps}(y) = -\int_0^y \omega_{vor,z}\, dy. \tag{3.2}$$

Thus, the boundary layer thickness δ is calculated as $\delta = y\,(0.99 u_{ps-max})$ and the U_e is then defined as $U_e = u_x\,(y = \delta)\,/0.99$ (Fig. 3.7). The advantage of the pseudo-velocity is that the vorticity goes to zero when reaching the outside of the boundary layer (irrotational flow).

The Fig. 3.8 illustrates the mean velocity profiles scaled with U_e and δ for several boundary layer points along the airfoil chord. The profiles with relative chord position x_c up to 0.85 indicate that the boundary layer stays attached. However, nearer to the leading edge, the separation bubble occurs, and the mean velocity profile at the $x_c = 0.90$ has an opposite direction (near the wall) to the free-stream flow. This is in agreement with the Fig. 3.6. In the default dataset, there are no other boundary layer points available between the $x_c = 0.85$ and $x_c = 0.90$. The flow in the separation bubble will not be investigated nor used for the wall pressure spectra model. In consequence, the number of used boundary layer points is reduced to 17.

In the Fig. 3.9 the raw wall pressure spectra are illustrated.

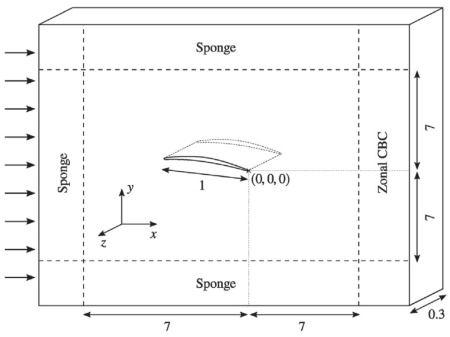


Figure 3.5: Deuse – simulation geometry. Picture taken from [9]

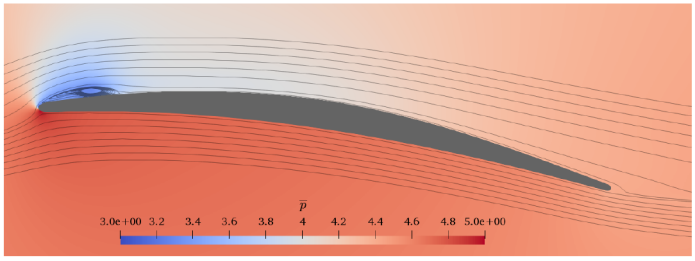


Figure 3.6: Deuse – average non-dimensional pressure field and mean streamlines, $M = 0.4$, DNS results. Picture taken from [9].

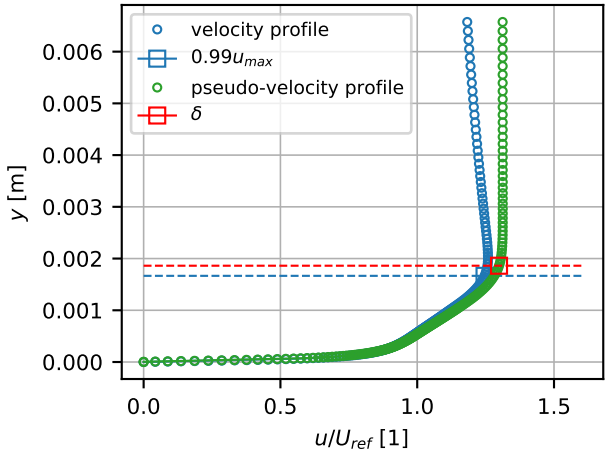


Figure 3.7: Deuse, $x_c = -0.4$, $U_{ref} = 69.438\ \mathrm{m\,s^{-1}}$. Boundary layer thickness definition $0.99u_{ps-max}$.

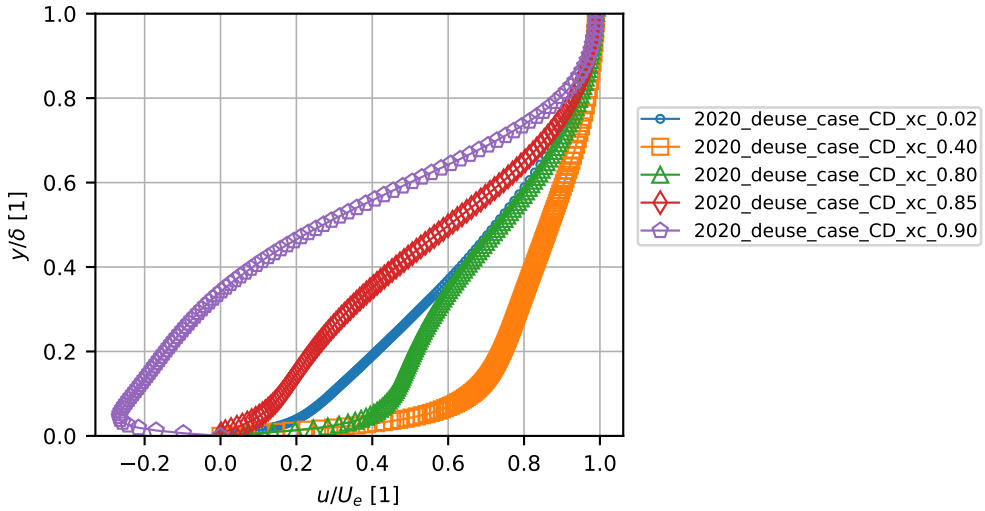


Figure 3.8: Deuse – normalized velocity profiles for different airfoil positions.

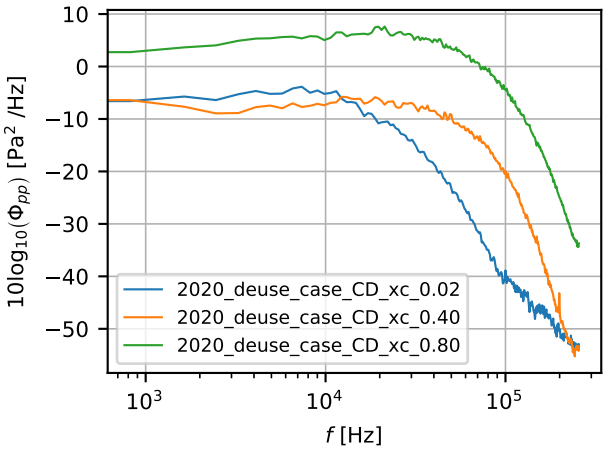


Figure 3.9: Deuse – WPS for different airfoil positions.

Wu

Wu *et al.* [10] conducted a 3D compressible DNS of a CD airfoil in a wind-tunnel flow at a constant angle of attack 8°, constant chord-based Reynolds number 1.5×10^5 and constant free-stream Mach number 0.25. The simulation accounted for the aerodynamic installation effects of the wind tunnel by setting proper inflow boundary profiles. The simulation was also compared to particle-image velocimetry and hot-wire measurements to assure the correctness of the DNS results.

The flow state is similar to the Deuse case, as a separation bubble forms at the airfoil leading edge on the suction side and on the pressure side, the flow is laminar.

The Wu dataset yields 20 boundary layer points across the airfoil chord. As in Deuse dataset, the origin of the chord coordinate is at the trailing edge, and the relative chord position varies from $x_c = 0.02$ to $x_c = 0.95$.

The same method as in Deuse dataset is used for obtaining U_e and δ. The raw velocity and pseudo-velocity profiles and their respective δ are illustrated in the Fig. 3.11.

The Fig. 3.12 describes the mean velocity profiles scaled with U_e and δ. As in the Deuse dataset, due to the separation bubble, there is a mean velocity profile, where the flow has an opposite direction near the wall. This one point at $x_c = 0.95$ is not going to be used. This reduces the total number of Wu dataset boundary layer points to 19.

In the Fig. 3.13 the raw wall pressure spectra are illustrated. The spectrum contains a hump at middle frequencies at the point near the separation bubble. There is also a hump in the spectrum at high frequencies at the point near the leading edge. This high frequency hump was observed for the first time in such a flow case by Wu *et al.* [10], and it is suspected that the near wake influences it. Deuse and Sandberg [9] also observed the high frequency hump but for higher Mach numbers.

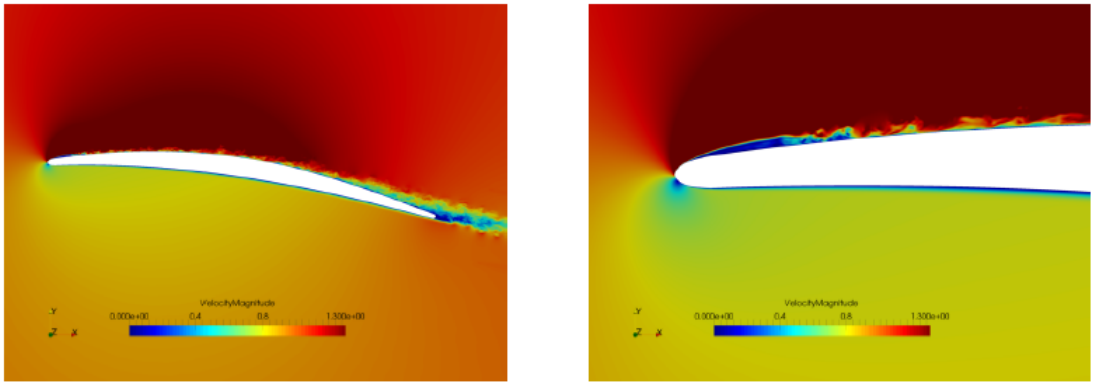


Figure 3.10: Wu – instantaneous velocity field and the detail on the leading edge, 3D DNS results. Picture taken from [10].

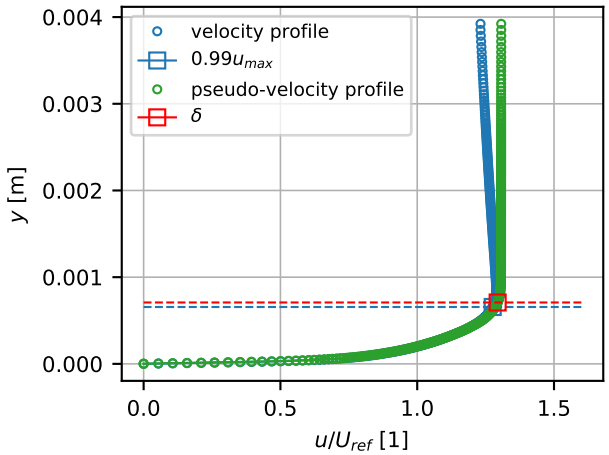


Figure 3.11: Wu, $x_c = -0.4$, $U_{ref} = 86.806\ \mathrm{m\,s^{-1}}$. Boundary layer thickness definition $0.99u_{ps-max}$.

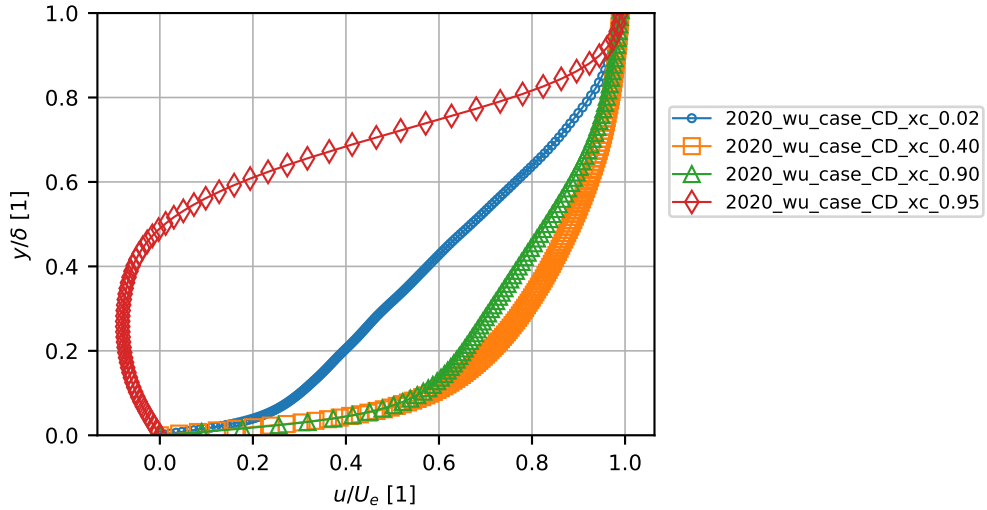


Figure 3.12: Wu – normalized velocity profiles for different airfoil positions.

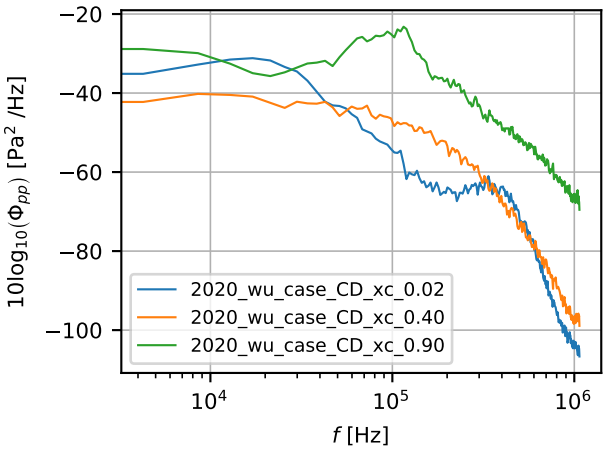


Figure 3.13: Wu – WPS for different airfoil positions.

Christophe

Christophe *et al.* [11] applied an uncertainty quantification framework to the broadband trailing edge noise of a CD airfoil. One part of this study was to prepare LES of the CD airfoil embedded in the potential core of the jet in the anechoic wind tunnel. The airfoil chord length is 0.1356 m, the free stream velocity is 16 ms^{-1}, therefore resultant chord-based Reynolds number is 1.6×10^5. The angle of attack is varied and thus yield eight different cases (Tab. 3.1).

Cases 1 to 6 are very similar to the Deuse and Wu CD airfoil simulations. Again, there is a laminar separation bubble at the leading edge on the suction side of the airfoil. This triggers the transition to the turbulent boundary layer, which stays attached until the trailing edge. At larger angles of attack, the recirculation bubble increases, but the flow stays attached. In case 7, the previous separation at the leading edge is weaker, and this phenomenon only applies for some narrow span. In case 8, the angle of attack is small enough that the separation moves beyond the mid chord (near the trailing edge).

Table 3.1: Christophe – the angle of attack for each case.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
AOA [°]	6.00	5.86	5.42	4.76	4.00	3.24	2.58	2.14

The Christophe dataset yields 18 points along the airfoil chord for each angle of attack, which means 144 boundary layer points in total. The boundary layer points vary from $x_c = 0.05$ to $x_c = 0.90$, where the origin is, again, at the trailing edge.

The same method as in the previous numerical datasets is used for obtaining U_e and δ. The raw velocity and pseudo-velocity profiles and their respective δ are illustrated in the Fig. 3.14.

The Fig. 3.15 describes the mean velocity profiles scaled with U_e and δ for the case 1. The available boundary layer points in cases 1 to 7 do not contain any point

in the separation bubble at the leading edge. Thus it is not necessary to remove any point in these cases. However, in case 8 (Fig. 3.16), the separation is beyond the mid chord, and the only boundary layer points available are after the separation. Therefore, case 8 is not going to be used for the wall pressure spectra model. This reduces the number of boundary layer points to 126.

In the Fig. 3.17 and Fig. 3.18 the raw wall pressure spectra are illustrated. As seen in the previous numerical datasets, there is a hump in the high frequency region near the trailing edge.

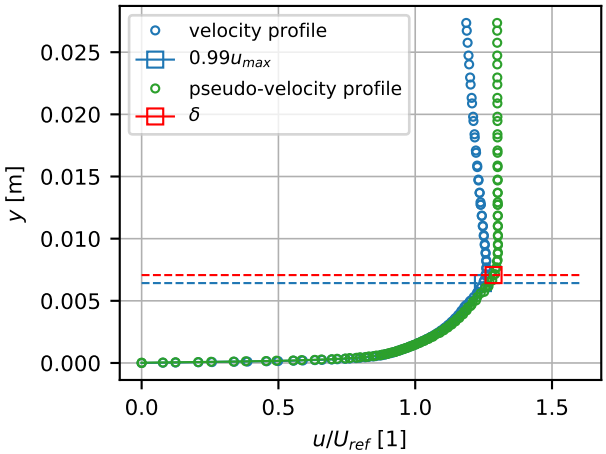


Figure 3.14: Christophe, $x_c = -0.4$, $U_{ref} = 16\ \mathrm{m\,s^{-1}}$. Boundary layer thickness definition $0.99u_{ps-max}$.

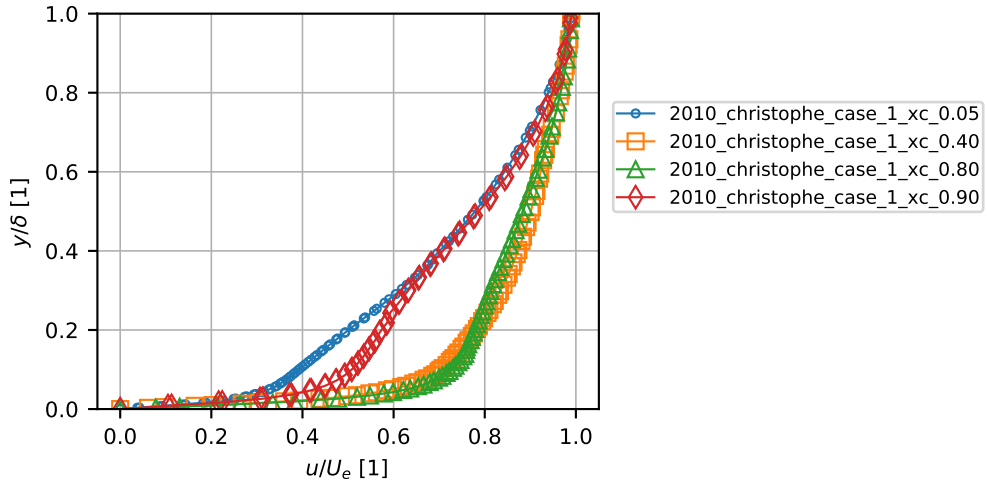


Figure 3.15: Christophe – normalized velocity profiles for different airfoil positions.

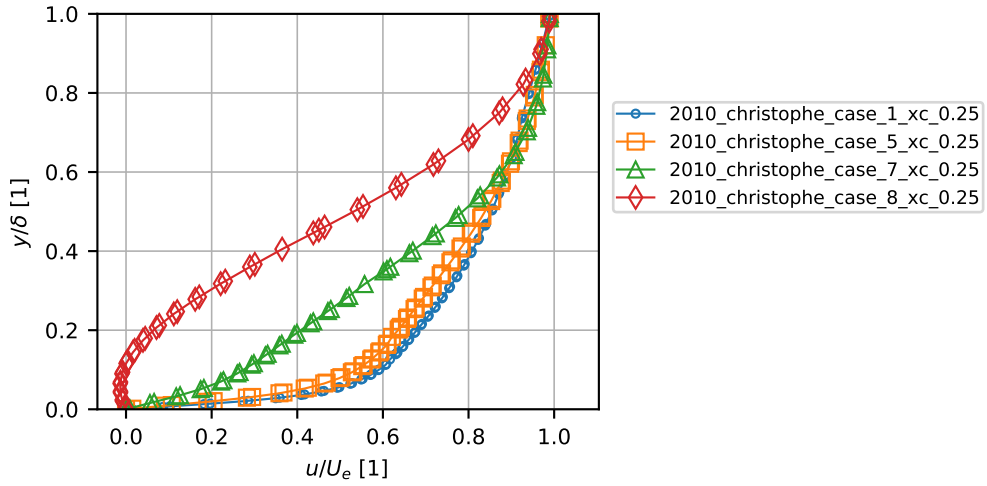


Figure 3.16: Christophe – normalized velocity profiles for different angles of attack.

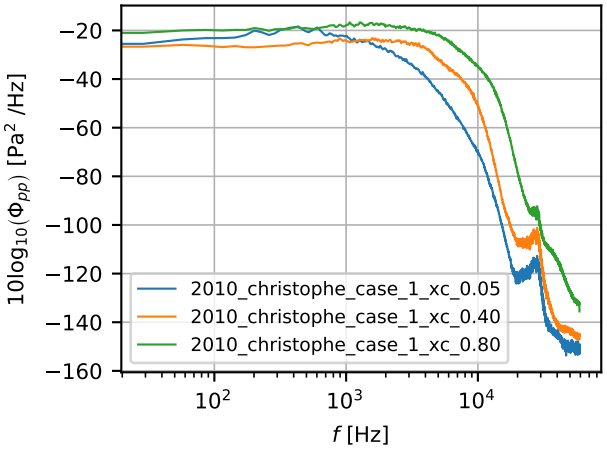


Figure 3.17: Christophe – WPS for different airfoil positions.

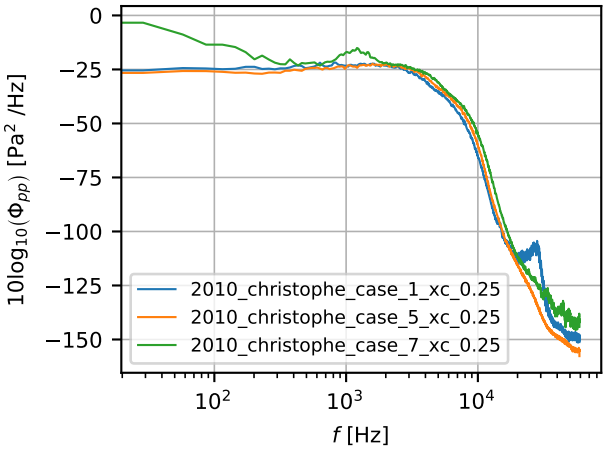


Figure 3.18: Christophe – WPS for different angles of attack.

3.2 Boundary Layer Parameters Selection

In the previous section, the selected datasets were introduced and discussed. This section continues with these datasets and also continues in further processing.

The main question in this further processing of the boundary layer datasets is: *What boundary layer parameters to select to compose a new data-driven model?* The answer to this question might be found in the subsection 2.1.2. There is a wide collection of parameters that can be somehow extracted from the boundary layer. Firstly, only the dimensional ones (with one exception) are extracted. The resultant selection is collected in the Tab. 3.2.

The U_e and δ were already introduced in the previous sections. To sum it up, the definition of U_e is, for the experimental results, the u_{max}, and for the numerical results, the $u_x\left(y=\delta\right)/0.99$. The definition of δ is, for the experimental results, the $y\left(0.99u_{max}\right)$, and for the numerical results, the $y\left(0.99u_{ps-max}\right)$.

To calculate the boundary layer displacement thickness δ^* and momentum thickness θ, the following definitions are used

$$\delta^* = \int_0^\delta \frac{1-u\left(y\right)}{U_e}\, dy, \tag{3.3}$$

$$\theta = \int_0^\delta \frac{u\left(y\right)}{U_e}\left(\frac{1-u\left(y\right)}{U_e}\right) dy. \tag{3.4}$$

Table 3.2: Selected boundary layer parameters.

Parameter	**Symbol**	**Where to obtain**
External velocity	U_e	Velocity profile
Boundary layer thickness	δ	Velocity profile
Displacement thickness	δ^*	Velocity profile
Momentum thickness	θ	Velocity profile
Wake parameter	Π	Velocity profile
Wall shear stress	τ_w	Velocity profile
Pressure gradient	$\mathrm{d}p/\mathrm{d}x$	Dataset
Kinematic viscosity	ν	Sutherland's law
Fluid density	ϱ	Ideal gas law
Speed of sound	c_0	Speed of sound id. gas

The pressure gradient $\mathrm{d}p/\mathrm{d}x$ is taken as a gradient in the x direction at the first y point in the boundary layer.

The fluid density ϱ is computed from the ideal gas law

$$\varrho = \frac{p}{RT_{ref}}, \tag{3.5}$$

where the specific gas constant is $R = 287.058$ and the pressure is considered as the atmospheric pressure $p_{atm} = 101325$ Pa.

The speed of sound c_0 is calculated for ideal gas as

$$c_0 = \sqrt{\gamma R T_{ref}}, \tag{3.6}$$

where the heat capacity ratio for ideal gas is $\gamma = 1.4$.

The kinematic viscosity can be obtained from the dynamic viscosity μ and the Sutherland's law [55]

$$\nu = \frac{\mu}{\varrho}, \tag{3.7}$$

$$\mu = \mu_0 \left(\frac{T_{ref}}{T_0}\right)^{\frac{3}{2}} \frac{T_0 + S}{T_{ref} + S}, \tag{3.8}$$

where $\mu_0 = 1.711 \times 10^{-5}$ Pa s, $T_0 = 273.15$ K and $S = 110.4$ K.

The remaining parameters are the Coles's wake parameter Π [56] and the wall shear stress τ_w, where both of them are directly connected to the mean velocity profile. The τ_w is defined as [13]

$$\tau_w = \varrho\nu \left(\frac{\mathrm{d}\langle u \rangle}{\mathrm{d}y}\right)_{y=0}. \tag{3.9}$$

Since there are no data in our dataset that would contain τ_w, it has to be somehow derived from the mean velocity profile. One way could be to compute the discretized derivative in the definition for the first set of points along the y coordinate in the boundary layer. However, this approach can lead to a strong error when the boundary layer is not adequately discretised. In this work, another way of computing τ_w is used. Moreover, instead of computing τ_w directly, the friction velocity u_τ is the one sought. τ_w and u_τ are connected through the definition

$$u_\tau = \sqrt{\frac{\tau_w}{\varrho}}. \tag{3.10}$$

The problem now switched to finding the wake parameter Π and the friction velocity u_τ. Both of them are connected to the mean velocity profile. Therefore, to find them, it is necessary to introduce the self-similar (non-dimensionalized) form of the mean velocity profile and see how are those parameters connected to it.

Self-Similar Mean Velocity Profile

The mean velocity profile near the wall can be scaled to its self-similar form. However, there is no universal scaling for the whole velocity profile since different physical phenomena influence each part of the velocity profile. In other words, some particular scales are only valid for some part of the velocity profile. Nonetheless, it is possible to merge a set of scaling laws to build a *composite mean velocity profile* [57, 58]. In

the following lines, the definition of the scaled distance from the wall y^+ and the scaled mean velocity u^+ is

$$y^+ = \frac{yu_\tau}{\nu}, \tag{3.11}$$

$$u^+ = \frac{\langle u \rangle}{u_\tau}. \tag{3.12}$$

The mean velocity profile can be divided into the following layers [13]:

- **Inner layer** – Location: $y/\delta < 0.1$. Determined mainly by viscous scales.
- **Overlap region** – Location: $y^+ > 50; y/\delta < 0.1$. Covers the end of the inner layer and the beginning of the outer layer.
- **Outer layer** – Location: $y^+ > 50$. The effect of viscosity on the velocity profile is negligible.

It is good to note that there is no sharp boundary between the layers/regions, and they partially overlap.

The *law of the wall* [59] $u^+ = f\left(y^+\right)$ is applicable in the inner layer. The inner layer can be further divided into the viscous sublayer, buffer region and log-law region, where the following relations apply

$$u^+ = y^+ \quad \text{(viscous sublayer – } \textit{linear law}\text{, } y^+ < 5), \tag{3.13}$$

$$u^+ = f\left(y^+\right) \quad \text{(buffer region, } 5 < y^+ < 30), \tag{3.14}$$

$$u^+ = \frac{1}{\kappa} \ln\left(y^+\right) + B \quad \text{(log-law region, } y^+ > 30), \tag{3.15}$$

where the *log-law* was firstly derived by Kármán [60]. The coefficients κ and B are generally considered as constants $\kappa = 0.41$ and $B = 5.2$ [13] for canonical flows. But it will be shown that the coefficients are dependent on the pressure gradient.

A single formula law of the wall is also possible. The first such formulation was introduced by Spalding [61]. Musker [62] wanted to overcome the implicit form of Spalding's law and also wanted to improve the satisfaction of the boundary conditions, which resulted in a new single formula law of the wall. The dimensionless velocity gradient of the latter model is

$$\frac{\mathrm{d}u^+}{\mathrm{d}y^+} = \frac{\kappa + Cy^{+\,2}}{\kappa + Cy^{+\,2} + C\kappa y^{+\,3}}, \tag{3.16}$$

where C is the constant of proportionality. To obtain the final algebraic form of the law, one has to integrate this equation with some values for κ and C. The constant C is found by minimizing the difference (e.g. by trial and error) between the Musker law and the log-law as $y^+ \to \infty$.

In the outer layer, the law of the wall differs from the actual mean velocity profile with the increasing distance from the wall. Coles [56] suggested a correction in the form

$$u^+ = f\left(y^+\right) + \frac{\Pi}{\kappa}\mathcal{W}\left(\frac{y}{\delta}\right), \tag{3.17}$$

where the term $(\Pi/\kappa)\,\mathcal{W}$ is the *law of the wake*, Π is the wake parameter and $\mathcal{W}$ is the wake function.

Chauhan *et al.* [58] suggested an exponential wake function

$$\mathcal{W}_{exp} = \frac{1 - \exp\left[-\frac{1}{4}\left(5a_2 + 6a_3 + 7a_4\right)\eta^4 + a_2\eta^5 + a_3\eta^6 + a_4\eta^7\right]}{1 - \exp\left[-\left(a_2 + 2a_3 + 3a_4\right)/4\right]} \times \\ \times 2\left(1 - \frac{1}{2\Pi}\ln\left(\eta\right)\right), \tag{3.18}$$

where $\eta = y/\delta$, $a_2 = 132.8410$, $a_3 = -166.2041$ and $a_4 = 71.9114$. This wake function should also assure that the slope $\mathrm{d}\langle u\rangle/\mathrm{d}y \to 0$ at the boundary layer edge.

As it was said previously, the coefficients κ and B may not be independent of the pressure gradient. Nickels [63] investigated the effects of pressure gradient on the inner region of turbulent wall-bounded flows and came up with a relationship

$$\frac{\kappa}{\kappa_0} = \sqrt{\frac{1}{1 + p_x^+ y_c^+}}, \tag{3.19}$$

where κ_0 is the coefficient κ for zero pressure gradient flows, $p_x^+ = \left(\nu/\varrho u_\tau^3\right)\left(\mathrm{d}p/\mathrm{d}x\right)$ is the dimensionless pressure and the y_c^+ is the dimensionless viscous sublayer thickness or it may be called a critical distance, where the viscous sublayer becomes turbulent. With a reference to Nickels [63], the y_c^+ may be considered as similar to the critical Reynolds number Re_c. From his investigation, the y_c^+ is around 11 to 12 for zero pressure gradient flows. He considered the $\kappa_0 = 0.39$ for zero pressure gradient. The model (Eq. (3.19)) then suggests the κ is less than 0.39 for strong adverse pressure gradient flows and the κ is more than 0.39 for strong favourable pressure gradient flows.

Chauhan *et al.* [64] described the dependency of κB on B for different pressure gradient data. Nagib and Chauhan [65] took this dependency and fitted a functional form

$$\kappa B = 1.6\left[\exp\left(0.1663B\right) - 1\right]. \tag{3.20}$$

With this relation, one can find the coefficient B with a known κ.

In this work, to find the wake parameter Π and the friction velocity u_τ, a self-similar composite mean velocity profile has to be fit on each velocity profile in the complete dataset. The composite profile (Fig. 3.19) consists of Musker profile (integrating the Eq. (3.16)) and the law of the wake with an exponential wake function (Eq. (3.18)).

$$u^+_{Comp} = u^+_{Musker}\left(y^+\right) + u^+_{exp}\left(\eta\right). \tag{3.21}$$

The main parameter governing the fit is u_τ. The coefficients κ and B are calculated from the Eq. (3.19) and the Eq. (3.20).

The wake parameter Π is used to correct the last velocity point to assure that it will be equal to $0.99U_e$

$$\Pi = \left(\frac{0.99U_e}{u_\tau} - u^+_{log-law}\right)\frac{\kappa}{\mathcal{W}}. \tag{3.22}$$

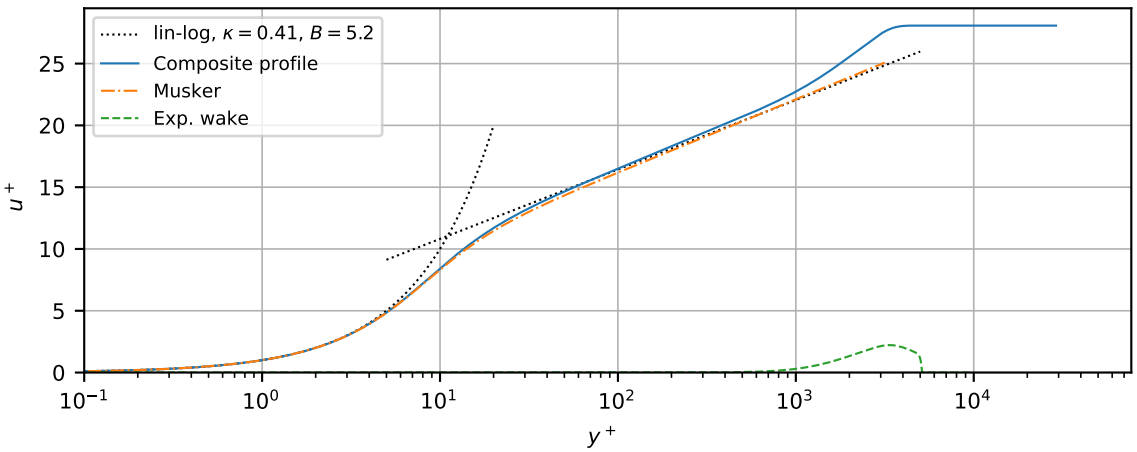


Figure 3.19: The composite mean velocity profile composition. Salze APG ($U_{ref} = 38\ \mathrm{m\,s^{-1}}$) boundary layer point was used for the profile.

The composite mean velocity profiles for the previously mentioned three points from the Salze dataset are in the Fig. 3.20. The wake correction is almost similar for the zero and adverse pressure gradient case. However, for the favourable pressure gradient, the wake is rather weak. Anyway, all the fits follow the data points.

The composite mean velocity profiles for the selected Deuse's data points are in the Fig. 3.21. The correction in the wake region due to the pressure gradient is strong in this case. Especially at the trailing edge, there is a large adverse pressure gradient. The pressure gradient is rather mild at the middle part of the airfoil, where the pressure gradient is evolving from the favourable to the adverse pressure gradient. Thus the wake correction is also small. The composite profiles, again, fit the data points successfully. In the Christophe and Wu dataset, the fit of the composite profiles is similar to the Deuse dataset. Therefore the illustrations are collected in the Appendix B.

In the previous lines, the process, how the selected boundary layer parameters (Tab. 3.2) are obtained was described. To sum up the obtained values, the ranges for each parameter are in the Tab. 3.3. The wake parameter Π cannot be zero as the parameter is in the denominator in the Eq. (3.18). Therefore, the minimum of the wake parameter is set to 1×10^{-6}. It is worth noting that these parameters are not directly used as inputs into the oncoming neural network wall pressure spectra models.

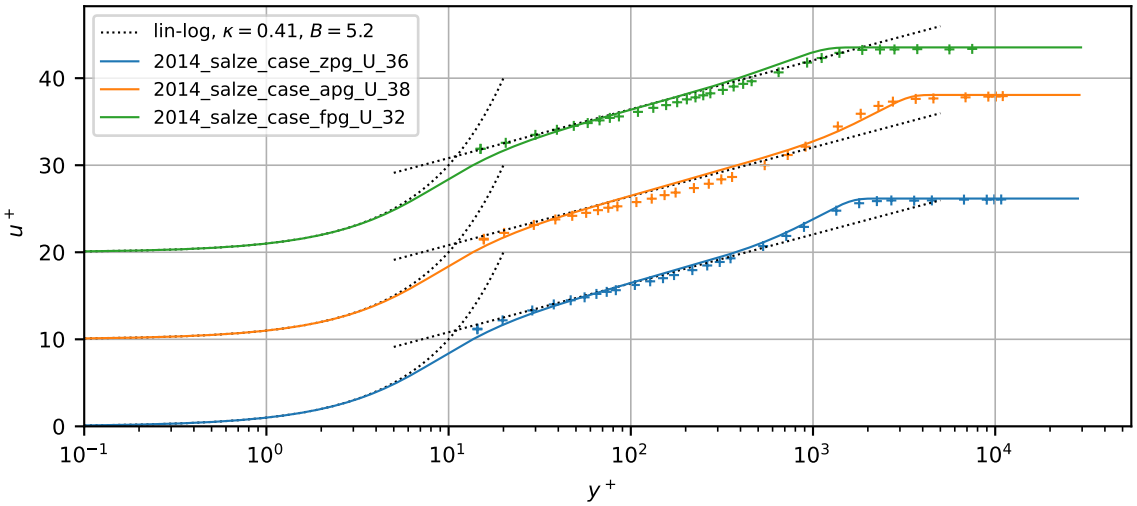


Figure 3.20: Examples of the composite mean velocity profiles for the Salze adverse, zero and favourable pressure gradient dataset. The profiles are shifted by $u^+ = 10$.

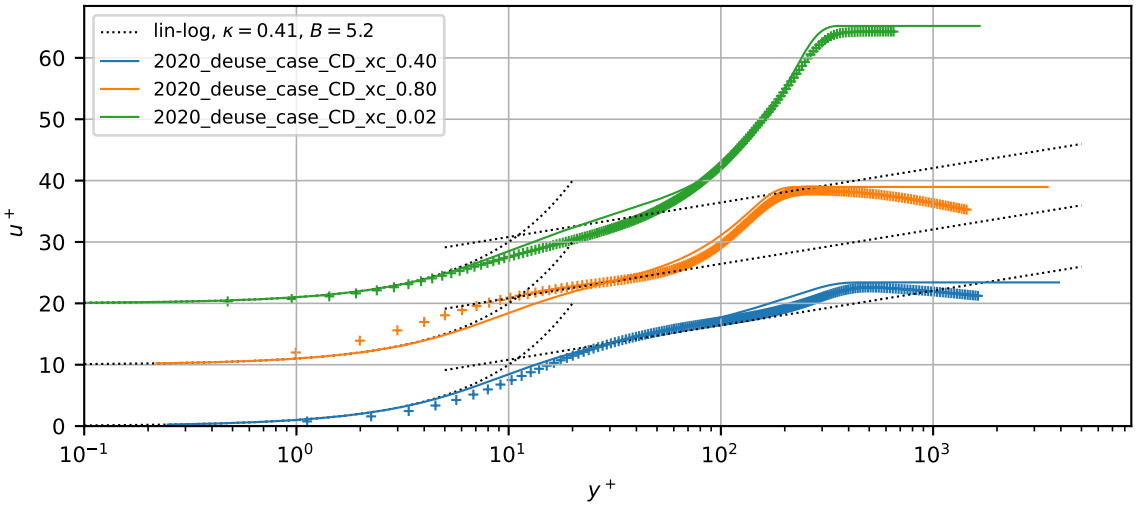


Figure 3.21: Examples of the composite mean velocity profiles for the Deuse dataset. The profiles are shifted by $u^+ = 10$.

Table 3.3: The obtained boundary layer parameters dataset overview. The references of the data are Salze *et al.*(2014) [8], Wu *et al.*(2018) [10], Deuse and Sandberg(2020) [9] and Christophe *et al.*(2014) [11].

Dataset	N. of boundary layer points	U_e [m s^{-1}]	δ [mm]	δ^* [mm]	θ [mm]	Π [1]	τ_w [Pa]	$\frac{\mathrm{d}p}{\mathrm{d}x}$ [Pa m^{-1}]	$\nu \times 10^6$ [m^2 s^{-1}]	ϱ [kg m^{-3}]	c_0 [m s^{-1}]
Salze											
ZPG	6	10.9 to 75.9	19.451 to 28.103	2.709 to 3.554	2.064 to 2.684	0.232 to 0.514	0.284 to 8.937	0	15.107 to 15.262	1.193 to 1.200	343.8 to 344.8
APG	7	8.3 to 76.7	29.486 to 69.682	4.790 to 8.173	3.445 to 5.837	0.194 to 0.822	0.138 to 7.219	13 to 1128	14.735 to 15.408	1.187 to 1.217	341.4 to 345.8
FPG	4	10.3 to 63.1	15.205 to 18.992	1.617 to 2.006	1.305 to 1.600	1×10^{-6} to 0.007	0.313 to 7.657	−2518 to −69	14.591 to 15.225	1.195 to 1.224	340.5 to 344.6
Wu											
Case 1	19	91.2 to 116.9	0.301 to 1.345	0.058 to 0.469	0.039 to 0.219	1×10^{-6} to 2.306	6.335 to 56.367	−776650 to 307515	15.643	1.177	347.2
Deuse											
Case 1	17	72.5 to 97.5	1.027 to 3.388	0.232 to 1.096	0.162 to 0.518	0.141 to 2.265	3.145 to 22.644	−24686 to 230708	15.643	1.177	347.2
Christophe											
Case 1	18	16.7 to 22.1	3.156 to 11.329	0.641 to 3.133	0.444 to 1.700	1×10^{-6} to 1.416	0.305 to 1.541	−314 to 2225	14.835	1.212	342.1
Case 2	18	16.7 to 22.1	3.135 to 11.035	0.608 to 3.027	0.424 to 1.640	1×10^{-6} to 1.402	0.313 to 1.565	−325 to 2113	14.835	1.212	342.1
Case 3	18	16.9 to 21.8	2.670 to 1.000	0.515 to 2.671	0.350 to 1.466	1×10^{-6} to 1.283	0.348 to 1.638	−413 to 2115	14.835	1.212	342.1
Case 4	18	16.9 to 21.7	2.229 to 7.978	0.394 to 2.242	0.269 to 1.209	1×10^{-6} to 1.333	0.361 to 1.757	−686 to 2141	14.835	1.212	342.1
Case 5	18	17.1 to 21.8	1.616 to 6.509	0.290 to 1.926	0.194 to 1.012	1×10^{-6} to 1.432	0.367 to 1.866	−2223 to 2169	14.835	1.212	342.1
Case 6	18	17.3 to 21.9	1.190 to 5.355	0.243 to 1.631	0.144 to 0.850	1×10^{-6} to 1.421	0.395 to 1.944	−1988 to 2180	14.835	1.212	342.1
Case 7	13	17.4 to 22.1	0.938 to 4.743	0.247 to 1.078	0.112 to 0.628	0.673 to 1.520	0.621 to 1.381	−1009 to 2793	14.835	1.212	342.1
All	174	8.3 to 116.9	0.301 to 69.682	0.058 to 8.173	0.039 to 5.837	1×10^{-6} to 2.306	0.138 to 56.367	−776650 to 307515	14.591 to 15.643	1.177 to 1.224	340.5 to 347.2

3.3 Wall Pressure Spectra – Data and Semi-Empirical Models

This section is devoted to a brief description of the obtained wall pressure spectra, scaling, and semi-empirical models.

As it was described in the subsection 2.1.1, and especially in the Tab. 2.1, various scalings are used to collapse the wall pressure spectra. Concretely, the scaling is varying from outer to inner variable scaling with increasing frequency. This ensures that the spectra should collapse in each frequency region. In the Fig. 3.22 all the wall pressure spectra from the Salze dataset are plotted with different scaling. In the Fig. 3.22a the time scale δ^*/U_e and the pressure scale τ_w are used. This scaling should collapse the spectra in the lower frequencies. This collapse is visible for the zero pressure gradient data, and the spectra also collapse partially in the mid frequencies. The collapse is also visible for the adverse pressure gradient (except one boundary layer point), and it is less visible for the favourable pressure gradient. The complete dataset, however, does not collapse to one curve. The same, as in the previous scaling, applies to the Fig. 3.22b. Here, the time scale δ/U_e and the pressure scale τ_w are used. In the last, outer variable scaling, in Fig. 3.22c the complete dataset almost collapses to one curve. The best collapse is again achieved with the zero pressure gradient dataset. Here, the time scale ν/u_τ^2 and the pressure scale τ_w are used.

Comparing the wall pressure spectra with the semi-empirical models referenced in the subsection 2.1.2, the following conclusions can be made.

Salze's wall pressure spectra for the zero pressure gradient agrees with almost all the semi-empirical models (Fig. 3.23a), except the Kamruzzaman model does not respect the shape in the low and in the high frequencies. The best prediction here seems to be from the Hu model and the Lee model. The Rozenberg model, in the case of zero pressure gradient, should collapse to the Goody model. Indeed, the Rozenberg model collapses to the Goody model. For Salze's wall pressure spectra in the adverse pressure gradient (Fig. 3.23b), the prediction is still good. The best ones are probably from the Hu model and the Goody model. In the latter, the good prediction is surprising since the model was not built on adverse pressure gradient data. In the case of favourable pressure gradient, none of the semi-empirical models predicts the spectra as in the previous cases. It is not surprising since the favourable pressure gradient data were sparsely used in building these models. The best prediction here is probably by the Hu model, where some favourable pressure gradient data were used to build the model.

Looking at Deuse's wall pressure spectra in Fig. 3.24, it can be seen that with increasing distance from the trailing edge, the semi-empirical models are getting worse. This is, in fact, true for this whole airfoil. At least the shape of the obtained spectra is respected at the trailing edge, but the amplitude is not.

The airfoil wall pressure spectra from Wu and Christophe (see the Appendix C) show similar bad predictions from the semi-empirical models. Moreover, there are some humps visible that are described in section 3.1.

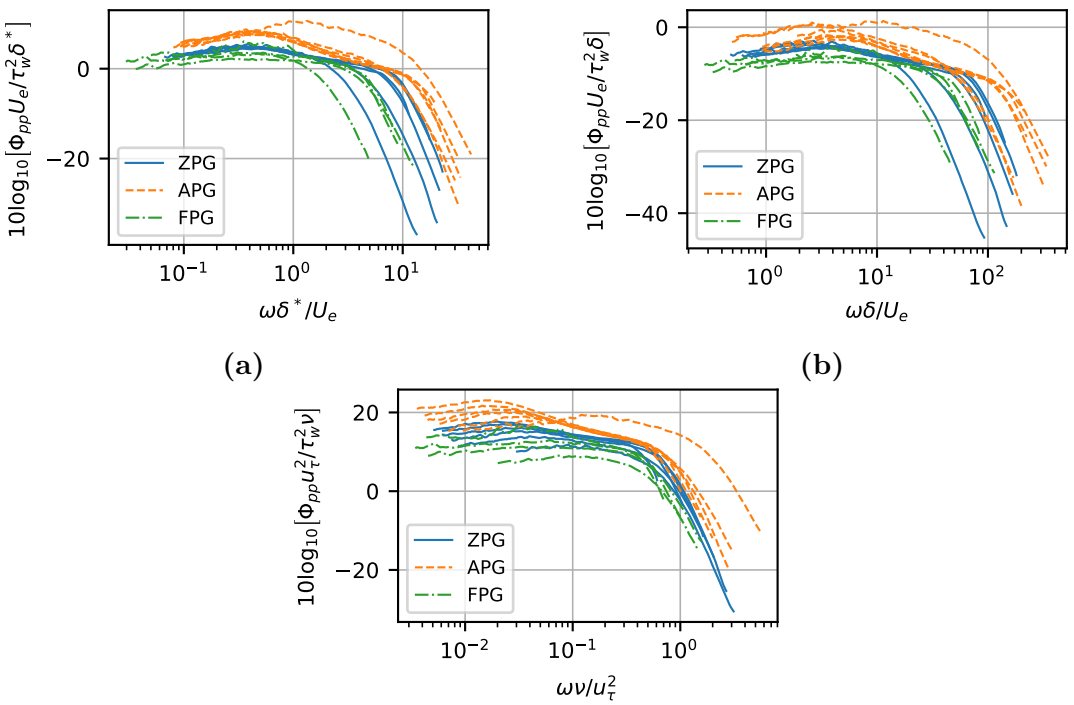
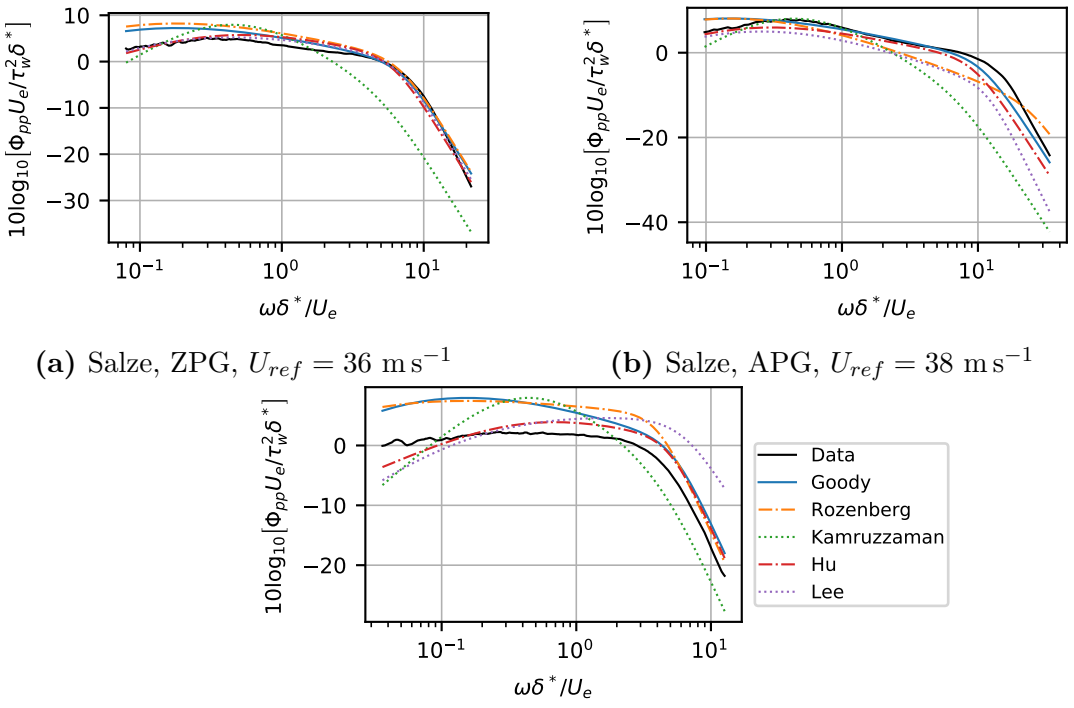


Figure 3.22: Complete Salze's wall pressure spectra plotted with different scaling.

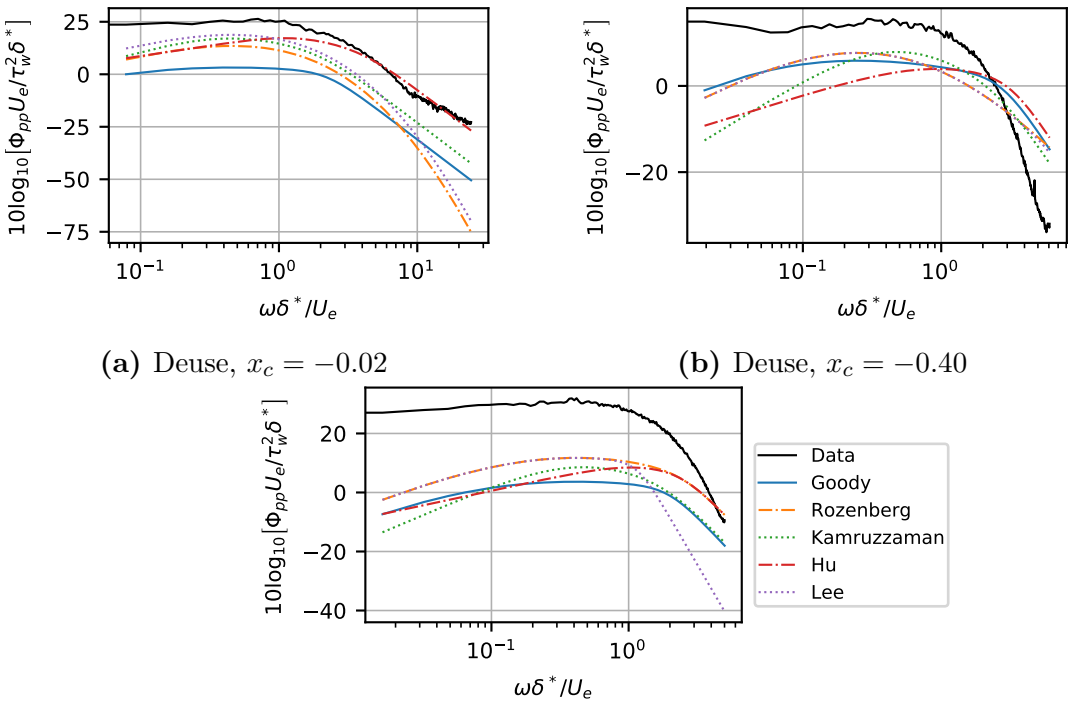


(a) Salze, ZPG, $U_{ref} = 36\ \mathrm{m\,s^{-1}}$

(b) Salze, APG, $U_{ref} = 38\ \mathrm{m\,s^{-1}}$

(c) Salze, FPG, $U_{ref} = 32\ \mathrm{m\,s^{-1}}$

Figure 3.23: Selected Salze's wall pressure spectra with their respective semi-empirical models.



(a) Deuse, $x_c = -0.02$ **(b)** Deuse, $x_c = -0.40$

(c) Deuse, $x_c = -0.80$

Figure 3.24: Selected Deuse's wall pressure spectra with their respective semi-empirical models.

3.4 Model Inputs

As it was stated in the subsection 2.1.1, the self-similarity is a core principle in fluid mechanics. To satisfy the similitude feature in the proposed models, there has to be a way, how to find a relation consisting of dimensionless (or scaled) parameters. One of the ways is to use the Buckingham Π-Theorem [66].

The default set of parameters was obtained in the section 3.2. The set can be written in the following implicit relation

$$f\left(\delta, \delta^*, \theta, U_e, \nu, \rho, \tau_w, \frac{\mathrm{d}p}{\mathrm{d}x}, c_0, \Pi, \Phi_{pp}, \omega\right) = 0 \tag{3.23}$$

where all of the parameters are dimensional, except the Coles's wake parameters Π. Now, it is possible to write down the dimensions of those parameters

$$\begin{aligned} [\delta] &= m = L, & \left[\frac{\mathrm{d}p}{\mathrm{d}x}\right] &= ML^{-2}T^{-2}, \\ [\delta^*] &= L, & [\Phi_{pp}] &= M^2L^{-2}T^{-3}, \\ [\theta] &= L, & [c_0] &= LT^-1, \\ [U_e] &= L^1T^{-1}, & [\omega] &= T^{-1}, \\ [\nu] &= L^2T^{-1}, & [\Pi] &= 1, \\ [\varrho] &= ML^{-3}, & & \\ [\tau_w] &= ML^{-1}T^{-2}, & & \end{aligned} \tag{3.24}$$

where L, M and T denote the common *length*, *mass* and *time* scale, or so-called *fundamental units*. It can be assumed that those units are e.g. *meter*, *kilogram* and *seconds*, respectively. The Π-Theorem states, that it is possible to find an unknown function of dimensionless parameters and that the number of such dimensionless parameters is

$$i = n - k, \tag{3.25}$$

where n is, in here, the number of dimensional parameters from the Eq. (3.23) and k is, in here, the number of independent fundamental units. Thus, in this case, $n = 11$ and $k = 3$ results in $i = 8$. Therefore, the unknown function is going to have $8 + 1$ dimensionless parameters (together with the wake parameter Π)

$$\psi\left(\Pi_1, \Pi_2, \Pi_3, \Pi_4, \Pi_5, \Pi_6, \Pi_7, \Pi_8, \Pi\right) = 0, \tag{3.26}$$

where Π_i denote each dimensionless parameter. Now, in order to find those parameters, three dimensionally independent quantities as fundamental units have to be selected to scale the rest of the dimensional parameters. In this work, δ^*, τ_w and U_e are selected. The combination of these three parameters and some particular dimensional quantity must have dimension equal to one. Illustrated on an example, to find the scaling of Φ_{pp}, one must find the exponents in

$$L^{\alpha} M^{\beta} L^{-\beta} T^{-2\beta} L^{\gamma} T^{-\gamma} M^2 L^{-2} T^{-3} = 1^0. \tag{3.27}$$

This will result in a set of equations

$$\begin{aligned} L\text{:}\quad & \alpha - \beta + \gamma - 2 = 0, \\ M\text{:}\quad & \beta + 2 = 0, \\ T\text{:}\quad & -2\beta - \gamma - 3 = 0. \end{aligned} \tag{3.28}$$

By calculating the exponents α, β and γ, one can find the combination of those three parameters and Φ_{pp} and thus produce the dimensionless wall pressure spectra $\Pi_1\left(\Phi_{pp}\right) = \dfrac{\Phi_{pp} U_e}{\delta^* \tau_w^2}$. With this same process, it is possible to find all the dimensionless parameters in the Eq. (3.26). This will result in a relation

$$\frac{\Phi_{pp} U_e}{\delta^* \tau_w^2} = \phi\left(\frac{\omega \delta^*}{U_e}, \frac{\delta^*}{\tau_w}\frac{dp}{dx}, \frac{\nu}{\delta^* U_e}, \frac{\rho U_e^2}{\tau_w}, \frac{\theta}{\delta^*}, \frac{\delta}{\delta^*}, \frac{c_0}{U_e}, \Pi\right). \tag{3.29}$$

The previous equation could be directly used to design the model for wall pressure spectra. However, the Π-Theorem does not state anything about the physics in this relation. Therefore, the Eq. (3.29) can be adjusted to contain previously used (and historically proven) physical quantities. Such adjustment can be done simply by combining different dimensionless parameters in the Eq. (3.29). This modified equation is

$$\frac{\Phi_{pp} U_e}{\delta^* \tau_w^2} = \kappa\left(\frac{\omega \delta^*}{U_e}, \beta_c, R_T, C_f, H, \Delta, M, \Pi\right). \tag{3.30}$$

The final modification is made to the output of the relation, where $10\log_{10}$ is applied on the output, in order to shift the dimensionless spectra into reasonable ranges. Thus the final relation is

$$\boxed{10\log_{10}\left(\frac{\Phi_{pp}U_e}{\delta^*\tau_w^2}\right) = \zeta\left(\frac{\omega\delta^*}{U_e}, \beta_c, R_T, C_f, H, \Delta, M, \Pi\right)}. \tag{3.31}$$

Interpolation

The wall pressure spectra from all datasets are not equally distributed along the frequency in logarithmic scale. Particularly, in the low frequency region there is a lower frequency point density compared to the higher frequency region (Fig. 3.25). Also, it is not that simple to set some chosen number of frequency points for some set using only default frequency points. Therefore, the complete set of frequency points is interpolated with `SciPy.interpolate.splrep` function, where B-splines are used for the interpolation. The vector x in the interpolator is the scaled frequency $\frac{\omega\delta^*}{U_e}$ and the vector y is the scaled wall pressure spectra in logarithmic scale $10\log_{10}\left(\frac{\Phi_{pp}U_e}{\delta^*\tau_w^2}\right)$.

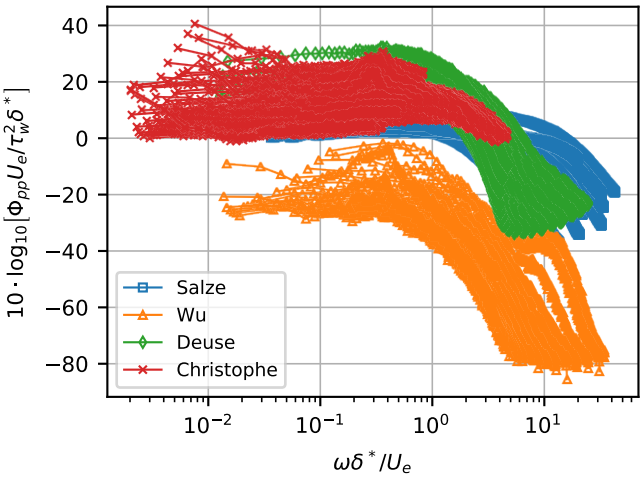


Figure 3.25: Wall pressure spectra from all datasets.

To perform the interpolation, one has to specify the vector x, where the values should be obtained. The vector is specified as logarithmically spaced between the minimum and the maximum of the scaled frequency in logarithmic scale. The number of points in this vector is then calculated as

$$\text{N. of points} = \text{Round}\left(Dens \times \left(\log_{10}\left(\frac{\omega\delta^*}{U_e}\right)_{\max} - \log_{10}\left(\frac{\omega\delta^*}{U_e}\right)_{\min}\right)\right). \tag{3.32}$$

It is some specified frequency point density multiplied by the frequency range in the logarithmic scale and rounded to the nearest integer. Different densities are used

for each author's dataset to achieve the same number of frequency points for each dataset. After several experiments, the number of frequency points for each dataset was set to 28000.

By the definition, for one wall pressure spectrum (in one boundary layer point), the spectrum should have a uniform distribution in a histogram with logarithmic bins. That is, in fact, true and it can be seen in the Fig. 3.26a. Of course, a histogram with linear bins looks accordingly (Fig. 3.26b).

The interpolated wall pressure spectra for all datasets are in the Fig. 3.27.

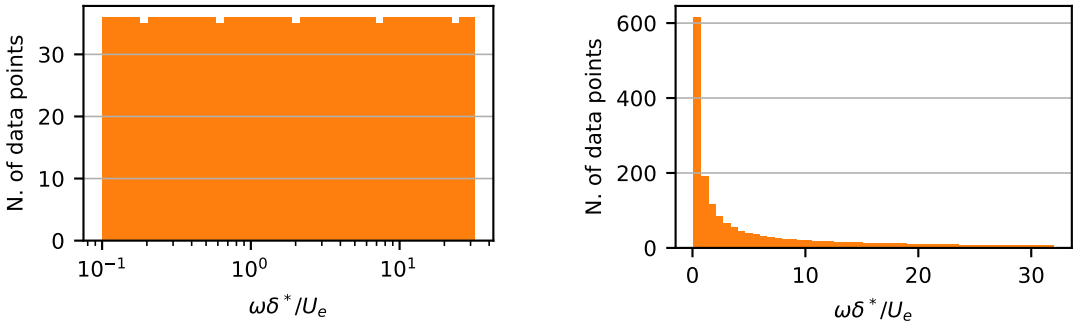


Figure 3.26: Histograms of Salze's wall pressure spectrum frequency points, APG, $U_{ref} = 38\ \mathrm{m\,s^{-1}}$.

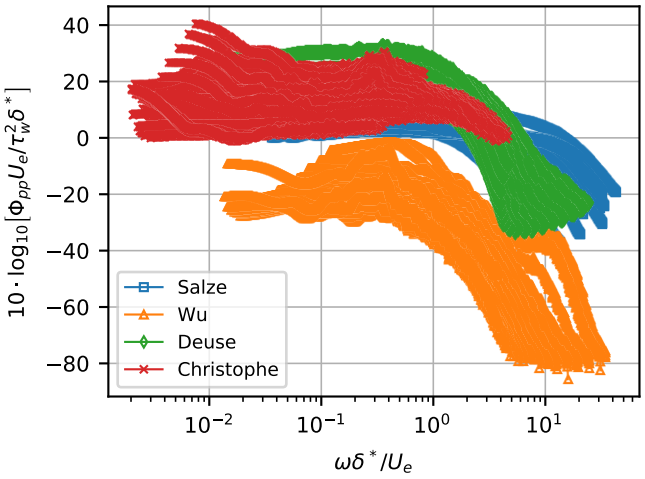


Figure 3.27: Wall pressure spectra from all datasets – interpolated.

In the previous lines, the route to the final relation (Eq. (3.31)) was described. Also, the complete wall pressure spectra dataset was interpolated. The summary of values for the final relation is in the Tab. 3.4. These values are almost ready for the wall pressure spectra modelling. In the next chapter, the final data adjustments, the neural network models, and their results will be described.

Table 3.4: The overview of the models' inputs. The references of the data are Salze *et al.*(2014) [8], Wu *et al.*(2018) [10], Deuse and Sandberg(2020) [9] and Christophe *et al.*(2014) [11]. All variables in this table are dimensionless.

Dataset	N. of boundary layer points	x_c	$10\log_{10}\left(\frac{\Phi_{pp}U_e}{\delta^*\tau_w^2}\right)$	$\frac{\omega\delta^*}{U_e}$	β_c	R_T	$C_f \times 10^3$	H	Δ	M	Π
Salze											
ZPG	6	-	−36.843 to 5.481	0.067 to 23.104	0	4.278 to 18.536	2.484 to 4.013	1.288 to 1.354	6.904 to 8.331	0.032 to 0.220	0.232 to 0.514
APG	7	-	−30.538 to 10.645	0.080 to 42.159	0.418 to 0.691	7.597 to 29.975	2.070 to 3.309	1.318 to 1.436	5.658 to 10.663	0.024 to 0.222	0.194 to 0.822
FPG	4	-	−21.780 to 6.093	0.030 to 12.642	−0.460 to −0.338	3.372 to 11.471	3.219 to 4.798	1.230 to 1.291	9.083 to 10.411	0.030 to 0.183	1×10^{-6} to 0.007
Wu											
Case 1	19	0.02 to 0.90	−85.458 to −1.680	0.014 to 34.501	−0.542 to 6.223	1.421 to 2.249	1.295 to 7.828	1.445 to 2.145	2.865 to 7.690	0.263 to 0.337	1×10^{-6} to 2.306
Deuse											
Case 1	17	0.02 to 0.80	−33.601 to 32.286	0.013 to 24.272	−0.215 to 15.590	2.196 to 3.545	1.017 to 4.368	1.393 to 2.116	3.093 to 6.484	0.209 to 0.281	0.141 to 2.265
Christophe											
Case 1	18	0.05 to 0.90	−0.546 to 27.866	0.005 to 4.683	−0.098 to 7.947	2.087 to 3.476	1.799 to 5.468	1.314 to 1.843	3.616 to 8.425	0.049 to 0.065	1×10^{-6} to 1.416
Case 2	18	0.05 to 0.90	−0.898 to 26.299	0.005 to 4.514	−0.094 to 7.431	2.104 to 3.399	1.838 to 5.559	1.312 to 1.846	3.645 to 8.456	0.049 to 0.064	1×10^{-6} to 1.402
Case 3	18	0.05 to 0.90	0.167 to 23.601	0.004 to 3.955	−0.098 to 6.325	1.915 to 3.130	2.016 to 5.796	1.328 to 1.822	3.743 to 8.810	0.049 to 0.064	1×10^{-6} to 1.283
Case 4	18	0.05 to 0.90	1.194 to 21.704	0.003 to 3.305	−0.114 to 5.246	1.650 to 2.810	2.074 to 6.292	1.358 to 1.855	3.559 to 8.671	0.050 to 0.063	1×10^{-6} to 1.333
Case 5	18	0.05 to 0.90	1.473 to 21.675	0.002 to 2.810	−0.253 to 4.502	1.302 to 2.373	2.064 to 6.801	1.416 to 1.903	3.379 to 8.142	0.050 to 0.064	1×10^{-6} to 1.432
Case 6	18	0.05 to 0.90	0.199 to 28.087	0.002 to 2.355	−0.188 to 3.787	1.027 to 2.068	2.174 to 7.217	1.513 to 1.919	3.283 to 7.655	0.051 to 0.064	1×10^{-6} to 1.421
Case 7	13	0.05 to 0.75	−0.985 to 40.345	0.002 to 1.548	−0.082 to 1.934	0.864 to 2.205	2.698 to 4.868	1.717 to 2.249	3.432 to 4.401	0.051 to 0.065	0.673 to 1.520
All	174	0.02 to 0.90	−85.458 to 40.345	0.002 to 42.159	−0.542 to 15.590	0.864 to 29.975	1.017 to 7.828	1.230 to 2.249	2.865 to 10.663	0.024 to 0.337	1×10^{-6} to 2.306

4 Results

There were two goals specified in the subsection 2.1.2: 1) To train a feedforward neural network on a broader dataset to predict wall pressure spectra (a One-to-One model); 2) To investigate the history effects and the possibilities to build a model, that would use the upstream values to predict wall pressure spectra (a Many-to-One model).

To fulfil the first goal, the preprocessed dataset from the previous chapter is used to build an FNN model. The model is built with several different architectures to find the one that predicts the spectra reasonably well and is also computationally efficient.

To fulfil the second goal, firstly, the history effects are investigated in principle. Then, to confirm those investigations, FNN models with a fixed number of airfoil positions are trained. To generalize the Many-to-One model, a CNN is trained with variable-length inputs. A One-to-One FNN model is also trained only on the airfoil data to have some relevant model for comparison. For the description of the basics of the proposed models' architectures, see section 2.2.

4.1 Feedforward Neural Network Model – One-to-One (FNNM-OtO)

4.1.1 FNNM-OtO – All Datasets

A general architecture of the Feedforward Neural Network Model – One-to-One (FNNM-OtO) is described in the Fig. 4.1. The input parameters were already chosen and preprocessed in the section 3.4. Specifically, the model takes the parameters for one boundary layer point and one scaled frequency point to predict the wall pressure spectrum at one point for one scaled frequency. The final number of examples is 111956 ($\approx$ 28000 frequncy points $\times$ 4 datasets).

The default dataset was divided into a training (60 %), validation (20 %) and test set (20 %). Each of these sets are similarly distributed along the frequency (Fig. 4.2 and Fig. 4.3c). As it was said, the number of frequency points was fixed to 28000 per dataset. This agrees with the distribution in the Fig. 4.3b. In the Fig. 4.3a, it can be seen that the adverse pressure gradient is much more represented in the input dataset than the favourable and the zero pressure gradient. The reason is that the zero pressure gradient is only represented by the Salze dataset. There are no points from other datasets that would be at least near the zero pressure gradient.

The favourable pressure gradient is also heavily represented by the Salze dataset. However, there are also some points from the airfoil datasets. It is needed to note that the pressure gradient is similarly represented in the training, validation and test set.

Finally, the input dataset and the output wall pressure spectra were normalized to the range (0; 1) to improve the training of the neural network.

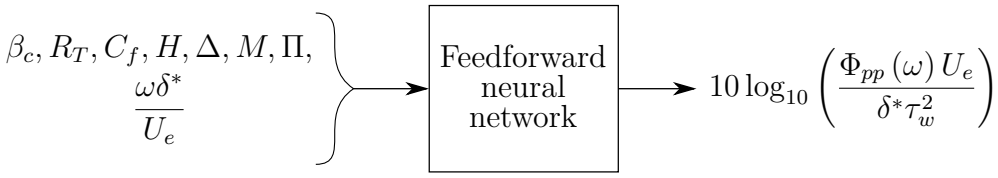


Figure 4.1: FNNM-OtO – All Datasets – the model architecture.

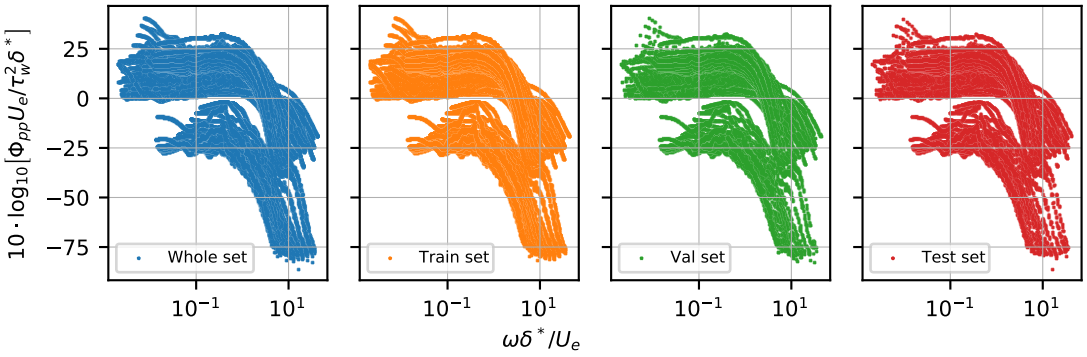
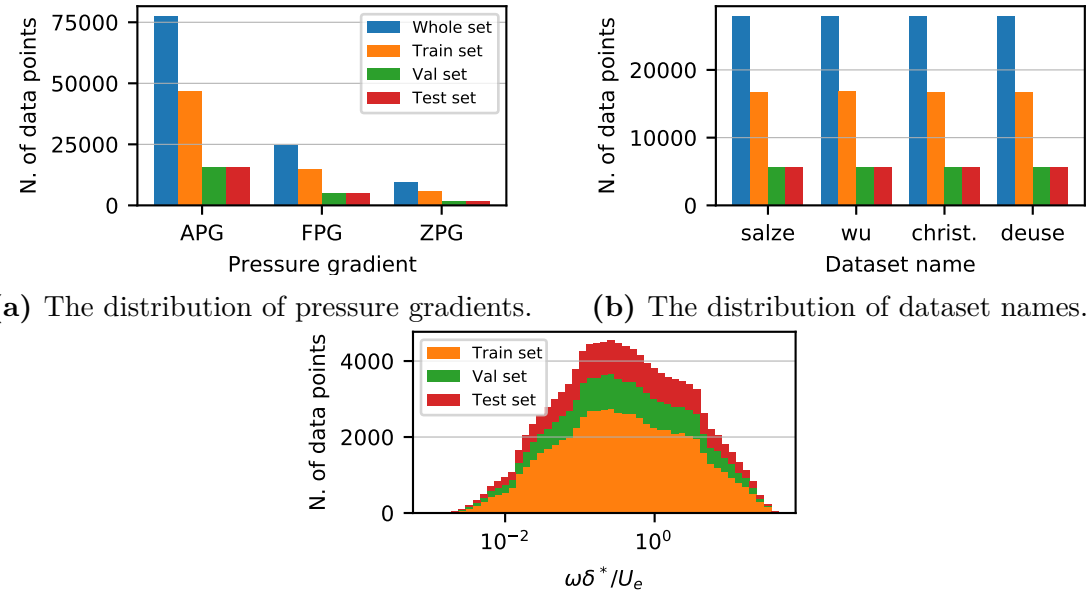


Figure 4.2: FNNM-OtO – All Datasets – the training, validation and test points.

The neural network used the Nadam optimizer, the mean squared error as the cost function, the SELU activation function for each unit and the LeCun initializer for the weights. The biases were initialized to zeros. The learning rate was scheduled so that for the first 50 epochs, the learning rate was 0.001, and after the 50 epochs, the learning rate was set to 0.0001. This approach partially damped the oscillations in training. The batch size was set to 32. With smaller batch size, the validation loss was highly oscillating.

The neural network was trained for one and two hidden layers and six different numbers of units in a hidden layer (Tab. 4.1). In the case of two hidden layers, each hidden layer has the same number of units. The training was done for 5000 epochs. In most of the cases, the 5000 epochs were enough for the losses to converge to some value. To account the non-deterministic behaviour of the training, each setup was trained with three tryouts.

The resultant training and validation loss curves for one tryout are in the Fig. 4.4. Overfitting does not occur as the validation loss is not rising. Rather it converges to some level. Therefore no regularization technique was used. The training loss curves are smooth, but the validation loss curves are oscillating. Moreover, the oscillation increases with the increasing number of parameters. The oscillations might be caused



(a) The distribution of pressure gradients.

(b) The distribution of dataset names.

(c) The distribution of scaled frequency points (stacked histogram).

Figure 4.3: FNNM-OtO – All Datasets – the distributions in the input dataset.

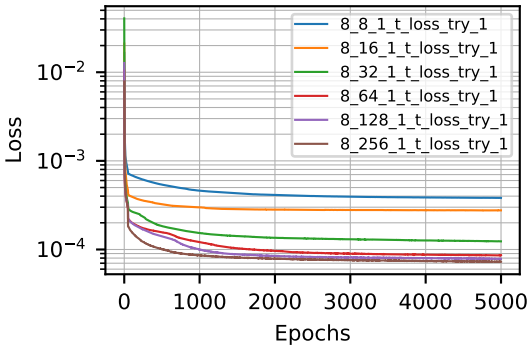
by the computation of the validation loss, as it is computed only for some small batch from the validation set at each epoch.

The last training and the last validation losses are summed up in the Fig. 4.5. Each point contains the mean value and the standard deviation denoted by the error bars. The standard deviation describes, in this case, the distribution of the last losses from the tryouts. As the number of hidden units is increasing, the losses are decreasing. Both the training losses and the validation losses are converging to some constant value. It is not surprising that this constant value is greater in the case of one hidden layer than in the case of two hidden layers since neural networks with two hidden layers contain more trainable parameters. The validation loss is rising in the case of a higher number of units. This might be a sign of overfitting when the architecture has such a number of units. However, since the values in these plots are taken as the last values of their respective loss curves, it might be caused by the increasing oscillations of the validation loss curves.

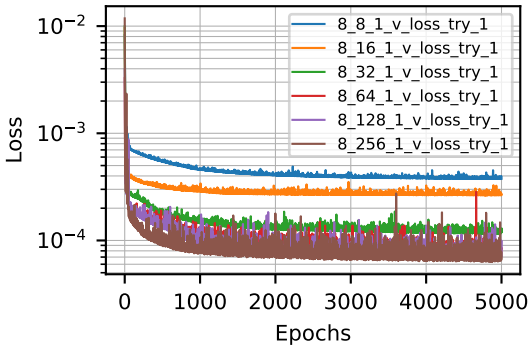
The results of the FNNM-OtO with all datasets are in the Tab. 4.1 (already illustrated in the Fig. 4.5). The number of trainable parameters (or just parameters) in this table is calculated as

$$n_{FNN} = \sum_{l=1}^{n_L-1} n_u^{(l+1)} + \sum_{l=1}^{n_L-1} \left(n_u^{(l)} n_u^{(l+1)} \right), \tag{4.1}$$

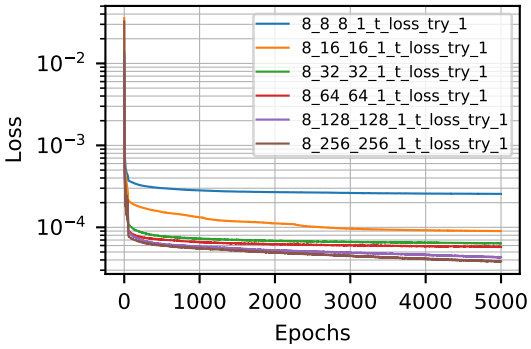
where n_{FNN} is the number of trainable parameters of the FNN, n_L is the number of layers in the FNN and $n_u^{(l)}$ is the number of units in the layer l. The number of units in the first (input) layer is equal to the number of parameters of the input dataset. Thus, in this case of the FNNM, the number of units in the first layer is eight.



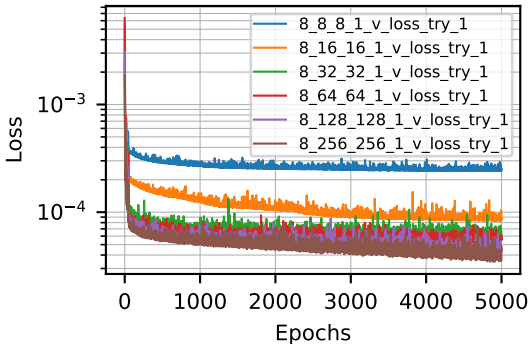
(a) 1 hid. layer, train. loss.



(b) 1 hid. layer, val. loss.

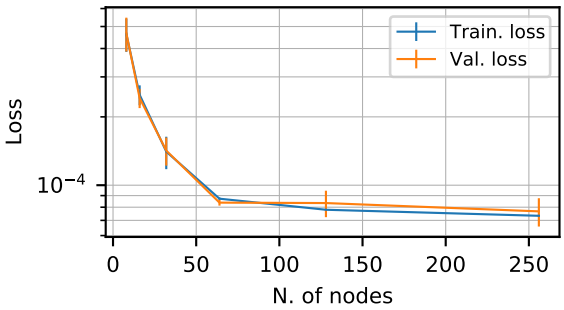


(c) 2 hid. layers, train. loss.

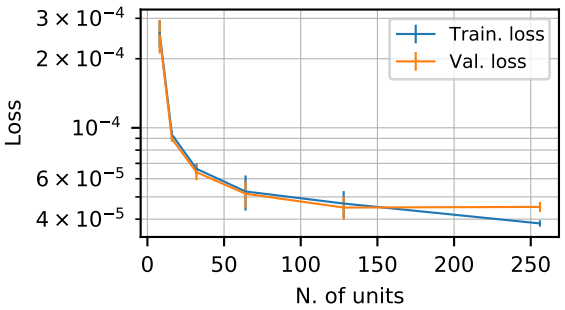


(d) 2 hid. layers, val. loss.

Figure 4.4: FNNM-OtO – All Datasets – the training and validation loss.



(a) 1 hid. layer.



(b) 2 hid. layers.

Figure 4.5: FNNM-OtO – All Datasets – the training and validation losses, mean values and standard deviation.

Table 4.1: FNNM-OtO – All Datasets – the results.

Mean val. loss $\times 10^6$	Hidden layers	Units	Parameters	Mean val. loss$^2\times$ Par. $\times 10^6$
467.394	1	8	81	17.695
242.003	1	16	161	9.429
142.067	1	32	321	6.479
83.803	1	64	641	4.502
83.439	1	128	1281	8.918
76.690	1	256	2561	15.062
252.974	2	8	153	9.791
89.505	2	16	433	3.469
64.075	2	32	1377	5.653
51.624	2	64	4801	12.795
44.941	2	128	17793	35.936
45.234	2	256	68353	139.856

Comparing the different architectures, the architecture with the least mean validation loss is the one with 2 hidden layers and 128 units in each hidden layer. The one with 256 units has a similar mean validation loss, but its validation loss curve is more oscillating, and it might be a sign of overfitting. However, these architectures are not the computationally efficient ones. To find the computationally efficient architecture that would also have low validation loss, a simple measure was introduced. The measure, which multiplies the number of trainable parameters with the validation loss squared, has the minimum when the number of parameters is low and also the validation loss is low. This approach brings out the most efficient architecture with the measure value 3.469×10^{-6}. The most efficient architecture thus has 2 hidden layers and 16 units in each hidden layer.

The predictions of the FNNM-OtO with 2 hidden layers and 16 units in each hidden layer are in the Fig. 4.6. The predictions are made for every dataset and every boundary layer point in this work. It can be observed that the predictions are good, and they reasonably fit the default data. In the Fig. 4.7 there are only the predictions for the specific points mentioned in the previous chapter. The predictions for these points are, again, pretty good.

4.1.2 FNNM-OtO – Airfoil Datasets

The proposed Many-to-One models are only valid for the airfoil datasets (Deuse, Wu, Christophe), where there are boundary layer points stacked next to each other. Therefore, the FNNM-OtO is trained only for the airfoil datasets to have a One-to-One model for comparison with the proposed Many-to-One models.

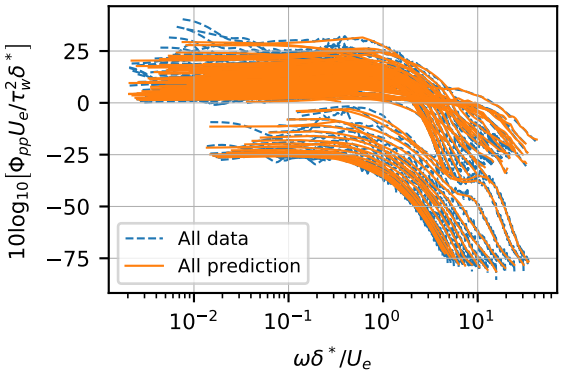


Figure 4.6: FNNM-OtO – All Datasets – the prediction of the WPS for all datasets.

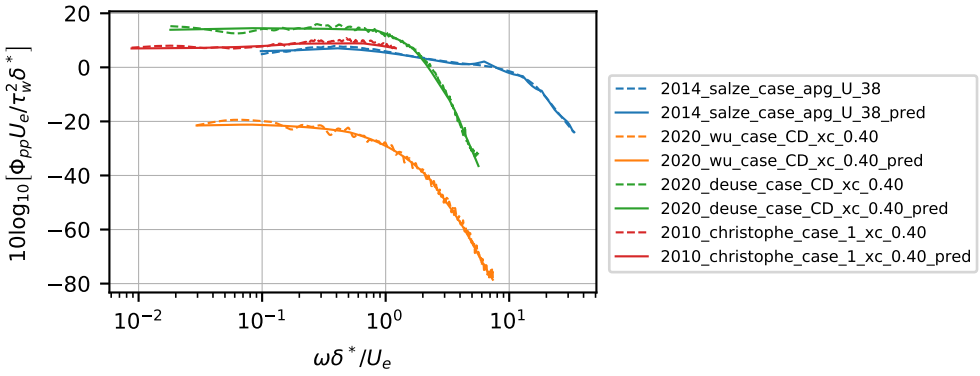


Figure 4.7: FNNM-OtO – All Datasets – the prediction of the WPS for selected points.

The architecture of the model is the same as in the Fig. 4.1. The final number of examples is, in this case, 83956 ($\approx$ 28000 frequncy points $\times$ 3 datasets).

The default dataset was divided into the training (60 %), validation (20 %) and test set (20 %) and scaled to the range (0; 1). The distribution of the frequency points for the airfoil datasets is similar as in the Fig. 4.2. The distribution of the pressure gradients is changed accordingly to the airfoil datasets (Fig. 4.8). Therefore, there are no zero pressure gradient data because the Salze dataset is not contained.

The setup of the model is the same as in the previous section. However, this model is only trained for two numbers of hidden layers and three numbers of units in a hidden layer. The reason is that this model will be used purely for comparison with the Many-to-One model, and it is not necessary to investigate the best architecture in detail.

The results of this model are in the Tab. 4.2 and in the Fig. 4.9. The architecture with the lowest mean validation loss is, again, the one with the highest number of parameters – 2 hid. layers and 128 units in each hid. layer. The best architecture according to the introduced measure is, again, the one with 2 hid. layers and 16 units in each hid. layer.

The original loss curves are in the Appendix D.

This model's resulting mean validation losses are similar to the mean validation

losses of the FNNM-OtO in the previous section. This might be surprising since this model should have more homogeneous wall pressure spectra in the input dataset.

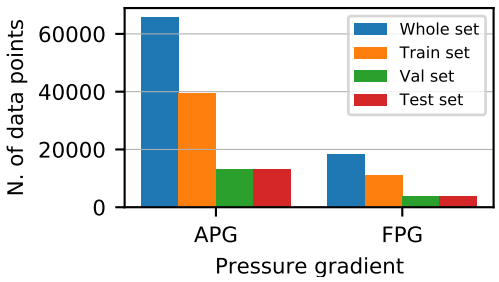
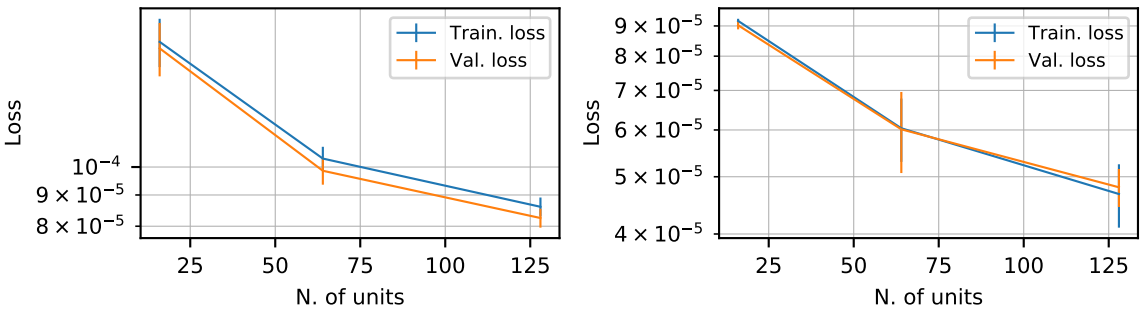


Figure 4.8: FNNM-OtO – Airfoil Datasets – the distribution of pressure gradients.

Table 4.2: FNNM-OtO – Airfoil Datasets – the results.

Mean val. loss $\times 10^6$	Hidden layers	Units	Parameters	Mean val. loss$^2\times$ Par. $\times 10^6$
156.435	1	16	161	3.940
98.604	1	64	641	6.232
82.513	1	128	1281	8.722
90.242	2	16	433	3.526
60.152	2	64	4801	17.371
47.986	2	128	17793	40.971



(a) 1 hid. layer. **(b)** 2 hid. layers.

Figure 4.9: FNNM-OtO – Airfoil Datasets – the training and validation losses, mean values and standard deviation.

4.2 History Effects

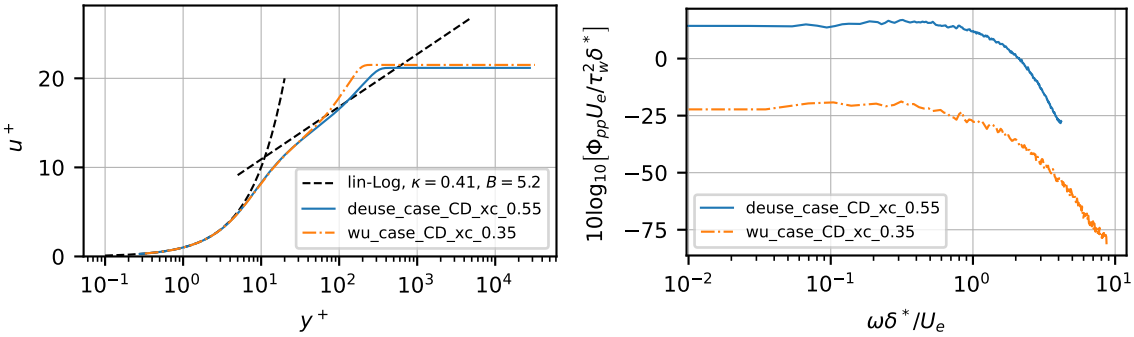
Before the work is moved onto the Many-to-One models, it would be good to investigate *a priori* the history effects resulting in the Many-to-One model.

It is assumed that the wall pressure spectrum at one point is predicted from the mean velocity profile parameters at that one point. This results in a question: *If there are any scaled mean velocity profiles that are very similar, are their respective scaled wall pressure spectra also similar, or not?* If the wall pressure spectra are similar, it will signify that the mean velocity profile at that one point is sufficient for the prediction of the wall pressure spectrum. In other words, it would mean that the local velocity profile contains everything it needs for the prediction of the spectrum. On the other hand, if the wall pressure spectra are different, it would mean that there is something else influencing the spectra. In this case, it might be, e.g. the history effect.

To answer the question, firstly, one has to find similar scaled mean velocity profiles. Since this investigation is devoted to the history effects, only the airfoil datasets are considered.

The boundary layer points on the airfoil are only available in sparse discrete points along the airfoil. To find similar mean velocity profiles, the mean squared error was used. The most similar profiles were found in the Deuse and Wu dataset. Specifically, the Deuse velocity profile is at $x_c = -0.55$, and the Wu velocity profile is at $x_c = -0.35$ on the airfoil. The mean squared error between these profiles is equal to 0.3827.

The scaled mean velocity profiles are in the Fig. 4.10 together with their respective wall pressure spectra. The velocity profiles are similar except for a small difference in the wake region. However, the wall pressure spectra are different. This may signify that this difference is caused by something else than by the mean velocity profiles solely. Since the velocity profiles are at different positions on the airfoil and thus they have a different history of the flow, the difference between the spectra might be caused by the history of the flow.



(a) The mean velocity profiles.

(b) The wall pressure spectra.

Figure 4.10: History Effects – the mean velocity profiles with their WPS.

However, as was observed in the section 4.1, the FNNM-OtO predicts the spectra

reasonably well. For the two investigated boundary layer points, the FNNM-OtO predictions are in the Fig. 4.11. The result is the same. The FNNM-OtO predicts the spectra reasonably well. But, what accounts for the difference in the input to output such different wall pressure spectra? To answer this question, the inputs into the FNNM-OtO need to be extracted. The summary of the FNNM-OtO inputs together with their normalized values and ranges is in the Tab. 4.3. Here, only the normalized values are the actual inputs into the model. In this table, the difference is calculated between the normalized values. The bold differences denote the parameters, where the difference is bigger than 0.1 and vice versa. It is believed that the parameters with the bold differences are the reason for the difference in the wall pressure spectra. In other words, the difference in these inputs might cause the difference in the spectra. This would mean that it is unnecessary to use the upstream values and use the Many-to-One model. However, this statement should be compared with the sensitivity of the model.

Unfortunately, the sensitivity analysis is not performed in this work. Therefore the investigation remains open.

So far, there is one argument pro the Many-to-One model and one argument against the Many-to-One model. The analysis of the FNNM-OtO inputs, so far, suggest that the Many-to-One model is not necessary. Still, the previous investigations should be compared with some experiments on the Many-to-One models to confirm the observations. Thus, in the following sections, the Many-to-One FNN model and the Many-to-One CNN model will be described.

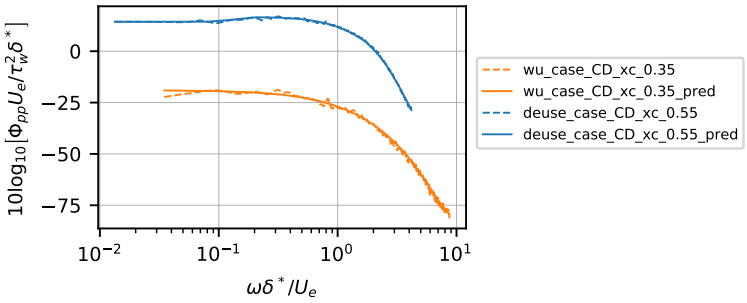


Figure 4.11: History Effects – the WPS prediction with the FNNM-OtO-Airfoil.

Table 4.3: History Effects – the inputs into the FNNM-OtO-Airfoil.

	M	β_c	R_T	C_f	H	Δ	Π
Wu xc_0.35	0.31648	0.90579	2.12382	0.00424	1.59457	5.28370	0.41979
Deuse xc_0.55	0.26790	0.69996	3.13994	0.00437	1.39296	6.42837	0.15887
Wu xc_0.35_norm	0.93003	0.08973	0.46980	0.47319	0.30136	0.40681	0.18202
Deuse xc_0.55_norm	0.76121	0.07697	0.84884	0.49204	0.08611	0.59937	0.06889
Difference	**0.16882**	0.01276	**0.37904**	0.01885	**0.21525**	**0.19256**	**0.11314**
Range Wu and Deuse	0.209 to 0.337	−0.542 to 15.590	1.421 to 3.545	0.0010 to 0.0078	1.393 to 2.145	2.865 to 7.690	1×10^{-6} to 2.306

4.3 Feedforward Neural Network Model – Many-to-One (FNNM-MtO)

4.3.1 FNNM-MtO – 2 Positions

The feedforward neural network can be trained with multiple airfoil positions as inputs. This approach is more straightforward than the CNN. However, the number of positions in the input layer has to be fixed.

The architecture of the Feedforward Neural Network Model – Many-to-One (FNNM-MtO) is in the Fig. 4.12. The model takes multiple positions on the airfoil and predicts the wall pressure spectra at one point. The rule is that the position of the spectrum and the position of the first point in the input is the same. For 2 positions, the n is 2. To give the neural network some sense of the position on the airfoil, the airfoil chord position x_c is added to the parameters.

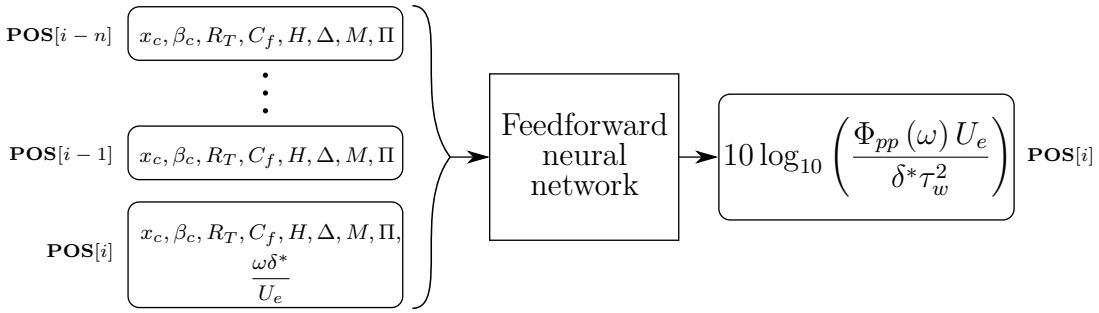


Figure 4.12: FNNM-MtO – the model architecture. i is the actual airfoil position, and n is the number of positions taken into account.

The FNNM-MtO uses only the airfoil datasets. The number of examples is reduced compared to the FNNM-OtO. As this FNNM-MtO uses 2 positions on the airfoil for the prediction, there is no possibility of predicting the spectra for the first position on the airfoil. This results in the number of examples equal to 79218.

Again, the dataset was normalized to the range (0; 1). The setup of the model is the same as in the FNNM-OtO.

The model was trained for one and two hidden layers and four different numbers of units in a hidden layer. One architecture was trained purely for comparison with the FNNM-OtO-Airfoil. This architecture has the same number of trainable parameters as the most efficient architecture from the FNNM-OtO-Airfoil, which means 430 parameters.

The results are in the Fig. 4.13 and in the Tab. 4.4. The number of trainable parameters can be calculated by the Eq. (4.1), where the input layer has a size of 17[1]. The lowest validation loss is achieved, again, with the highest number of parameters. According to the introduced measure, the most efficient architecture is the one with

[1]There are eight parameters for the first position, eight parameters for the second position and one scaled frequency. These parameters create one example together.

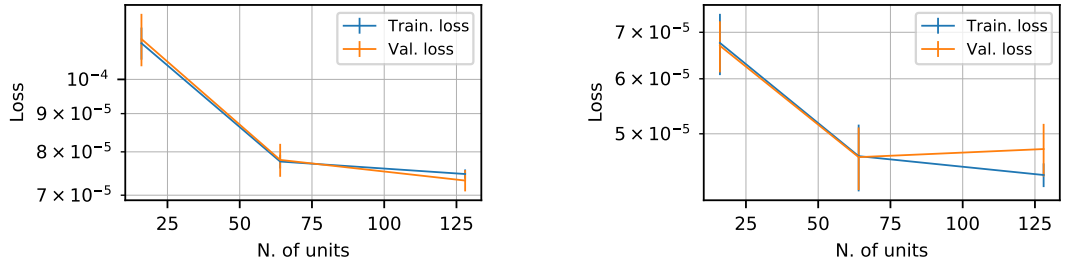
430 parameters (the one used for comparison). Compared to the FNNM-OtO with airfoil datasets, the measure is lower. Thus, the Many-to-One approach seems to be performing better. However, that might be caused by the introduction of the new parameter x_c into the input of the model.

The original loss curves are in the Appendix D.

The prediction of the wall pressure spectra is in the Fig. 4.14. As in the previous models, the prediction is very good.

Table 4.4: FNNM-MtO – 2 positions – the results.

Mean val. loss $\times 10^6$	Hidden layers	Units	Parameters	Mean val. loss$^2\times$ Par. $\times 10^6$
113.252	1	16	305	3.912
78.041	1	64	1217	7.412
73.233	1	128	2433	13.048
69.983	2	13	430	2.106
66.944	2	16	577	2.586
46.278	2	64	5377	11.516
47.537	2	128	18945	42.810



(a) 1 hid. layer. **(b)** 2 hid. layers.

Figure 4.13: FNNM-MtO – 2 positions – the training and validation losses, mean values and standard deviation.

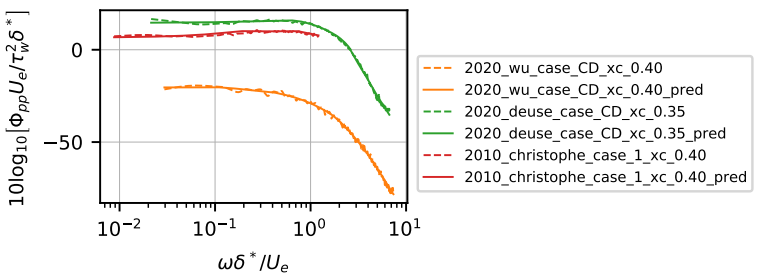


Figure 4.14: FNNM-MtO – 2 positions – the prediction of the WPS for selected points.

4.3.2 FNNM-MtO – 10 Positions

The same model as in the previous section was trained for 10 positions on the airfoil. Thus, the total number of examples is reduced to 41314.

This model aims to see how the neural network reacts to a larger number of positions. It would be a nice empirical result to see if the neural network improves the prediction and validation loss.

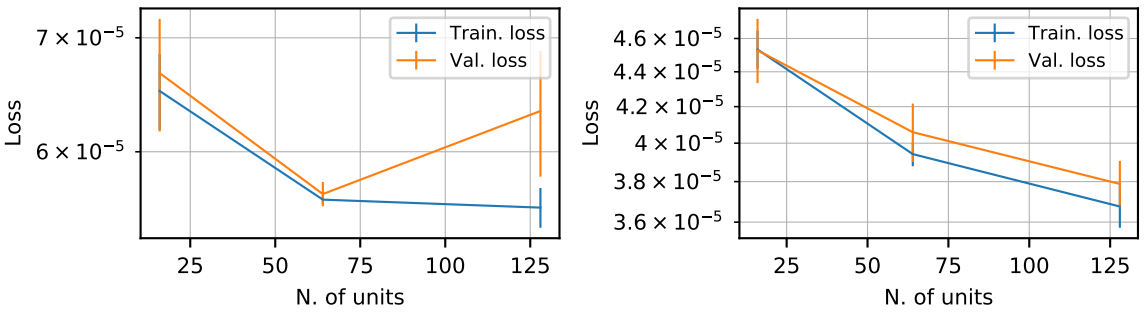
The model was trained for one and two hidden layers and four different numbers of units in a hidden layer. One architecture was, again, trained purely for comparison with the FNNM-OtO-Airfoil. This architecture has a similar number of trainable parameters as the most efficient architecture from the FNNM-OtO-Airfoil, which means 446 parameters.

The results are in the Fig. 4.15 and in the Tab. 4.5. The number of trainable parameters can be calculated by the Eq. (4.1), where the input layer has a size of 81[2]. The validation losses are, in most of the architectures, lower compared to the the 2 positional model. However, the validation loss of the comparison architecture with the 446 trainable parameters is worse. The measure's value is worse in most of the architectures, which means the 10 positional model is not that efficient. The worse result of the comparison architecture might be caused by the under-sampling in the first layer, where the input has 81 parameters, and the first hidden layer has only 5 units.

The original loss curves are in the Appendix D.

Comparing the 10 positional model with the FNNM-OtO-Airfoil, the measure's value is very similar or worse. Moreover, the 10 positional model requires more complicated preprocessing. Also, thanks to the 10 positional division of the boundary layer points, the number of examples is decreased. Therefore, the 10 positional FNNM-MtO is worse than the FNNM-OtO-Airfoil, as the 10 positional FNNM-MtO does not bring any advantage.

The predictions of the wall pressure spectra are in the Fig. 4.16. The predictions are still good.



(a) 1 hid. layer. **(b)** 2 hid. layers.

Figure 4.15: FNNM-MtO – 10 positions – the training and validation loss, mean values and standard deviation.

[2]There are ten positions with eight parameters and one scaled frequency. These parameters in all of the positions create one example together.

Table 4.5: FNNM-MtO – 10 positions – the results.

Mean val. loss $\times 10^6$	Hidden layers	Units	Parameters	Mean val. loss$^2\times$ Par. $\times 10^6$
66.742	1	16	1329	5.920
56.663	1	64	5313	17.059
63.402	1	128	10625	42.710
87.376	2	5	446	3.405
45.280	2	16	1601	3.283
40.592	2	64	9473	15.608
37.882	2	128	27137	38.943

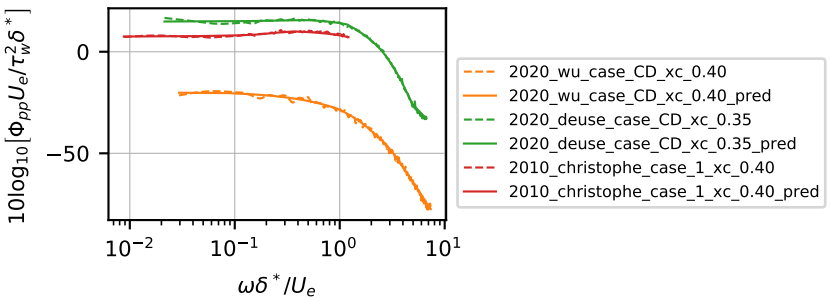


Figure 4.16: FNNM-MtO – 10 positions – the prediction of the WPS for selected points.

To find if the 10 positional FNNM-MtO uses the values upstream or not, it is possible to plot the weights in the first layer of the neural network. Such plot is in the Fig. 4.17. On the horizontal axis are the parameters. In the 10 positional FNNM-MtO, there are 81 input parameters (features). The number 81 is the result of 10 positions with 8 parameters + 1 scaled frequency. The scaled frequency is not included in the figure since it has different ranges of weights. The input parameters are stacked in a row, where the one particular parameter is grouped together with its historical (upstream values). Therefore, e.g. $x_c\,[\mathrm{POS0}]$ is next to $x_c\,[\mathrm{POS}-1]$, $x_c\,[\mathrm{POS}-2]$ and so on. The parameters in the Fig. 4.17 are in the following order: x_c, M, β_c, R_T, C_f, H, Δ, Π. On the vertical axis are the units in the following layer (16 units in this case).

If the weight is near zero, the input is not or very little used. Unfortunately, in the Fig. 4.17, there is no visible pattern. Some actual values are not used, and some upstream values are not used. Therefore, the conclusion is that the neural network uses some of the upstream values, but there is no sharp pattern in the data.

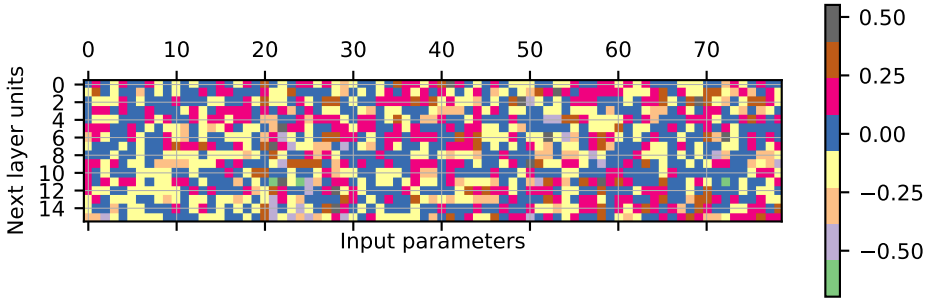


Figure 4.17: FNNM-MtO – 10 positions – the weights in the first layer.

4.4 Convolution Neural Network Model (CNNM)

A CNN is a network that extracts the inputs' features and has a high level of abstraction. The CNN is usually composed of convolution layers, pool layers and FNNs. The illustration of such architecture is in the Fig. 4.18. The global max pool layer takes the maximum value from each output vector of the convolution layer and sub-samples it. This approach can be used to fix the number of inputs into the FNN. As the length of the input to the CNN varies, the only thing that varies in the model is the length of the output vector of the convolution layer.

The CNN applied to our dataset is called the Convolution Neural Network Model (CNNM). The same applies here as for the previous Many-to-One models, which means several positions on the airfoil are used as inputs, and the output is the wall pressure spectra at one position. However, the CNNM can take variable-length inputs, which is the main difference from the previous Many-to-One models.

The input dataset from the section 3.4 was divided for several numbers of positions. Each division for some number of positions generated a different number of input examples. With an increasing number of positions on the airfoil, the number of examples decreases. In this model, the number of positions varied from 2 to 19 positions. Only the airfoil from the Wu dataset was included in the 19 positional division since it is the maximum number of boundary layer points on one airfoil. This means, in this case, that the whole airfoil was used to predict the wall pressure spectra at the trailing edge.

The complete dataset was normalized to the range (0; 1).

Two different architectures were trained. Both architectures used 16 kernels, a fixed kernel width equal to two and a stride (step) equal to one. The 16 kernels imply that the length of the global pool layer is 16. The changing part was the FNN. The first architecture used two hidden layers and 16 units in each hidden layer. The second architecture was adjusted to be comparable with the 433 parameters in the FNNM-OtO-Airfoil. The nearest was the architecture with two hidden layers and five units in each hidden layer, which created 425 parameters.

The architectures were trained with variable-length inputs. In other words, during the training, one architecture was firstly optimized to two positional data,

then three positional data, then four positional data, and so on. The architecture hyper-parameters were fixed.

The CNNM used the Adam optimizer with the default learning rate and the mean squared error as the cost function. The convolution layer used the Glorot initializer and the linear activation function. The FNN layer used the SELU activation function together with the LeCun initializer. The biases were initialized to zeros. The batch size was 32. The number of epochs for one positional dataset was set to 280. With 18 positional datasets, this results in 5040 epochs in total.

The number of trainable parameters of the CNNM can be calculated as

$$n_{CNNM} = n_k n_p k_{width} + n_k + \sum_{l=1}^{n_L-1} n_u^{(l+1)} + \sum_{l=1}^{n_L-1} \left(n_u^{(l)} n_u^{(l+1)} \right), \tag{4.2}$$

where n_{CNNM} is the number of trainable parameters in the CNNM, n_k is the number of kernels (filters), n_p is the number of input parameters in one position, k_{width} is the kernel width, n_L is the number of layers in the FNN and $n_u^{(l)}$ is the number of units in the layer l.

The results are in the Fig. 4.20 and in the Tab. 4.6. Compared to the FNNM-OtO-Airfoil, the measure of the CNNM with 425 parameters is worse than the FNNM-OtO-Airfoil with a similar number of parameters. So is the validation loss. The CNNM with 425 parameters is also worse than the FNNM-MtO with 2 and 10 positions. Even with an increased number of parameters, the CNNM has a worse value of the measure than the FNNM-OtO-Airfoil with 433 parameters. Considering the difficulty of training the CNNM, where the preprocessing of the data takes a long time, the CNNM is not performing well.

Looking at the Fig. 4.19, there is the reason for the bad performance. As the CNNM was trained with the variable-length inputs and fixed hyper-parameters, there are visible humps in the loss curves, where the input length changed. The losses are decreasing until some 4200 epochs. At this point, the dataset with 16 positions was trained. After that, the weights were scattered, and the losses increased.

The predictions of the wall pressure spectra for different positional datasets are in the Fig. 4.21. The architecture with 865 parameters performed all the predictions. The predictions are bad except for the 10 positional dataset prediction, which is the most similar to the original data. This may signify that the proper weights were still in the CNNM for a higher number of positions, but the proper weights were forgotten for the lower number of positions. This also concludes that with this setup of the CNNM, the same weights cannot be used together with, e.g. 2 positional dataset and 10 positional dataset. Increasing the number of parameters and the number of CNN layers might solve this problem.

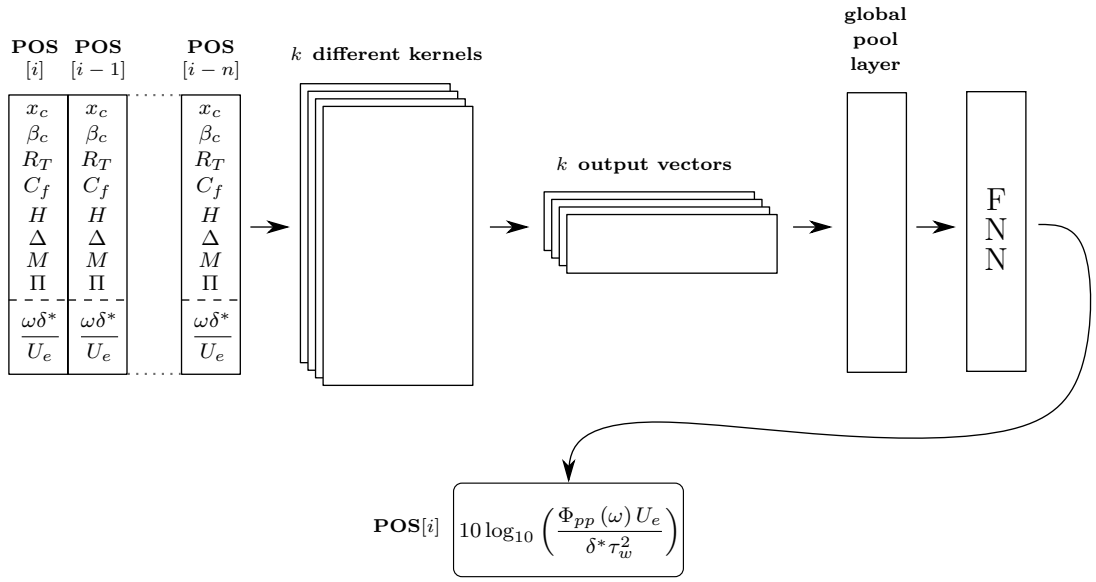
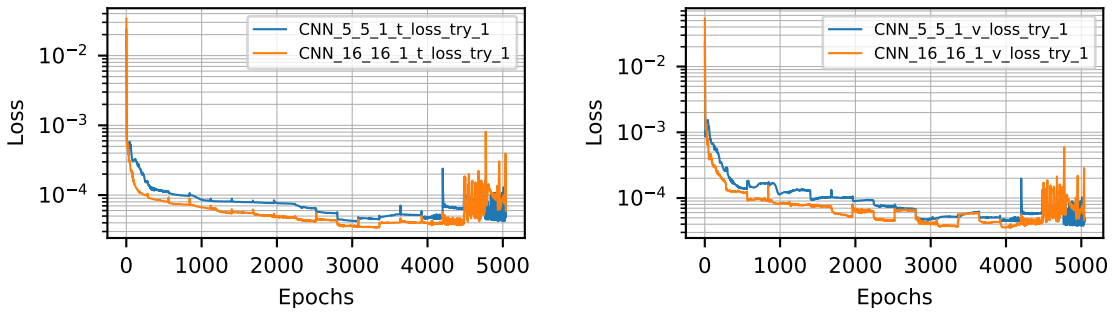


Figure 4.18: CNNM – the model architecture. i is the actual airfoil position, n is the number of positions taken into account and k is the number of kernels.



(a) Training loss. **(b)** Validation loss.

Figure 4.19: CNNM – the training and validation loss.

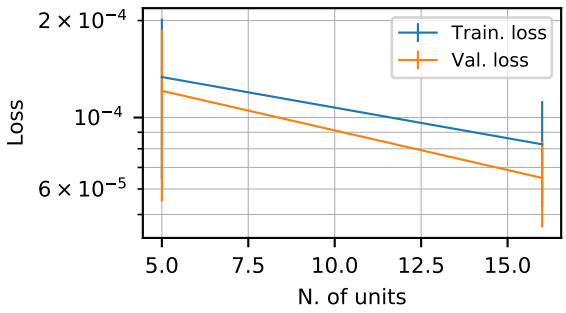


Figure 4.20: CNNM – the training and validation loss, mean values and standard deviation.

Table 4.6: CNNM – the results.

Mean val. loss $\times 10^6$	Kernels	Kernel width	FNN hidden layers	Units	Parameters	Mean val. loss$^2\times$ Par. $\times 10^6$
121.062	16	2	2	5	425	6.229
64.935	16	2	2	16	865	3.647

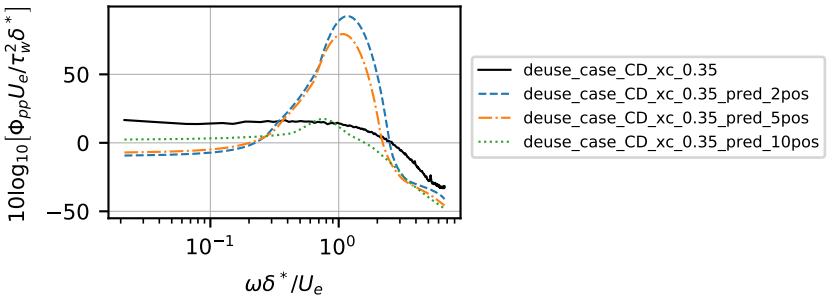


Figure 4.21: CNNM – the WPS prediction for 2, 5 and 10 positional data.

4.5 Discussion

Different neural network models were trained in the previous sections. The measure, combining the mean validation loss and the parameters, was introduced. The models were compared with this measure and with the mean validation loss. The best Many-to-One model is the feedforward neural network model with 2 positions in the input. This model outperforms the FNNM with 10 positions and the CNNM. The worst is the CNNM, which also takes a long time to train, and the preprocessing of the input dataset is more complicated than for the rest of the models. The overall results from the Many-to-One approach suggest that choosing too many positions for the model inputs will not improve the model.

Also, the One-to-One feedforward neural network models were trained. One of the models was trained on the whole dataset, including the experimental Salze dataset. This model performs well, and the fit is almost perfect. Another model was trained only on the airfoil datasets. This model was built mainly for comparison of the One-to-One approach and the Many-to-One approach.

Comparing the 2 positional FNNM-MtO and the FNNM-OtO-Airfoil, the following can be stated. The 2 positional FNNM-MtO has a lower measure and thus performs better in the case of the model with a similar number of trainable parameters (433 and 430). The difference in the mean validation loss is, in this case, 20.259×10^{-6}, and the difference in the measure is 1.420×10^{-6}. The measure in the 2 positional FNNM-MtO is also lower for some other architectures. The validation losses are lower in all of the architectures in the 2 positional FNNM-MtO. However, this might be caused by the introduction of a new parameter into the input dataset; the airfoil chord coordinate x_c.

The reason for the use of the Many-to-One approach was also investigated. The

analysis of the difference between the mean velocity profiles and their respective wall pressure spectra suggested that there is something more influencing the spectra than just the local values solely. However, the neural network trained only on the local values suggested that the local values are enough to predict the spectra. Thus, the Many-to-One approach might not be necessary. Still, a sensitivity analysis is needed in this case.

The Many-to-One approach can be compared with the One-to-One approach by how significant improvement a change in the model architecture brings. The default One-to-One model is the FNNM-OtO-Airfoil with two hidden layers and 16 units in each hidden layer. Holding the number of parameters and including one more position into the input of the default model, the measure is improved by 1.420×10^{-6} and the mean validation loss is improved by 20.259×10^{-6}. Increasing the number of units by 48 in each hidden layer of the default model, the measure is worsened by 13.845×10^{-6} and the mean validation loss is improved by 30.090×10^{-6}. By holding the number of units and adding a hidden layer to the FNNM-OtO-Airfoil with one hidden layer and 16 units in a hidden layer, the measure is improved by 0.414×10^{-6} and the mean validation loss is improved by 66.193×10^{-6}.

From the previous results, it can be concluded that adding units and hidden layers in the One-to-One model improves the mean validation loss significantly, but it costs the computational efficiency. However, adding one upstream position into the model input dataset and holding the number of parameters fixed can improve the mean validation loss (less than in the One-to-One model), but it also improves computational efficiency. Thus, changing the One-to-One model to the Many-to-One model can be, under some situations, better than modifying the One-to-One model architecture. If computational efficiency is not requested, then modifying the One-to-One model architecture is better and easier.

The One-to-One and the 2 and 10 positional Multi-to-One models predict the wall pressure spectra reasonably well. An illustration of such prediction is in the Fig. 4.22. The prediction was made with the FNNM-OtO-Airfoil. The predictions from the other models are very similar.

Comparing the semi-empirical predictions and the neural network predictions, the semi-empirical models do not fit the dataset as good as the neural network. The neural network will predict the wall pressure spectra reasonably well inside the input values ranges (see Tab. 3.4) because the neural network is partially able to interpolate. However, the problem is the extrapolation and the generalization of the purely data-driven model.

An example is in the Fig. 4.23. The prediction is made with the FNNM-OtO-Airfoil, where the Salze dataset was not introduced to the model during the training. The resultant predicted spectra in the Fig. 4.23 do not have any physical meaning, and the shape includes some humps that were never seen in the wall pressure spectra. That is one of the main disadvantages of the neural network, and the neural network should never be asked to extrapolate.

Finally, the test sets were evaluated for the best models' architectures. The FNNM-OtO with all datasets, 16 units, two hidden layers has the final loss 85.079×10^{-6}. The FNNM-OtO with airfoil datasets, 16 units, two hidden layers has the final loss

96.301 $\times 10^{-6}$. And the 2 positional FNNM-MtO, 13 units, two hidden layers has the final loss 69.409 $\times 10^{-6}$.

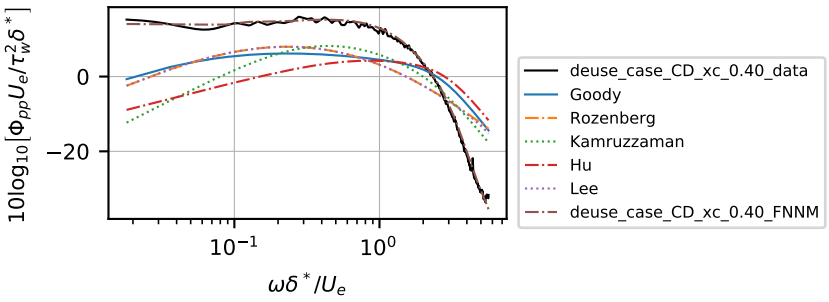


Figure 4.22: Deuse – the semi-empirical models and the FNNM-OtO-Airfoil.

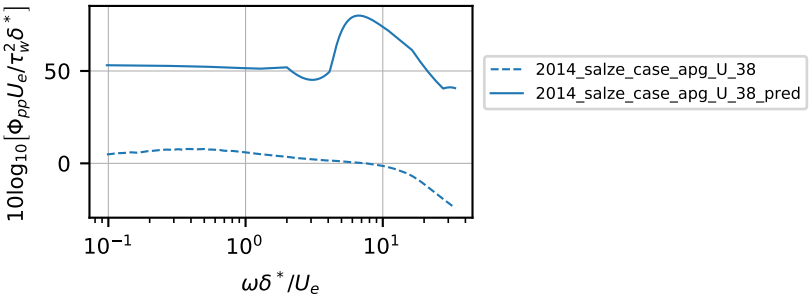


Figure 4.23: Salze – the prediction with the FNNM-OtO-Airfoil.

5 Conclusion and Future Work

In this work, some neural networks used local quantities to predict wall pressure spectra at one point. However, not all the local quantities account for the history of the flow. The main question in this work thus is: *Can we use the values upstream the point where we would like to predict the wall pressure spectra and is it going to improve the accuracy of such prediction compared to the existing models?* This question implicates the Many-to-One model, where many inputs from the boundary layer are used to predict the wall pressure spectra at one point.

Besides this main question, another goal was to train a feedforward neural network model, as a One-to-One model, on a broader dataset. This model used the local quantities to predict the spectra.

A database of relevant boundary layer parameters and wall pressure spectra was build. Data from one experimental (Salze [8]) and three numerical (Deuse and Sandberg [9], Wu *et al.* [10], Christophe *et al.* [11]) works were used for this database. All the numerical works investigated a flow around a controlled-diffusion airfoil. A self-similar mean velocity composite profile was used to fit the raw velocity profiles. The boundary layer parameters and the wall pressure spectra were then processed into a dimensionless relation, which specified the inputs and the output for the proposed models. The dimensionless input parameters thus were $\omega\delta^*/U_e$, β_c, R_T, C_f, H, Δ, M, Π and the dimensionless output was $10\log_{10}\left(\left(\Phi_{pp}U_e\right)/\left(\delta^*\tau_w^2\right)\right)$.

Two One-to-One feedforward neural network models were trained. The first was trained on the whole dataset and the second one was trained only on the airfoil dataset to have a comparison with the proposed Many-to-One models. Both of these models predicted the wall pressure spectra reasonably well. A mean validation loss and a measure were used to evaluate the models. The measure represents computational efficiency together with the validation loss. The best architecture for both One-to-One models was the one with two hidden layers and 16 units in each hidden layer.

Two Many-to-One feedforward neural network models were trained. The first used 2 airfoil positions and the second one used 10 airfoil positions. The 2 positional Many-to-One model had lower mean validation loss and lower measure than the One-to-One model for the same number of trainable parameters. The 2 positional model also had lower mean validation loss with all other architectures compared to the One-to-One model. The 10 positional Many-to-One model was computationally inefficient but had lower mean validation losses for some architectures compared to the 2 positional model. Therefore, the increase of the positions in the input did not guarantee better results. Both, the 2 positional and 10 positional model predicted

the spectra reasonably well.

A Many-to-One convolution neural network model was trained with variable-length inputs. This approach, where the architecture is fixed and the model is trained on variable-length data was not performing well. The reason is that with newly input dataset with different length, the previous weights were scattered. Increasing the number of parameters and the number of convolution layers might solve this problem. To conclude, the convolution neural network model performed worse than the other Many-to-One models.

Besides the neural networks, a theoretical investigation was done on the history effects in the prediction of wall pressure spectra. It was observed that similar mean velocity profiles had different wall pressure spectra. However, the result is that for the One-to-One model the local quantities were sufficient to predict those different wall pressure spectra.

Therefore, the answers to the main question are following:

- It is possible to use the values upstream to predict the wall pressure spectra at one point.
- Some Many-to-One neural network models can improve the prediction of the One-to-One models. But the improvement is not that significant. It should be considered if the extra preprocessing pays off the improvement of the accuracy.
- Still, both the One-to-One models and Many-to-One models predict the wall pressure spectra sufficiently well.

The neural network, as a universal approximator, is a powerful tool. As it was observed, the neural network is capable of predicting the wall pressure spectra. The development of a semi-empirical model has to go through a complicated investigation. Now, it is possible to train a neural network to obtain a similar or better fit than to the semi-empirical models. However, it is still needed to account for the interpolation and extrapolation abilities of the neural network.

The use of multiple positions to output wall pressure spectra is, in principle, a brilliant idea as the spectra are predicted from the flow, which is influenced by its history. This work shed light on the use of multiple positions with the help of feedforward and a convolution neural network. It is needed to note that only a small part of this idea was enlightened. There might be many other options where the improvement of the Many-to-One model will be more significant.

How to improve the Many-to-One model can be revealed from a deep analysis of the inputs to the neural network to see what upstream values are actually used. As the convolution neural network was very shallow in this work, there is a possibility to use some of the deep learning models that would utilize more feature extraction. A conjunction of a data-driven Many-to-One model and a physical model should be considered to enable the model to more generalize and respect the physics in the investigated cases.

Bibliography

1. BULL, M.K. Wall-Pressure Fluctuations Beneath Turbulent Boundary Layers: Some Reflections on Forty Years of Research. *Journal of Sound and Vibration*. 1996, vol. 190, no. 3, pp. 299–315. ISSN 0022-460X. Available from DOI: 10.1006/jsvi.1996.0066.
2. BLEVINS, R. D.; HOLEHOUSE, I.; WENTZ, K. R. Thermoacoustic loads and fatigue of hypersonic vehicle skin panels. *Journal of Aircraft*. 1993, vol. 30, no. 6, pp. 971–978. Available from DOI: 10.2514/3.46441.
3. MAESTRELLO, L. Radiation from and panel response to a supersonic turbulent boundary layer. *Journal of Sound and Vibration*. 1969, vol. 10, no. 2, pp. 261–295. ISSN 0022-460X. Available from DOI: 10.1016/0022-460X(69)90200-4.
4. HOWE, M. S. *Acoustics of Fluid-Structure Interactions*. Cambridge: Cambridge University Press, 1998. ISBN 9780521633208. Available from DOI: 10.1017/CBO9780511662898.
5. GOODY, Michael. Empirical Spectral Model of Surface Pressure Fluctuations. *AIAA Journal*. 2004, vol. 42, no. 9, pp. 1788–1794. ISSN 0001-1452. Available from DOI: 10.2514/1.9433.
6. ROZENBERG, Yannick; ROBERT, Gilles; MOREAU, Stéphane. Wall-Pressure Spectral Model Including the Adverse Pressure Gradient Effects. *AIAA Journal*. 2012, vol. 50, no. 10, pp. 2168–2179. ISSN 0001-1452. Available from DOI: 10.2514/1.J051500.
7. BERGHE, Jan Van den. *Inferring wall pressure spectral model using data driven approach*. Brussels, 2020. Master thesis. Vrije Universiteit Brussel.
8. SALZE, Edouard; BAILLY, Christophe; MARSDEN, Olivier; JONDEAU, Emmanuel; JUVE, Daniel. An experimental characterisation of wall pressure wavevector-frequency spectra in the presence of pressure gradients: AIAA AVIATION Forum. In: *20th AIAA/CEAS Aeroacoustics Conference*. American Institute of Aeronautics and Astronautics, 2014. ISBN 978-1-62410-285-1. Available from DOI: 10.2514/6.2014-2909.
9. DEUSE, Mathieu; SANDBERG, Richard D. Different noise generation mechanisms of a controlled diffusion aerofoil and their dependence on Mach number. *Journal of Sound and Vibration*. 2020, vol. 476, p. 115317. ISSN 0022-460X. Available from DOI: 10.1016/j.jsv.2020.115317.

10. WU, Hao; SANJOSE, Marlene; MOREAU, Stephane; SANDBERG, Richard D. Direct Numerical Simulation of the Self-Noise Radiated by the Installed Controlled-Diffusion Airfoil at Transitional Reynolds Number: AIAA AVIATION Forum. In: *2018 AIAA/CEAS Aeroacoustics Conference*. American Institute of Aeronautics and Astronautics, 2018. ISBN 978-1-62410-560-9. Available from DOI: 10.2514/6.2018-3797.

11. CHRISTOPHE, J.; MOREAU, S.; HAMMAN, C. W.; WITTEVEEN, J. A. S.; IACCARINO, G. Uncertainty Quantification for the Trailing-Edge Noise of a Controlled-Diffusion Airfoil. *AIAA Journal*. 2014, vol. 53, no. 1, pp. 42–54. ISSN 0001-1452. Available from DOI: 10.2514/1.J051696.

12. CHOU, P. Y. On velocity correlations and the solutions of the equations of turbulent fluctuation. *Quarterly of Applied Mathematics*. 1945, vol. 3, no. 1, pp. 38–54. Available from DOI: 10.1090/qam/11999.

13. POPE, S. B. *Turbulent flows*. Cambridge: Cambridge University Press, 2000. ISBN 9780521598866.

14. GRASSO, G.; JAISWAL, P.; WU, H.; MOREAU, S.; ROGER, M. Analytical models of the wall-pressure spectrum under a turbulent boundary layer with adverse pressure gradient. *Journal of Fluid Mechanics*. 2019, vol. 877, pp. 1007–1062. ISSN 0022-1120. Available from DOI: 10.1017/jfm.2019.616.

15. HWANG, Y.F.; BONNESS, William K.; HAMBRIC, Stephen A. Comparison of semi-empirical models for turbulent boundary layer wall pressure spectra. *Journal of Sound and Vibration*. 2009, vol. 319, no. 1, pp. 199–217. ISSN 0022-460X. Available from DOI: 10.1016/j.jsv.2008.06.002.

16. FARABEE, Theodore M.; CASARELLA, Mario J. Spectral features of wall pressure fluctuations beneath turbulent boundary layers. *Physics of Fluids A: Fluid Dynamics*. 1991, vol. 3, no. 10, pp. 2410–2420. ISSN 0899-8213. Available from DOI: 10.1063/1.858179.

17. SMOL'YAKOV, A. V. Calculation of the spectra of pseudosound wall-pressure fluctuations in turbulent boundary layers. *Acoustical Physics*. 2000, vol. 46, no. 3, pp. 342–347. ISSN 1562-6865. Available from DOI: 10.1134/1.29890.

18. BLAKE, William K. Turbulent boundary-layer wall-pressure fluctuations on smooth and rough walls. *Journal of Fluid Mechanics*. 1970, vol. 44, no. 4, pp. 637–660. ISSN 0022-1120. Available from DOI: 10.1017/S0022112070002069.

19. BROOKS, T.F.; HODGSON, T.H. Trailing edge noise prediction from measured surface pressures. *Journal of Sound and Vibration*. 1981, vol. 78, no. 1, pp. 69–117. ISSN 0022-460X. Available from DOI: 10.1016/S0022-460X(81)80158-7.

20. ROGER, Michel; MOREAU, Stephane. Trailing Edge Noise Measurements and Prediction for Subsonic Loaded Fan Blades: Aeroacoustics Conferences. In: *8th AIAA/CEAS Aeroacoustics Conference & Exhibit*. Breckenridge, Colorado: American Institute of Aeronautics and Astronautics, 2002. ISBN 978-1-62410-119-9. Available from DOI: 10.2514/6.2002-2460.

21. WANG, Meng; MOREAU, Stephane; IACCARINO, Gianluca; ROGER, Michel. LES Prediction of Wall-Pressure Fluctuations and Noise of a Low-Speed Airfoil. *International Journal of Aeroacoustics*. 2009, vol. 8, no. 3, pp. 177–197. ISSN 1475-472X. Available from DOI: 10.1260/147547208786940017.

22. MOREAU, Stephane; SANJOSÉ, Marlène; PEROT, Franck; KIM, Min-Suk. Direct self-noise simulation of the installed Controlled Diffusion airfoil. In: *17th AIAA/CEAS Aeroacoustics Conference (32nd AIAA Aeroacoustics Conference)*. American Institute of Aeronautics and Astronautics, 2011. Available from DOI: 10.2514/6.2011-2716.

23. SLAMA, Myriam; LEBLOND, Cédric; SAGAUT, Pierre. A Kriging-based elliptic extended anisotropic model for the turbulent boundary layer wall pressure spectrum. *Journal of Fluid Mechanics*. 2018, vol. 840, pp. 25–55. ISSN 0022-1120. Available from DOI: 10.1017/jfm.2017.810.

24. KRAICHNAN, Robert H. Pressure Fluctuations in Turbulent Flow over a Flat Plate. *The Journal of the Acoustical Society of America*. 1956, vol. 28, no. 3, pp. 378–390. ISSN 0001-4966. Available from DOI: 10.1121/1.1908336.

25. PANTON, Ronald L.; LINEBARGER, John H. Wall pressure spectra calculations for equilibrium boundary layers. *Journal of Fluid Mechanics*. 1974, vol. 65, no. 2, pp. 261–287. ISSN 0022-1120. Available from DOI: 10.1017/S0022112074001388.

26. BERTAGNOLIO, Franck; FISCHER, Andreas; ZHU, Wei Jun. Tuning of turbulent boundary layer anisotropy for improved surface pressure and trailing-edge noise modeling. *Journal of Sound and Vibration*. 2014, vol. 333, no. 3, pp. 991–1010. ISSN 0022-460X. Available from DOI: 10.1016/j.jsv.2013.10.008.

27. CHASE, D.M. Modeling the wavevector-frequency spectrum of turbulent boundary layer wall pressure. *Journal of Sound and Vibration*. 1980, vol. 70, no. 1, pp. 29–67. ISSN 0022-460X. Available from DOI: 10.1016/0022-460X(80)90553-2.

28. ZAGAROLA, Mark V.; SMITS, Alexander J. Mean-flow scaling of turbulent pipe flow. *Journal of Fluid Mechanics*. 1998, vol. 373, pp. 33–79. ISSN 0022-1120. Available from DOI: 10.1017/S0022112098002419.

29. CLAUSER, Francis H. Turbulent Boundary Layers in Adverse Pressure Gradients. *Journal of the Aeronautical Sciences*. 1954, vol. 21, no. 2, pp. 91–108. Available from DOI: 10.2514/8.2938.

30. LEE, Seongkyu. Empirical Wall-Pressure Spectral Modeling for Zero and Adverse Pressure Gradient Flows. *AIAA Journal*. 2018, vol. 56, no. 5, pp. 1818–1829. ISSN 0001-1452. Available from DOI: 10.2514/1.J056528.

31. KAMRUZZAMAN, M.; BEKIROPOULOS, D.; LUTZ, Th.; WÜRZ, W.; KRÄMER, E. A Semi-Empirical Surface Pressure Spectrum Model for Airfoil Trailing-Edge Noise Prediction. *International Journal of Aeroacoustics*. 2015, vol. 14, no. 5-6, pp. 833–882. ISSN 1475-472X. Available from DOI: 10.1260/1475-472X.14.5-6.833.

32. HU, Nan; HERR, Michaela. Characteristics of Wall Pressure Fluctuations for a Flat Plate Turbulent Boundary Layer with Pressure Gradients: Aeroacoustics Conferences. In: *22nd AIAA/CEAS Aeroacoustics Conference*. American Institute of Aeronautics and Astronautics, 2016. Available from DOI: 10.2514/6.2016-2749.

33. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning* [online]. MIT Press, 2016 [visited on 2021-05-01]. ISBN 978-0262035613. Available from: http://www.deeplearningbook.org.

34. MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*. 1943, vol. 5, no. 4, pp. 115–133. ISSN 1522-9602. Available from DOI: 10.1007/BF02478259.

35. LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*. 2015, vol. 521, no. 7553, pp. 436–444. ISSN 1476-4687. Available from DOI: 10.1038/nature14539.

36. HORNIK, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*. 1991, vol. 4, no. 2, pp. 251–257. ISSN 0893-6080. Available from DOI: 10.1016/0893-6080(91)90009-T.

37. GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. 1st ed. Boston: O'Reilly, 2017. ISBN 978-1-491-96229-9.

38. GLOROT, X.; BORDES, A.; BENGIO, Y. Deep Sparse Rectifier Neural Networks. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics: PMLR 15*. 2011, pp. 315–323. Available also from: http://proceedings.mlr.press/v15/glorot11a.html.

39. *Built-in activation functions* [online] [visited on 2021-05-01]. Available from: https://www.tensorflow.org/api_docs/python/tf/keras/activations.

40. MAAS, Andrew L.; HANNUN, Awni Y.; NG, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In: *Proceedings of the 30th International Conference on Machine Learning*. 28th ed. Atlanta, Georgia, USA, 2013. Available also from: http://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

41. KLAMBAUER, G.; UNTERTHINER, T.; MAYR, A.; HOCHREITER, S. Self-Normalizing Neural Networks. In: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. 2017. Available also from: https://arxiv.org/abs/1706.02515.

42. *Module: tf.keras.initializers* [online] [visited on 2021-05-01]. Available from: https://www.tensorflow.org/api_docs/python/tf/keras/initializers.

43. GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 249–256. Available also from: http://proceedings.mlr.press/v9/glorot10a.html.

44. HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification: Surpassing Human-Level Performance on ImageNet Classification. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034. ISSN 2380-7504. Available from DOI: 10.1109/ICCV.2015.123.

45. LECUN, Y.; BOTTOU, L.; ORR, G.; MULLER, K. Efficient BackProp. In: *Neural Networks: Tricks of the trade*. Springer, 1998. Available also from: http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf.

46. KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. In: *3rd International Conference for Learning Representations*. San Diego, 2015. Available also from: https://arxiv.org/abs/1412.6980v9.

47. DOZAT, T. *Incorporating Nesterov Momentum into Adam*. Stanford, 2015. Available also from: http://cs229.stanford.edu/proj2015/054_report.pdf.

48. CHOROMANSKA, A.; HENAFF, M.; MATHIEU, M.; AROUS, G. B.; LECUN, Y. The Loss Surfaces of Multilayer Networks. In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. 38th ed. San Diego, California, USA: PMLR, 2015, pp. 192–204. Available also from: http://proceedings.mlr.press/v38/choromanska15.html.

49. FUKUSHIMA, K. Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*. 1980, vol. 36, no. 4, pp. 193–202. Available from DOI: 10.1007/BF00344251.

50. KIRANYAZ, S.; AVCI, O.; ABDELJABER, O.; INCE, T.; GABBOUJ, M.; INMAN, D. J. 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*. 2021, vol. 151, no. 107398. ISSN 0888-3270. Available from DOI: 10.1016/j.ymssp.2020.107398.

51. MALEK, S.; MELGANI, F.; BAZI, Y. One-dimensional convolutional neural networks for spectroscopic signal regression. *Journal of Chemometrics*. 2018/05/01, vol. 32, no. 5. ISSN 0886-9383. Available from DOI: 10.1002/cem.2977.

52. ABADI, Martín et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems* [online]. 2015 [visited on 2021-04-25]. Available from: https://www.tensorflow.org/.

53. CHOLLET, François et al. *Keras* [online]. 2015 [visited on 2021-04-25]. Available from: https://keras.io/.

54. CHRISTOPHE, Julien; SCHRAM, Christophe; DOMINIQUE, Joachim. *BLtools: Turbulent Boundary Layer Processing toolbox*. Ver. 0.1. Von Karman Institute for Fluid Dynamics, 2020. In-house Python package.

55. SUTHERLAND, William. The viscosity of gases and molecular force. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*. 1893, vol. 36, no. 223, pp. 507–531. ISSN 1941-5982. Available from DOI: 10.1080/14786449308620508.

56. COLES, D. The law of the wake in the turbulent boundary layer. *Journal of Fluid Mechanics*. 1956, vol. 1, no. 2, pp. 191–226. ISSN 0022-1120. Available from DOI: 10.1017/S0022112056000135.

57. COLES, D. The young person's guide to the data. In: *AFOSR-IFP Stanford Conference: Computation of Turbulent Boundary Layers*. Stanford University, 1968.

58. CHAUHAN, Kapil; NAGIB, Hassan; MONKEWITZ, Peter. On the Composite Logarithmic Profile in Zero Pressure Gradient Turbulent Boundary Layers: Aerospace Sciences Meetings. In: *45th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 2007. Available from DOI: 10.2514/6.2007-532.

59. PRANDTL, L. 7. Bericht über Untersuchungen zur ausgebildeten Turbulenz. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*. 1925, vol. 5, no. 2, pp. 136–139. ISSN 0044-2267. Available from DOI: 10.1002/zamm.19250050212.

60. KÁRMÁN, Th. von. Mechanische Ähnlichkeit und Turbulenz. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen – Mathematisch-Physikalische Klasse*. 1930, pp. 58–76. Available also from: http://resolver.sub.uni-goettingen.de/purl?PPN252457811_1930.

61. SPALDING, D. B. A Single Formula for the "Law of the Wall". *Journal of Applied Mechanics*. 1961, vol. 28, no. 3, pp. 455–458. ISSN 0021-8936. Available from DOI: 10.1115/1.3641728.

62. MUSKER, A. J. Explicit Expression for the Smooth Wall Velocity Distribution in a Turbulent Boundary Layer. *AIAA Journal*. 1979, vol. 17, no. 6, pp. 655–657. ISSN 0001-1452. Available from DOI: 10.2514/3.61193.

63. NICKELS, T. B. Inner scaling for wall-bounded flows subject to large pressure gradients. *Journal of Fluid Mechanics*. 2004, vol. 521, pp. 217–239. ISSN 0022-1120. Available from DOI: 10.1017/S0022112004001788.

64. CHAUHAN, Kapil A.; NAGIB, Hassan M.; MONKEWITZ, Peter A. Flow Development in Boundary Layers with Pressure Gradient. In: *Advances in Turbulence XI: Springer Proceedings Physics*. Vol. 117. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 239–241. ISBN 978-3-540-72604-3. Available from DOI: 10.1007/978-3-540-72604-3_76.

65. NAGIB, Hassan; CHAUHAN, Kapil. Variations of von Kármán coefficient in canonical flows. *Physics of Fluids*. 2008, vol. 20, no. 10, p. 101518. ISSN 1070-6631. Available from DOI: 10.1063/1.3006423.

66. BUCKINGHAM, E. On Physically Similar Systems; Illustrations of the Use of Dimensional Equations. *Phys. Rev.* 1914, vol. 4, no. 4, pp. 345–376. Available from DOI: 10.1103/PhysRev.4.345.

List of Appendices

Appendix A – Pressure Fluctuations Equation

The derivation of the pressure fluctuations equation from the Navier-Stokes equation is described in the following lines.
Reynolds-Averaged continuity and Navier-Stokes equation (RANS) are:

$$\frac{\partial \langle u_j \rangle}{\partial x_j} = 0, \tag{A.1}$$

$$\frac{\partial \langle u_i \rangle}{\partial t} + \langle u_j \rangle \frac{\partial \langle u_i \rangle}{\partial x_j} = -\frac{1}{\varrho}\frac{\partial \langle p \rangle}{\partial x_i} - \frac{1}{\varrho}\frac{\partial}{\partial x_j}\left(\varrho \langle u_i' u_j' \rangle\right) + \nu \nabla^2 \langle u_i \rangle. \tag{A.2}$$

Incompressible continuity and Navier-Stokes equation are:

$$\frac{\partial u_j}{\partial x_j} = 0, \tag{A.3}$$

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\varrho}\frac{\partial p}{\partial x_i} + \nu \nabla^2 u_i. \tag{A.4}$$

Reynolds decomposition of a general variable X is:

$$X_i = \langle X_i \rangle + X_i'. \tag{A.5}$$

Reynolds decomposition of the incompressible continuity and Navier-Stokes equation is:

$$\frac{\partial \langle u_j \rangle}{\partial x_j} + \frac{\partial u_j'}{\partial x_j} = 0, \tag{A.6}$$

$$\begin{aligned} &\frac{\partial \langle u_i \rangle}{\partial t} + \frac{\partial u_i'}{\partial t} + \left(\langle u_j \rangle + u_j'\right)\left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial u_i'}{\partial x_j}\right) = \\ &= -\frac{1}{\varrho}\left(\frac{\partial \langle p \rangle}{\partial x_i} + \frac{\partial p'}{\partial x_i}\right) + \nu \nabla^2 \left(\langle u_i \rangle + u_i'\right). \end{aligned} \tag{A.7}$$

Subtracting RANS from the Eq. (A.6) and (A.7) will result in:

$$\frac{\partial u_j'}{\partial x_j} = 0, \tag{A.8}$$

$$\frac{\partial u_i'}{\partial t} + \langle u_j \rangle \frac{\partial u_i'}{\partial x_j} + u_j' \frac{\partial \langle u_i \rangle}{\partial x_j} + u_j' \frac{\partial u_i'}{\partial x_j} = -\frac{1}{\varrho}\frac{\partial p'}{\partial x_i} + \nu \nabla^2 u_i' + \frac{\partial}{\partial x_j}\langle u_i' u_j' \rangle. \tag{A.9}$$

Divergence of Eq. (A.9) is:

$$\begin{aligned} &\frac{\partial}{\partial x_i}\overset{①}{\frac{\partial u_i'}{\partial t}} + \frac{\partial}{\partial x_i}\left(\overset{②}{\langle u_j \rangle}\frac{\partial u_i'}{\partial x_j}\right) + \frac{\partial}{\partial x_i}\left(\overset{③}{u_j'}\frac{\partial \langle u_i \rangle}{\partial x_j}\right) + \frac{\partial}{\partial x_i}\left(\overset{④}{u_j'}\frac{\partial u_i'}{\partial x_j}\right) = \\ &= -\frac{1}{\varrho}\overset{⑤}{\frac{\partial^2 p'}{\partial x_i^2}} + \nu\overset{⑥}{\frac{\partial^3}{\partial x_i^3}}u_i' + \overset{⑦}{\frac{\partial^2}{\partial x_i \partial x_j}}\langle u_i' u_j' \rangle. \end{aligned} \tag{A.10}$$

Using the continuity Eq. (A.8), the 1st and the 6th term in Eq. (A.10) will be zero. The 2nd, the 3rd and the 4th term can be written together as:

$$\underset{①}{\frac{\partial\langle u_j\rangle}{\partial x_i}}\frac{\partial u_i'}{\partial x_j}+\langle u_j\rangle\underset{②}{\frac{\partial^2 u_i'}{\partial x_i\partial x_j}}+\frac{\partial u_j'}{\partial x_i}\underset{③}{\frac{\partial\langle u_i\rangle}{\partial x_j}}+u_j'\underset{④}{\frac{\partial^2\langle u_i\rangle}{\partial x_i\partial x_j}}+\frac{\partial u_j'}{\partial x_i}\underset{⑤}{\frac{\partial u_i'}{\partial x_j}}+u_j'\underset{⑥}{\frac{\partial^2 u_i'}{\partial x_i\partial x_j}}. \quad \text{(A.11)}$$

Using the continuity Eq. (A.8) and (A.1), the 2nd and the 4th term in Eq. (A.11) will be zero. However, the 6th term should also be zero, but it will be used for writing a more neat form of the pressure fluctuations equation. Therefore, the 5th and the 6th term can be rewritten, by adding a zero term, as:

$$\begin{aligned}\frac{\partial u_j'}{\partial x_i}\frac{\partial u_i'}{\partial x_j}+u_j'\frac{\partial^2 u_i'}{\partial x_i\partial x_j}&=\frac{\partial}{\partial x_i}\left(u_j'\frac{\partial u_i'}{\partial x_j}\right)=\frac{\partial}{\partial x_i}\left(u_j'\frac{\partial u_i'}{\partial x_j}+u_i'\frac{\partial u_j'}{\partial x_j}\right)=\\&=\frac{\partial}{\partial x_i}\left(\frac{\partial}{\partial x_j}\left(u_i'u_j'\right)\right)=\frac{\partial^2 u_i'u_j'}{\partial x_i\partial x_j}.\end{aligned} \quad \text{(A.12)}$$

Putting all the terms back together into Eq. (A.10) will result in:

$$\frac{\partial\langle u_j\rangle}{\partial x_i}\frac{\partial u_i'}{\partial x_j}+\frac{\partial u_j'}{\partial x_i}\frac{\partial\langle u_i\rangle}{\partial x_j}+\frac{\partial^2 u_i'u_j'}{\partial x_i\partial x_j}=-\frac{1}{\varrho}\frac{\partial^2 p'}{\partial x_i^2}+\frac{\partial^2}{\partial x_i\partial x_j}\langle u_i'u_j'\rangle. \quad \text{(A.13)}$$

Rewriting the Eq. (A.13) will introduce the final form of the pressure fluctuations equation:

$$\frac{1}{\varrho}\nabla^2 p'=-2\frac{\partial\langle u_i\rangle}{\partial x_j}\frac{\partial u_j'}{\partial x_i}-\frac{\partial^2}{\partial x_i\partial x_j}\left(u_i'u_j'-\langle u_i'u_j'\rangle\right). \quad \text{(A.14)}$$

Appendix B – Selected composite mean velocity profiles of Wu and Christophe

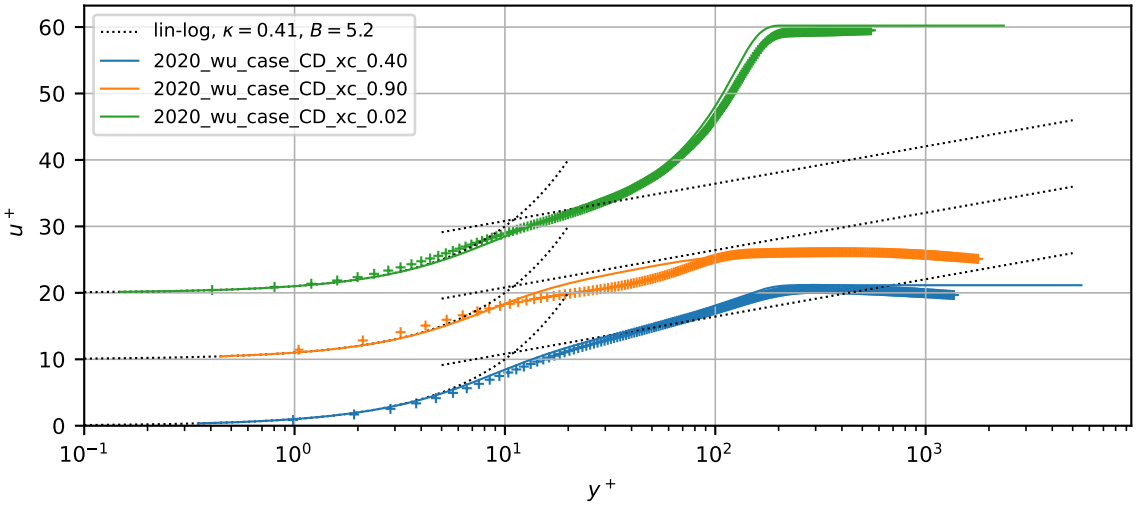


Figure B.1: Examples of the composite mean velocity profiles for Wu's dataset. Profiles are shifted by $u^+ = 10$.

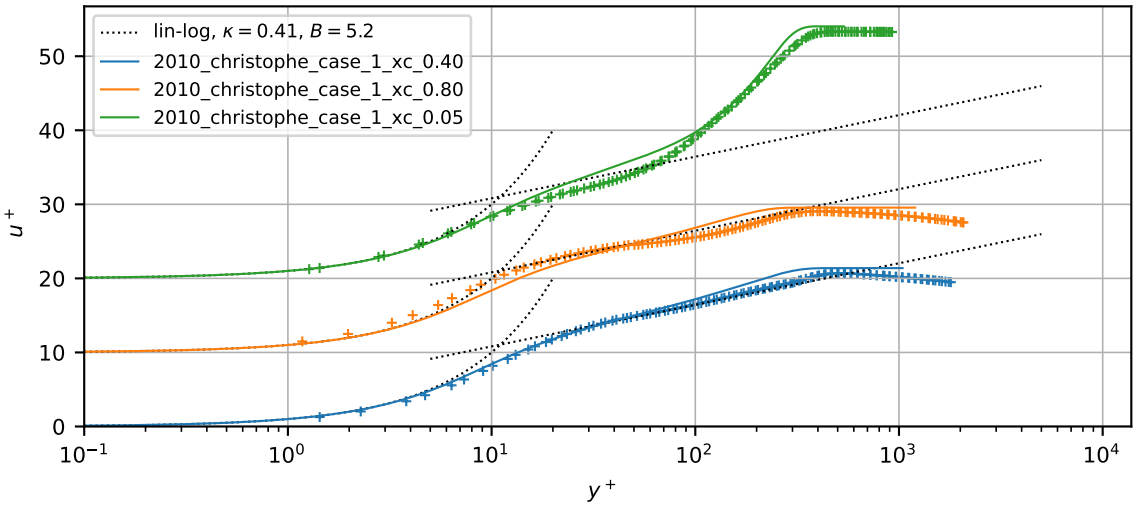


Figure B.2: Examples of the composite mean velocity profiles for Christophe's case 1 dataset. Profiles are shifted by $u^+ = 10$.

Appendix C – Selected WPS of Wu and Christophe

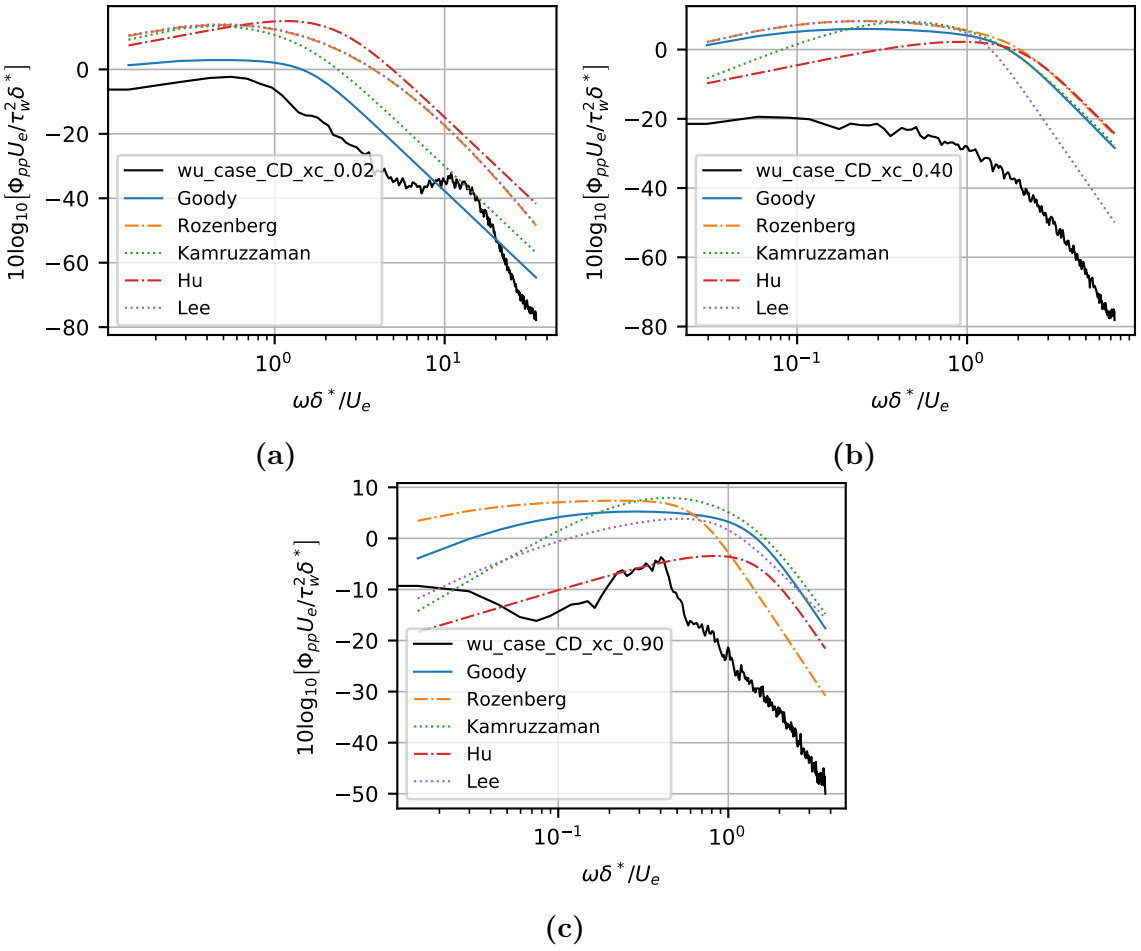


Figure C.1: Selected Wu's wall pressure spectra with their respective semi-empirical models.

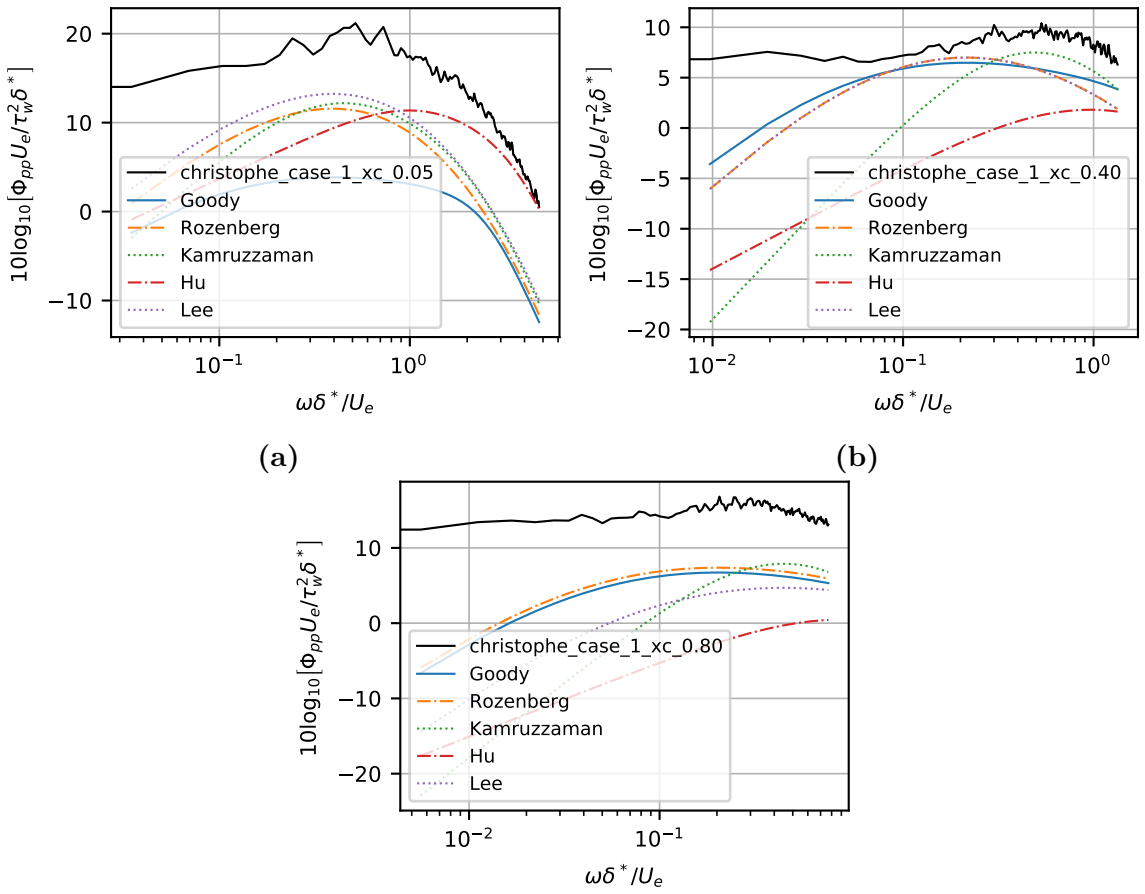
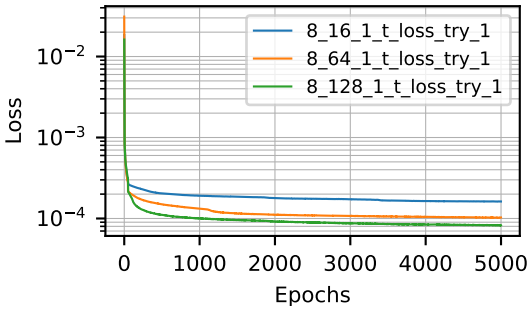
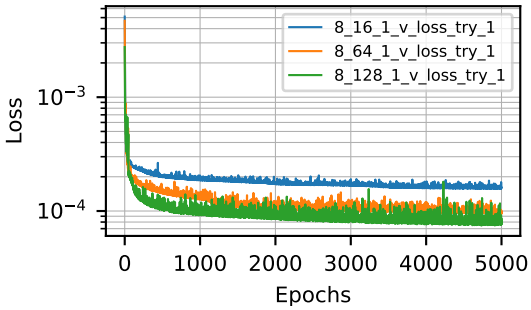


Figure C.2: Selected Christophe's wall pressure spectra with their respective semi-empirical models.

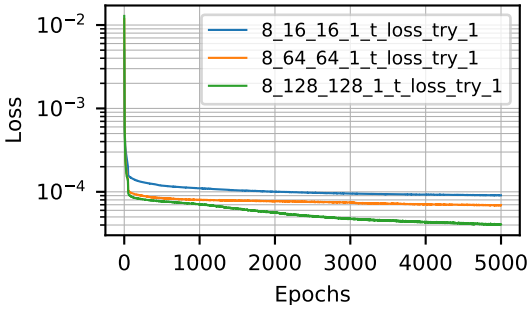
Appendix D – Training and Validation Loss Plots



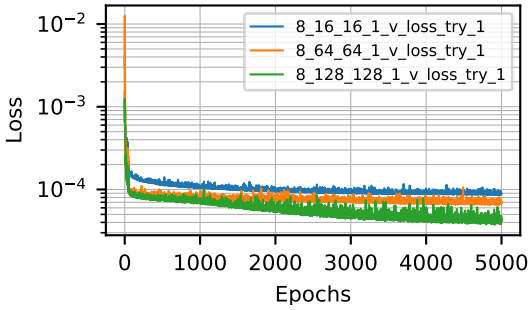
(a) 1 hid. layer, train. loss.



(b) 1 hid. layer, val. loss.

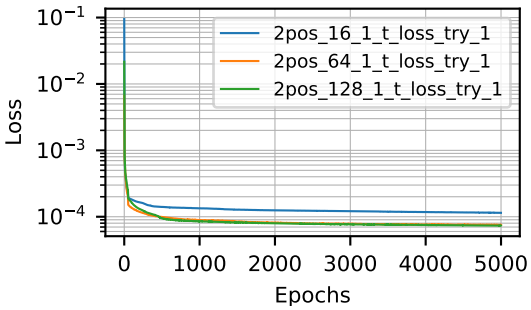


(c) 2 hid. layers, train. loss.

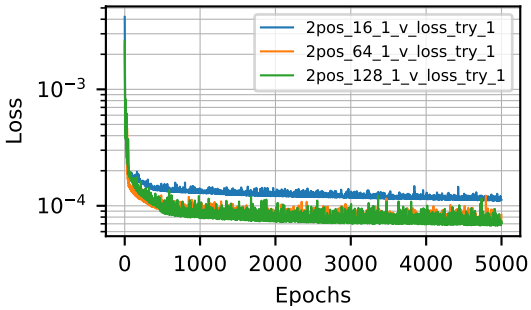


(d) 2 hid. layers, val. loss.

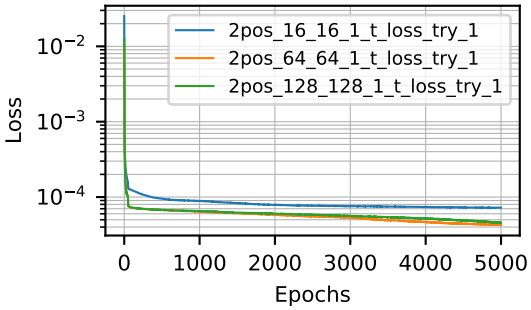
Figure D.1: FNNM-OtO – Airfoil Datasets – training and validation loss.



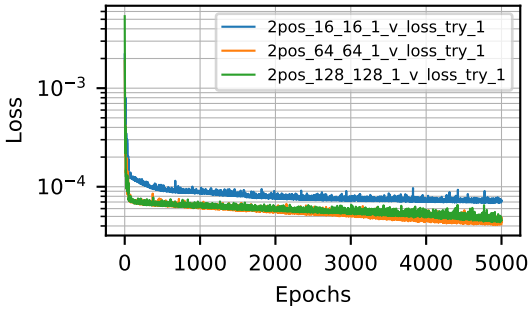
(a) 1 hid. layer, train. loss.



(b) 1 hid. layer, val. loss.

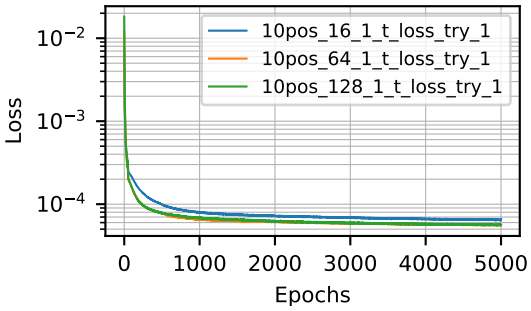


(c) 2 hid. layers, train. loss.

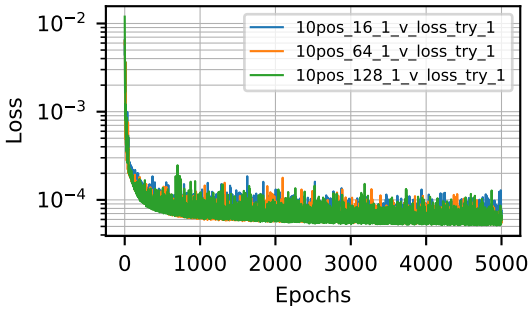


(d) 2 hid. layers, val. loss.

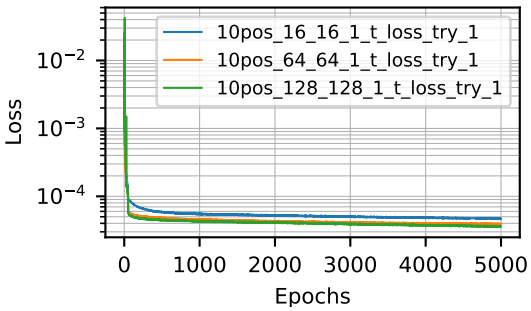
Figure D.2: FNNM-MtO – 2 positions – training and validation loss.



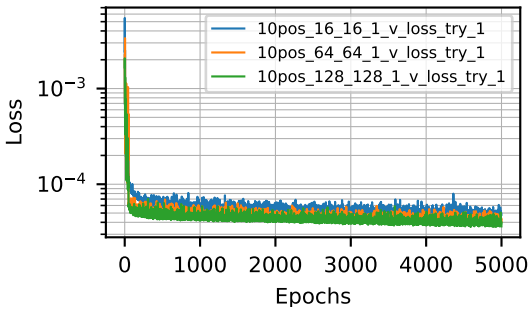
(a) 1 hid. layer, train. loss.



(b) 1 hid. layer, val. loss.



(c) 2 hid. layers, train. loss.



(d) 2 hid. layers, val. loss.

Figure D.3: FNNM-MtO – 10 positions – training and validation loss.