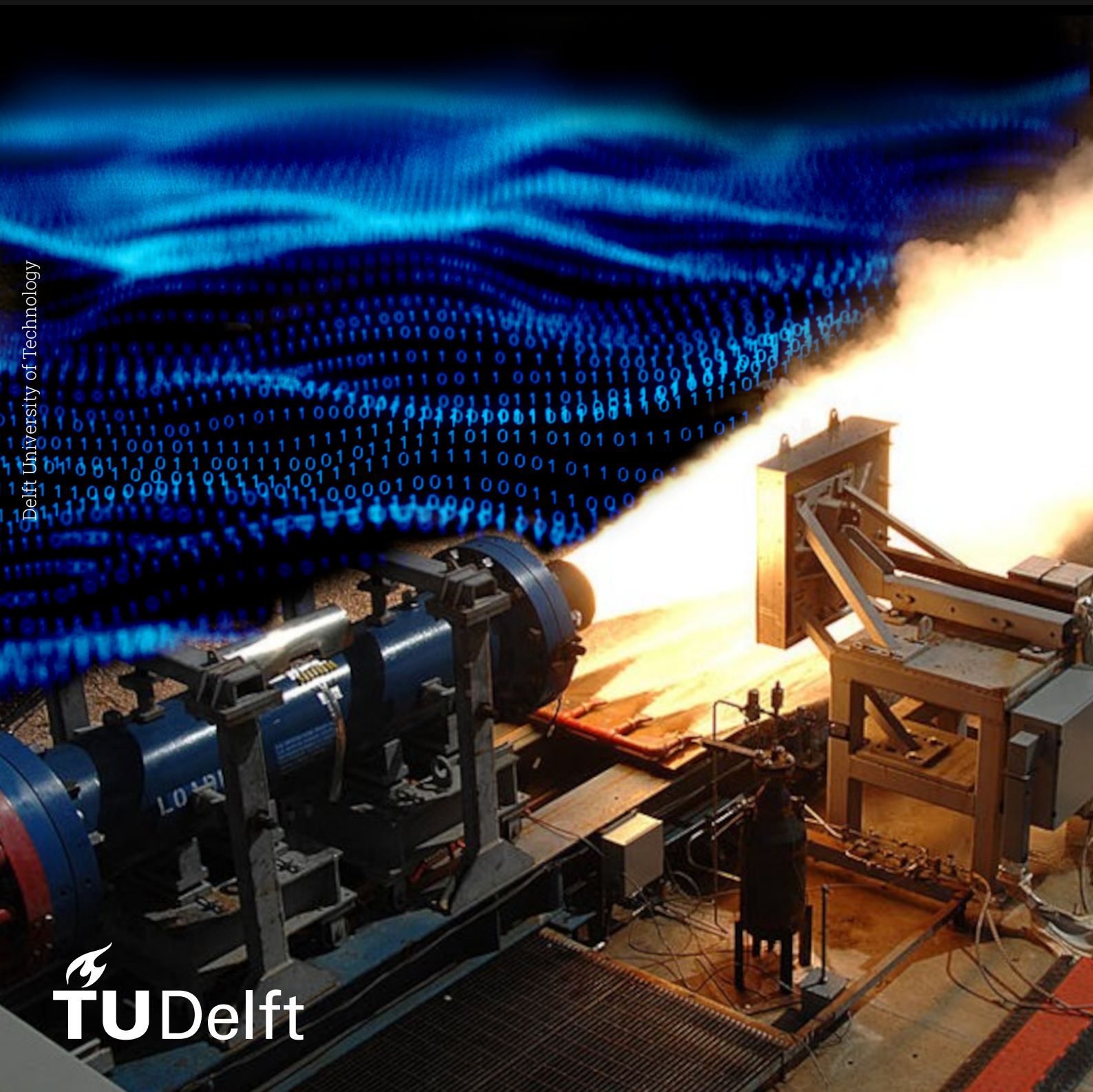
Physics-Informed Neural Networks for Aerospace Applications

Advancing Simulation and Surrogate Modelling

T. D. Hirs



Delft University of Technology

TU Delft

(This page is intentionally left blank)

Physics-Informed Neural Networks for Aerospace Applications

Advancing Simulation and Surrogate Modelling

by

T. D. Hirs

to obtain the degree of Master of Science
at the Delft University of Technology - Faculty of Aerospace Engineering,
to be defended publicly on Wednesday June 18, 2025 at 10:00 AM.

Student number:	5027578	
Project duration:	September 2, 2024 – June 18, 2025	
Thesis committee:	Dr. Ir. R. Sabzevari,	TU Delft, Supervisor
	Ir. M.C. Naeije	TU Delft, Chair
	Dr. Ir. P.P. Sundaramoorthy	TU Delft, Examiner
	Dr. A. Cervone	TU Delft, Additional member

Cover Image: Adapted from [85, 76]

Disclaimer: Portions of the LaTeX code used in this thesis, specifically for formatting elements such as tables, image positioning, layout adjustments, and resolving technical errors, were generated with the assistance of OpenAI's ChatGPT. This assistance was limited strictly to formatting and technical support; no content, data interpretation, or research findings were generated or influenced by AI.

An electronic version of this thesis is available at: [https://repository.tudelft.nl].

TU Delft

Preface

Six years after starting my Bachelor's in Aerospace Engineering at Delft University of Technology, I am presenting my Master's thesis. This project posed many challenges, but in the end, I am grateful that everything came together. Learning a new skill is never easy, and undertaking a thesis in a subject I had not previously mastered was a significant risk. Fortunately, I had supervisors who believed in me and were willing to take that risk alongside me, diving into this project and guiding me through its many phases and challenges.

I would like to thank Professor Sabzevari and Professor Cervone for their invaluable guidance and for all the tips and tricks that helped make this project a success. Moreover, this work heavily depended on access to data, which is often difficult to obtain from public sources. I am extremely grateful to DARE and Professor Jyoti for their willingness to share experimental data from the *Exercise Thermal Rocket Propulsion* and *Space Engineering Practical* courses.

I also wish to express my appreciation to the academic counsellors of the Aerospace Engineering department and the Export Control and Knowledge Security Office at Delft University of Technology. Their support and advice were invaluable during the more difficult moments of this project. While many situations can be planned for, I could never have anticipated some of the challenges I encountered. Without the help of these various departments, this project might never have taken off.

Furthermore, I would like to thank the friends I made along the way for their incredible support and belief in me. Without them, I would not have come this far. I hope our friendships will last a lifetime.

Lastly, I would like to thank my family, especially my parents, for their unwavering support and willingness to help during this project. Life is rarely straightforward, and during moments when all seemed lost, you were always there to listen without judgment and offer guidance. No matter the path I chose, you stood by me, and I hope that will always remain true in my future endeavours.

T. D. Hirs
Delft, May 2025

Summary

The recent advancements in artificial intelligence (AI), driven by increasing computational power, have unlocked a wide range of possibilities in the aerospace domain [45]. In this thesis, the potential of PINNs to enhance simulation and surrogate modelling in aerospace applications has been explored, making use of a specifically developed PINN software tool in the Python coding language, making use of the PyTorch framework. This specific tool can generate and adapt Feed Forward Neural Network (FNN) architectures, having the ability to control the cost function by extending it with additional loss functions and the ability to add separate individual network parameters for the generation of Inverse PINNs (I-PINN). Using this tool, three main use cases were investigated: the cooling down of objects modelling, Computational Fluid Dynamics (CFD) simulations, and Solid Rocket Motor (SRM) modelling. Each use case was evaluated for the accuracy, computational efficiency, and data efficiency of PINNs in comparison to a Numerical and purely Data-Driven method, as well as their surrogate modelling capability.

Examining the first use case, it was found that the cooling down of objects had a decrease in average RMSE of 7 [$^{\circ}$C] and a decrease in average maximum error of 9 [$^{\circ}$C] compared to the Numerical model, while requiring fewer data points compared to a purely Data-Driven method for similar accuracy levels. However, the training time of PINNs is about 2-3 times higher compared to a purely Data-Driven model, with a similar order of execution time compared to a Numerical model, showing that while there is an improvement in accuracy and data efficiency, there is a decrease in computational efficiency. Furthermore, it was found that a PINN and especially I-PINNs are suitable to be used as surrogate modelling methods, showing an increased accuracy in average RMSE and average maximum error compared to purely Data-Driven and Numerical methods when leaving out data groups during the training phase.

Examining the second use case, the ability of PINNs to compute derivatives analytically rather than numerically was examined by generating CFD simulations around various geometries and for various flow conditions. The largest benefit of this method lies in the fact that there is no longer a need to generate a computational mesh, decreasing pre-processing time. It was found that for low Mach numbers and Reynolds numbers, the PINN can provide comparable results to a traditional numerical model using a relatively shallow NN compared to previous research, having differences within 10% for the maximum and minimum values. However, higher Mach and Reynolds number flows require a large computational time and result in unsatisfactory performance. When incorporating experimental data in the form of measured pressure at specified locations on a NACA 0012 airfoil, it was found that, while observing some slight improvements, the PINN was not able to provide comparable results to a traditional Numerical method when validated against the section lift coefficient of a NACA 0012 airfoil.

Lastly, the third use case, the modelling of SRM performance, was examined. This use case is similar to the first use case, but with an increased difficulty level in terms of the incorporated physical model and the number of inputs and outputs used. While the PINN did show an improvement in terms of accuracy compared to a Numerical and purely Data-Driven model, the PINN method was not able to act as a surrogate model. Furthermore, it was shown that the PINN has an increased data efficiency of 50% compared to the purely Data-Driven method when reaching similar accuracy levels. However, the PINN showed an increase in computational time of 25-125 times compared to a purely Data-Driven method, while also showing an increased execution time compared to the Numerical models, meaning that the PINN has a decreased computational efficiency.

Concluding, this research showed that a PINN can provide improvements in accuracy and data-efficiency compared to a Numerical and purely Data-Driven method. However, a PINN does show a decrease in computational efficiency, having higher training and execution times compared to a Numerical and purely Data-Driven method. While this research showed promising results, it can be seen as a good starting point for future research. It is recommended to improve the developed software tool to be able to be used on dedicated hardware, such as CPUs. When the tool is optimised, it is recommended to examine the second and third use cases further, decreasing computational time and opening up the possibility for deeper NN architectures. Lastly, it is recommended to identify more use cases for PINNs by domain-specific experts, leading to a general conclusion related to PINN simulations and surrogate modelling capabilities.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
1-D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
A/D	Analog-to-Digital
Adam	Adaptive Momentum Estimation
AI	Artificial Intelligence
Avg	Average
BEM	Ballistic Evaluation Motor
CDOB	Cooling Down of OBjects
CEA	Chemical Equilibrium with Applications
CFL3D	Computational Fluids Laboratory 3-Dimensional
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
CSV	Comma-Separated Value
DAG	Directed Acyclic Graph
DARE	Delft Aerospace Rocket Engineering
DNN	Deep Neural Network
ELU	Exponential Linear Unit
EXP	Experimental
E.g.	exempli gratia
FDM	Finite Difference Method
FEM	Finite Element Method
FFT	Fast Fourier Transform
FNN	Feed-forward Neural Network
FVM	Finite Volume Method
I-PINN	Inverse Physics-Informed Neural Network
ISA	International Standard Atmosphere
KN	Potassium Nitrate
KNSB	Potassium Nitrate (KN) Sorbitol
L1	Mean Average Error
L2	Mean Squared Error
LC	Load cell
LOO-CV	Leave-One-Out Cross Validation
LSTM	Long-Short Term Memory
lr	Learning Rate
MAE	Mean Average Error
MAPE	Mean Absolute Percentage Error
Max	Maximum
MDT	Measurements During Test
ME	Mean Error
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NACA	National Advisory Committee for Aeronautics
NASA	National Aeronautics and Space Administration

Abbreviation	Definition
NN	Neural Network
O/F	Oxidizer-to-fuel ratio
OMAT	Object Measurements After test
OMBT	Object Measurements Before Test
OpenFOAM	Open Field Operation And Manipulation
PEFT	Parameter Efficient Fine Tuning
PINN	Physics-Informed Neural Network
PS	Pressure Sensor
PTG	Primary Test Goal(s)
RANS	Reynolds-averaged Navier-Stokes
ReLu	Rectified Linear Unit
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
simpleFoam	Semi-Implicit Method for Pressure Linked Equations
SRM	Solid Rocket Motor
STG	Secondary Test Goal(s)
Tanh	Hyperbolic Tangent
tdms	Technical Data Management System
TE	Test Equipment
USU	Utah State University
Var	Variance
VB	Voice Box

Symbols

Symbol	Definition	Unit
A	Surface area	[m^2]
A_{burn}	Burn area	[m^2]
A_e	Nozzle exit area	[m^2]
$A_{surface}$	Object surface area	[m^2]
A_t	Nozzle throat area	[m^2]
B	Bias of artificial neuron	[-]
B_i	Network Bias	[-]
B_i	Biot number	[-]
C_d	Discharge coefficient	[-]
$C_{f,ideal}$	Theoretical thrust coefficient	[-]
$C_{f,real}$	Measured thrust coefficient	[-]
C_f	Thrust coefficient	[-]
C_l	Section lift coefficient	[-]
C_{p_l}	Pressure coefficient of the lower part of the airfoil	[-]
C_{p_u}	Pressure coefficient of the upper part of the airfoil	[-]
C_p	Pressure coefficient	[-]
D	Diameter	[mm]
D_0	Propellant grain outer diameter	[m]
D_{cc}	Diameter of combustion chamber	[m]
D_{grain_i}	Inner grain diameter	[m]
D_{grain_o}	Outer grain diameter	[m]
D_{nozzle_e}	Nozzle exit diameter	[m]
D_{nozzle_t}	Nozzle throat diameter	[m]
E_{sensor}	Individual sensor error	[-]
E_{total}	Total error	[-]
E_t	Total energy of the system	[J]
F	Thrust force	[N]

Symbol	Definition	Unit
$F(t)$	Thrust force over time	[N]
F_{ideal}	Theoretical thrust force	[N]
F_{real}	Measured thrust force	[N]
H_0	Null hypothesis	[-]
$I_{measured}$	Measured current	[A]
$I_{sp,\text{ideal}}$	Theoretical specific impulse	[s]
$I_{sp,\text{real}}$	Measured specific impulse	[s]
I_{sp}	Specific Impulse	[s]
J	Cost function	[-]
K	Empirical scaling factor	[-]
L	Length	[m]
L	Length	[mm]
L	Characteristic length	[m]
L'	Lift force per unit span of wing generated by the airfoil	[N]
$L_{Boundary}$	Boundary loss term	[-]
$L_{Physics}$	Physics loss term	[-]
L_{cc}	Length of combustion chamber	[m]
$L_{chamber}$	Combustion chamber length	[m]
L_c	Characteristic length	[m]
L_{data}	Data loss term	[-]
L_{eff}	Effective length	[mm]
L_{grain}	Length of the propellant grain	[m]
L_{port}	Length of the port of the grain	[m]
M	Number of used data points	[-]
M	Mach number	[-]
M_{crit}	Threshold Mach number	[-]
M_c	Total mass of gasses in the combustion chamber	[kg]
M_{port}	Critical Mach number of the grain port	[-]
N	Number of used data points	[-]
N	Number of propellant grains	[-]
N_{grains}	Number of propellant grains	[-]
P	Static pressure in the domain	[pa]
P	Pressure	[pa]
P_0	Chamber pressure	[pa]
Q	Heat flux out of a body	[W/m^2]
R	Specific gas constant	[J/kgK]
R_a	Universal gas constant	[J/molK]
R_e	Reynolds number	[-]
T	Temperature	[K]
T	Surface temperature of object	[K]
T	Temperature of the fluid	[K]
T	Static air temperature	[K]
T_0	Starting temperature of object	[°C]
T_0	Adiabatic flame temperature	[K]
T_c	Temperature inside the combustion chamber	[K]
T_{env}	Temperature of the environment	[K]
T_{env}	Temperature of the environment	[°C]
$T_{i,0}$	Reference temperature of the grain	[K]
T_i	Current temperature of the grain	[K]
U	Internal energy	[J]
U_∞	Horizontal, upstream velocity of the fluid upstream of the computational domain	[m/s]
U_∞	Inflow velocity	[m/s]
U_e	Nozzle exit exhaust velocity	[m/s]

Symbol	Definition	Unit
V	Velocity	[m/s]
V	Velocity of the object in air or the air to the object	[m/s]
V	Characteristic velocity of the fluid	[m/s]
V	Velocity of the airflow	[m/s]
V_∞	Velocity of the freestream airflow	[m/s]
V_{body}	Volume of the objects body	[m^3]
$V_{meausred}$	Measured voltage	[V]
W	Network weight	[-]
W_g	Molar mass of the combustion gasses	[g/mol]
W_i	Network weight	[-]
X_s	Input data	[-]
Y	Output of artificial neuron	[-]
Y_s	Output prediction	[-]
$\vec{W}$	Weight of artificial neuron	[-]
$\vec{X}$	Inputs of artificial neuron	[-]
$\dot{P}_0$	Change in chamber pressure over time	[pa/s]
$\dot{T}_{object}$	Change in object's surface temperature over time	[K]
$\dot{m}$	Mass flow rate	[kg/s]
$\dot{m}(t)$	Mass flow rate over time	[kg/s]
$\dot{m}_{Nozzle}$	Mass flow rate from the combustion chamber through the nozzle	[kg/s]
$\dot{m}_{propellant}$	Mass flow rate from the propellant to the combustion chamber	[kg/s]
$\dot{r}$	regression rate	[mm/s]
$\hat{f}(x)$	Predicted output value	[-]
$\hat{y}$	Predicted output value	[-]
$\mathbf{f}$	External forces applied to a fluid	[N]
$\mathbf{q}$	Heat flux vector	[W/m^2]
$\mathbf{u}$	Velocity vector	[m/s]
$\mathcal{O}(h)$	First-order of discretization error	[-]
$\mathcal{O}(h^2)$	Second-order of discretization error	[-]
a	activation function	[-]
a	Burning rate coefficient	[mm/s - Pa^n]
a_0	Burning rate coefficient for set temperature	[mm/s - Pa^n]
c	Chord length of the airfoil	[m]
c^*_{ideal}	Theoretical characteristic velocity	[m/s]
c^*_{real}	Measured characteristic velocity	[m/s]
c_p	Specific heat capacity	[J/kgK]
d_0	Propellant grain inner diameter	[m]
$d_{chamber}$	Chamber diameter	[m]
$d_{grain_{in}}$	Inner grain diameter	[m]
$d_{grain_{out}}$	Outer grain diameter	[m]
d_{grain_out}	Outer grain diameter	[m]
$f(x+h)$	Function evaluated at point $x+h$	[-]
$f(x)$	True output value	[-]
$f(x)$	Function evaluated at point x	[-]
g_0	Earth's gravitational constant at sea level	[m/s^2]
g_i	gradient of network parameters	[-]
h	Step size	[-]
h	Heat transfer coefficient	[W/m^2K]
k	Thermal conductivity of the fluid	[W/mK]
k_b	Thermal conductivity of the body	[W/mK]
m	Mass of object	[kg]
m	Mass of object	[g]

Symbol	Definition	Unit
m	Wet mass of object	[kg]
m	Mass flow rate	[kg/s]
m_{grain}	Mass of the grain	[kg]
m_{ideal}	Theoretical mass flow rate	[kg/s]
m_{real}	Measured mass flow rate	[kg/s]
m_t	Mean of the gradient	[-]
n	Burning rate exponent	[-]
n	Number of moles	[-]
p	Pressure of the fluid	[Pa]
p	Static pressure at the pressure tab on the airfoil's surface	[pa]
p_0	Total pressure	[pa]
p_∞	Static pressure of the freestream airflow	[pa]
p_∞	Freestream pressure	[pa]
p_a	Ambient pressure	[pa]
p_c	Chamber pressure	[pa]
p_e	Nozzle exit pressure	[pa]
r	Radius	[m]
r	Radius of the core of the grain	[m]
r	regression rate	[mm/s]
t	Time	[s]
t	Simulation time	[s]
t_{liner}	Liner thickness	[m]
u	Magnitude of velocity in horizontal direction	[m/s]
v	Magnitude of velocity in vertical direction	[m/s]
v	velocity	[m/s]
v_i	Update of network parameters of current step	[-]
v_t	Uncentered variance of the gradient	[-]
x	Variable	[-]
x	X-coordinate on the airfoil	[m]
x_{LE}	x-coordinate of the leading edge	[m]
x_{TE}	x-coordinate of the trailing edge	[m]
x_{domain}	Size of the horizontal length of the domain	[m]
x_i	Input data	[-]
y	True output value	[-]
y_i	True output value	[-]
α	Angle of attack	[$^\circ$]
β_1	Gradient weight for mean update	[-]
β_2	Gradient weight for uncentered variance update	[-]
ϵ	Inherent randomness of the world	[-]
ϵ	Parameter to prevent division by zero	[-]
η	Learning rate	[-]
γ	Fraction of update previous update vector	[-]
γ	Specific heat ratio	[-]
Γ	Vandenkerckhove function	[-]
λ	Hyperparameter controlling amount of regularization	[-]
τ	Viscous stress tensor	[Pa]
μ	Data mean	[-]
μ	Dynamic viscosity	[kg/ms]
∇	Del operator	[-]
ν	Kinematic viscosity	[m^2/s]
ρ_∞	Density of the freestream airflow	[kg/m^3]
ρ_c	Total density of the combustion gasses in the combustion chamber	[kg/m^3]
ρ_p	Propellant density	[kg/m^3]

Symbol	Definition	Unit
ρ	Density	[kg/m^3]
σ	Standard deviation	[-]
σ	Temperature coefficient	[-]
θ	Network parameters	[-]
ξ_c	Combustion quality	[-]
ξ_F	Thrust quality	[-]
ξ_n	Nozzle flow quality	[-]
ξ_s	Propellant consumption quality	[-]

List of Figures

List of Tables

1

Introduction

The recent advancements in artificial intelligence (AI), driven by increasing computational power, have unlocked a wide range of possibilities in the aerospace domain [45]. Machine learning (ML), a subset of AI, excels at uncovering hidden relationships in experimental data, leading to improved models and insights. However, traditional ML methods require large amounts of high-quality data to generalise effectively [109].

Currently, simulations in the aerospace domain rely heavily on numerical and analytical methods [83]. These simulations provide valuable insights into design problems but often come with high computational costs and extensive pre-processing requirements [22]. Additionally, complex physical phenomena sometimes lack accuracy or existing models, reducing the reliability of simulations [22]. To address these limitations, experimental testing is conducted to validate simulations. However, testing is expensive, introduces risks, and typically generates limited data compared to its simulating alternatives [82].

Physics-Informed Neural Networks (PINNs) represent a promising new approach to ML. PINNs integrate existing physical laws into the learning process, combining experimental data with known physical models to uncover missing processes or unknown parameters [99]. This hybrid modelling approach requires less data compared to traditional ML methods while offering improved generalisation and accuracy. Moreover, PINNs handle derivatives analytically, eliminating truncation errors inherent in numerical differentiation and enabling meshless modelling [99].

In this research, the potential of PINNs to enhance simulation and surrogate modelling in aerospace applications is explored. Three main use cases are investigated: the modelling of the cooling down of objects, Computational Fluid Dynamics (CFD) simulations, and Solid Rocket Motor (SRM) modelling. Each use case will evaluate the accuracy, computational efficiency, and data efficiency of PINNs in comparison to traditional numerical and data-driven methods.

Research Question

In this report, the following research question will be examined:

> *For aerospace simulation and surrogate modelling applications, to what extent can Physics-Informed Neural Networks enhance computational and data efficiency while maintaining accuracy, compared to traditional numerical methods and purely data-driven models?*

Report Structure

The report is structured into 5 parts. The first part of the report contains a general literature review, which will be presented in chapter 2, containing a detailed introduction to AI, ML, NN and PINNs. After this, the generated code class for the generation of PINNs used for all the other parts of the report will be presented in chapter 3. In the second part of the report, as presented in Part I, the first use case, the modelling of the cooling down of objects, will be examined and explained in detail, introducing the sub-research questions and presenting a literature review related to this topic, as well as presenting experimental data gathered and lastly the various models used and the comparison of performance between the various modelling methods. In the third part of the report, as presented in Part II, the usage of PINNs for CFD simulations will be examined, by introducing the topics and relevant sub-research questions, as well as a literature review, overview of experimental data used and a comparison of the performances between the various modelling methods. In the fourth part, which is presented in Part III, SRM modelling will be examined, firstly introducing the topic and sub-research questions, whereafter the provided experimental data will be presented as well as the processing steps. Next, the various modelling methods will be explained, whereafter a comparison will be presented of the performance of the various modelling methods. In the final part of the report, the research question and various sub-questions will be answered and discussed in chapter 21, whereafter in chapter 22, recommendations for future research will be provided.

2

Literature Review

To investigate PINNs, a good understanding of AI and ML is required. To obtain the necessary knowledge, a literature review will be presented which explains the general principles of AI, ML and other important aspects of this field, as will be presented in section 2.1. In this thesis, various experimental data sets will be used, either measured during the thesis or obtained from external sources. To better understand real-world signals, data collection and noise filtering methods, a part of the literature research is dedicated to these topics, as will be presented in section 2.2. Next, the required theory related to statistical analysis will be examined in section 2.3, which is required to assess the results in a more general context. Lastly, a structured overview of the various sources used during this part of the literature review is presented in section 2.4

2.1. Artificial Intelligence

Artificial intelligence, shortened AI, is a field of science investigating how computers and machines can reason, learn and act in ways which normally require human intelligence [133]. This means that computers are being programmed so that they respond to certain inputs or data in a similar way an intelligent being would. Overall, the field of AI can be split up into a variety of subfields, which are presented in the list below [8]:

- Machine Learning, shortened ML
- Deep Learning
- Natural Language Processing
- Robotics
- Export Systems

The individual AI subfields can themself be subdivided into various subcategories, but for this thesis, which will have its main focus on Physics-Informed Neural Networks, shortened PINNs, only the subfields ML and Deep Learning will be examined in more detail.

2.1.1. Machine Learning

Machine Learning is a subfield within AI which investigates how a machine could learn from data, in a similar way intelligent beings would. In the 1950s, Arthur Samuel defined ML as:

> *"The field of study that gives computers the ability to learn without being explicitly programmed" [66]*

ML is mostly known and used for regression and classification problems [116]. A regression problem is a problem related to predicting continuous series or individual numerical values based on input features [116]. An example of a regression problem is predicting house prices based on location, square footage and number of bedrooms [116]. A classification problem, on the other hand, is a problem related to predicting preassigned labels or categories to input data [116]. An example of this would be predicting

if something is an apple or a banana based on its mass and greenness, or if a picture is a dog or a cat. The difference between regression and classification is thus that regression will predict a new value based on old values, while classification will predict if something belongs to a certain predefined group.

ML by itself can be divided into other subcategories. The previously presented examples are typical examples of supervised ML, one of the four subcategories of ML. The various subcategories of ML are presented in the list below [116]:

- Supervised learning
- Semi-Supervised learning
- Unsupervised learning
- Reinforcement learning

The previously listed subcategories differ by how the system is being trained using input data. In the field of supervised learning, input data is provided, as well as a corresponding output [25]. For semi-supervised learning, all input data is used, but only a portion of the corresponding output is available [25]. Unsupervised learning, on the other hand, only uses input data to find patterns within the data [25]. A completely different way of training is reinforcement learning, where the training of a system is based on punishment and rewards [25]. An object or agent is moving through an environment, and for each state, it will receive a reward or punishment based on its performance and goal [25].

2.1.1.1. Artificial Neurons

To be able for machines to learn certain patterns in data in ML, inspiration was taken from the way the human brain learns and stores new information and skills. The human brain contains many individual neurons [15]. These neurons allow the transfer of electrical and chemical signals to other cells, but only if a certain potential is reached [132]. The way the human brain learns a new skill or information is by forming new connections between individual neurons or by strengthening and modifying existing connections [132]. The connection between neurons exists out of the dendrites of one neuron, which are the input of a neuron, and the axon of another neuron, which are the outputs of a neuron [132]. An illustration of a single neuron of the human brain is presented in Figure 2.1.

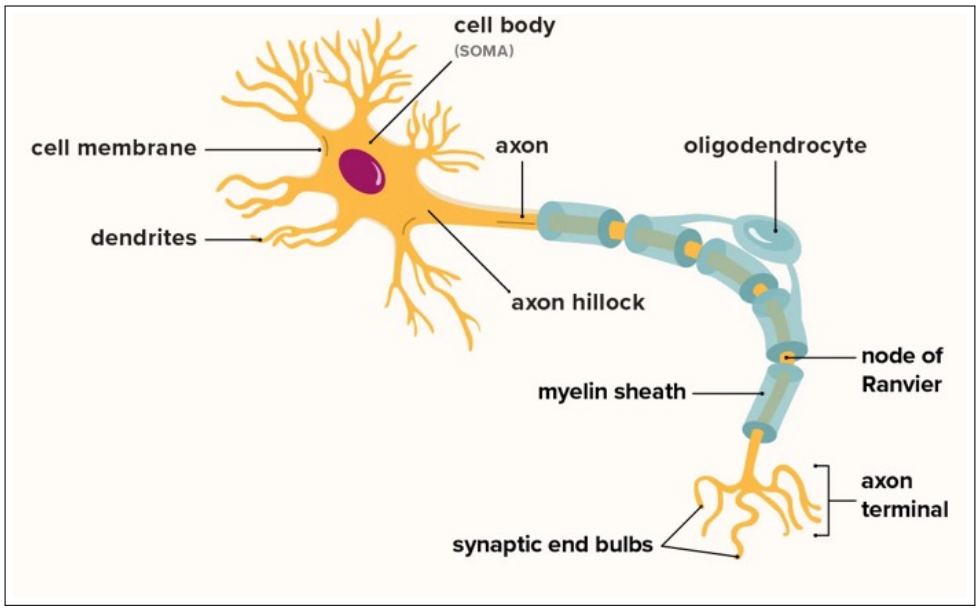


Figure 2.1: An illustration of a single neuron from the human brain [132]

In the field of ML, the concept of how humans learn by strengthening or modifying existing connections between neurons is being used to enable ML [46]. This is being done by modelling the behaviour of neurons found in nature and their connections by making use of artificial neurons [46]. These artificial neurons are connected, forming a network of artificial neurons, which is better known as a neural

network [46]. An artificial neuron is a function which produces an output by making use of an input and applying a weight and bias to this input, whereafter an activation function is applied [46]. The applied weight and bias are first initialised, whereafter they need to be optimised during the training phase to be able to learn the pattern between the input and output data [46]. An illustration of an artificial neuron is presented in Figure 2.2.

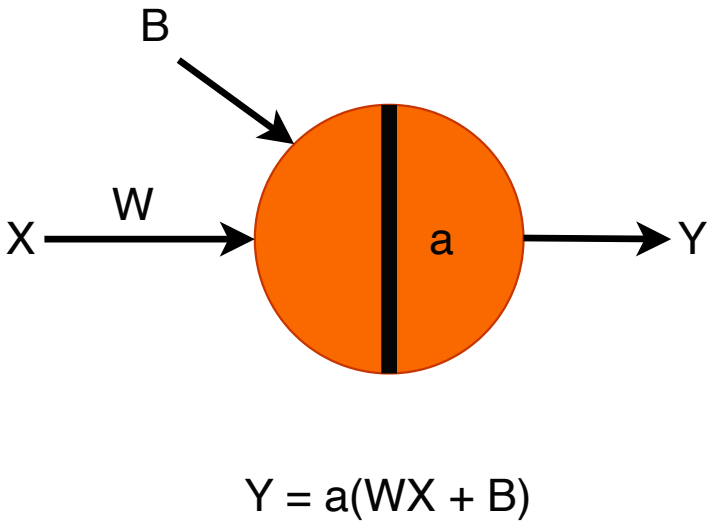


Figure 2.2: An illustration of an artificial neuron used in neural networks

In Figure 2.2, it can be seen that the artificial neuron makes use of two functions. The first function is a linear equation, which is presented in Equation 2.1. In this equation, which is represented as a vector equation, $\vec{W}$ represents the weight vector, $\vec{X}$ represents the input vector, and B represents the bias. It should be noted that an artificial neuron can have multiple inputs, each having its weight [46]. This function will then be used as input for the function a, which is called the activation function. The activation function is used to add non-linearity to the system, such that the artificial neuron would be able to learn and produce non-linear outputs [33]. The bias is used to shift the value produced by the activation function, which is similar to the constant in a linear function [121]. The final output of the artificial neuron is provided by Equation 2.2.

$$f\left(\vec{X}\right) = \vec{W} \cdot \vec{X} + B \tag{2.1}$$

$$Y = a\left(\vec{W}\vec{X} + B\right) \tag{2.2}$$

2.1.1.2. Activation Function

In subsubsection 2.1.1.1, the function a was introduced and is better known as the activation or logistic function. As mentioned in subsubsection 2.1.1.1, the activation function is being used to add non-linearity to the system. However, this is not the only reason that an activation function is used. As previously explained in subsubsection 2.1.1.1, neurons found in nature will pass electrical and chemical signals once they reach a certain activation threshold or potential. To mimic this idea, the activation function can also be used. This is useful, for example, for classification problems where the output can only exist out of two labels and not something in between. Multiple activation functions exist, but the most well-known are presented in the list below, and their equations and shapes er presented in Figure 2.3 [4]:

- The Sigmoid function

- The Hyperbolic Tangent, or shorthand Tanh
- Rectified Linear Unit, or shortened ReLU

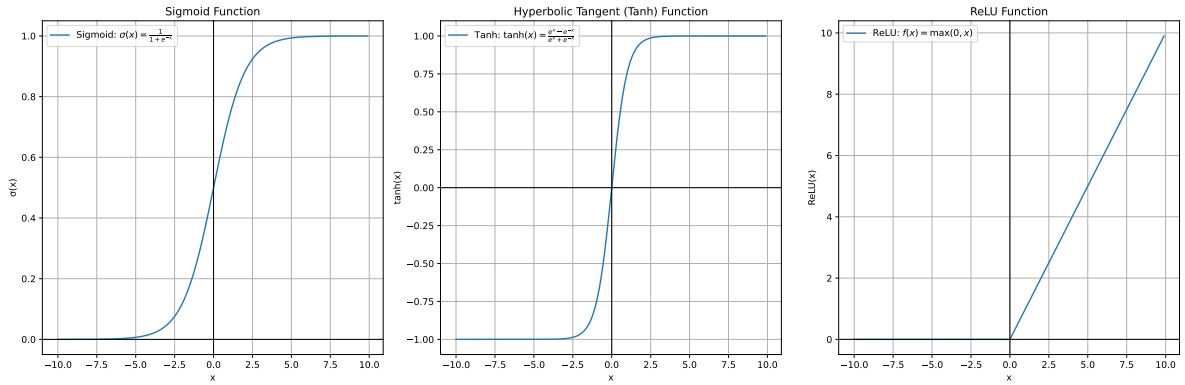


Figure 2.3: Illustrations and equations of the Sigmoid, Tanh and ReLU activation functions

The choice of the activation function depends on the data [46]. The most used activation function is the ReLU activation function [33]. The main advantage of this activation function compared to the Sigmoid and Tanh is that the ReLU activation function is differentiable everywhere in the positive side of the computational domain [33]. The Sigmoid and the Tanh function, on the other hand, will have small values for the gradient at the edges of their respective domains [33]. The values of the gradient become important during the optimisation process, which is also known as the training phase, where the weights and biases of the neural network are being learned [33]. In case, for example, the Sigmoid function is used as an activation function, the training could suffer from the so-called vanishing gradient problem [33]. The vanishing gradient problem is a phenomenon where the gradient of a function becomes close to zero, leading to almost no meaningful updates of the weights and biases during the optimisation process in the training phase [33]. To circumvent this phenomenon, other activation functions can be considered. In general, certain activation functions could be more suitable for a certain problem compared to another activation function [4]. For example, it has been found that it is not recommended to use the ReLU activation function in PINNs for certain applications, since the second-order derivative of the ReLU activation function is zero [127]. This is not useful in case the network has a problem which can be described by a second-order differential equation. In case a PINN has a second-order differential equation embedded, it is recommended to use the hyperbolic tangent function, since this activation function is twice differentiable, thereby preventing the zero-valued second-order derivative and has a lower vanishing gradient property compared to the Sigmoid activation function [127].

2.1.1.3. The Learning Process

As mentioned in subsubsection 2.1.1.1, during the ML training phase, weights and biases of artificial neurons are being learned, or in other words, the optimal weights and biases for a certain input and output are being found [46]. To start the learning process, first, the weights and biases are initialised with initial values. During the process of developing a machine learning algorithm, it is advised to make use of the same initial values to be able to give a fair comparison between training results [46]. After the initialisation, input data is fed into the artificial neurons and predicted outcomes are calculated, passing the various outputs generated by the neurons through the network, which for an individual artificial neuron can be calculated by making use of Equation 2.2. The final predicted outputs are then compared to the known outputs given similar inputs, which were provided to the artificial neurons, whereafter the loss will be calculated. This loss can be calculated using various metrics, where the mean squared error, shortened MSE, is the most used [46]. The equation to calculate the MSE is provided in Equation 2.3 [72].

$$MSE = \frac{1}{N} \sum \left(f\left(x\right) - \hat{f}\left(x\right) \right)^2 \tag{2.3}$$

In Equation 2.3, N presents the total number of data points used, $f(x)$ presents the true output value corresponding to the input data and $\hat{f}(x)$ presents the predicted output value corresponding to the input data.

When all the predicted outputs are being compared to the outputs corresponding to the input data, or, in other words, to the true values, a total cost is obtained [46]. In this thesis, the cost function is the total addition of individual loss functions. The main goal of the ML training process is to minimise the cost. To do so, after the cost has been obtained, the guesses for the weights and biases are updated based on the cost [46]. After the weights and biases have been updated, the process will start over, updating the weights and biases each time based on the cost [46]. The value of the cost can increase or decrease, which leads to the preferred direction in which the weights and biases should direct themselves [46]. The process stops after a certain threshold for the cost is reached, or if the maximum number of iterations has been exceeded, the set number. An illustration of this process is provided in Figure 2.4

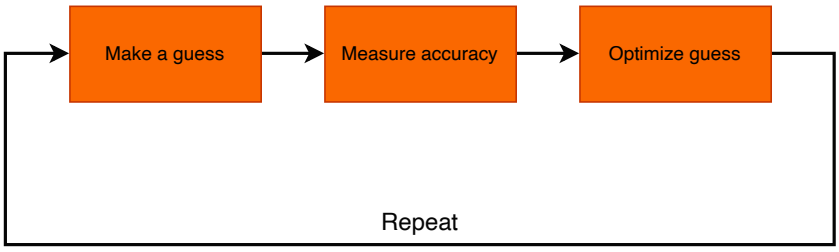


Figure 2.4: An illustration of the learning process within the ML subfield

2.1.1.4. Weight and Bias Initialization

As described in subsubsection 2.1.1.3, the weights and biases need to be provided with an initial guess before they can be optimised. Neural Networks are very sensitive to the initialisation of weights and biases, which has a consequence on the stability of the learning rate and convergence time [20]. The main purpose of weight initialisation is to prevent vanishing or exploding gradients of the weights and biases during the training process [20]. A network with suboptimal initialisation leads to instability to converge and/or slow convergence rates [20]. The most used weight initialisation methods are presented in the list below [20]:

- All-zeros initialisation and Constant initialisation
  This method sets all weights to zero or a constant and sets all activations to the same values, leading to a linear model [20]
- Random initialisation
  The weights are being initialised with random values close to zero [20]. This method prevents learning the same features from the inputs of the Neural Networks [20].
- LeCun initialisation
  The weights are being set such that the variance is set constant [20]. It prevents vanishing or explosive gradients during the training process [20].
- Xavier initialisation
  The weights are being set in such a way that they are constant across every layer [20]. This prevents vanishing and exploding gradients during the training process [20].
- Kaiming initialisation
  The weights are initialised in such a way that the non-linearity of the activation function, such as ReLU, is preserved [20]. With this method, the magnitude of the inputs can be reduced or magnified exponentially [20].

For complex data sets, weight initialisation is very important [20]. Choosing a correct weight initialisation method aids in the performance of the Neural Networks and convergence rates [20]. In general, Kaiming initialisation works best for non-differentiable activation functions such as Relu. LeCun and

Xavier, on the other hand, provide good results for differential activation functions such as the Sigmoid or Tanh [20].

2.1.1.5. Cost Function

During the ML training phase, the cost function is being minimised according to a certain metric, which was explained in subsubsection 2.1.1.3. Various metrics exist, and the most commonly used ones will be explained in this section.

The Mean Error, or shortened ME, calculates the average error out of all measurements [71]. This metric is barely used because once you sum the errors, the positive errors will compensate for the negative errors, which leads to an unrealistic representation of the error [71]. To obtain the ME, use can be made of Equation 2.4 [71].

$$ME = \frac{1}{N} \sum (\hat{y} - y) \tag{2.4}$$

In Equation 2.4, N represents the total number of data points, $\hat{y}$ represents the predicted output value, and y represents the true output value.

To overcome the problem of compensation of the positive and negative errors, as introduced by the ME, use can be made of the Mean Average Error, or shortened MAE. This error is also known as the L1 error [120]. This error penalises all errors equally [120]. The MAE can be calculated by making use of Equation 2.5 [120].

$$MAE = \frac{1}{N} \sum |\hat{y} - y| \tag{2.5}$$

In Equation 2.5, N represents the total number of data points, $\hat{y}$ represents the predicted output value, and y represents the true output value.

Another metric which is commonly used is the Mean Squared Error, or shortened MSE [120]. This error penalises large errors more heavily compared to smaller errors [120]. The MSE is also known as the L2 error [120]. A downside of the MSE is that the calculated error is not on the same scale as the predicted label values [120]. To obtain the MSE, use can be made of Equation 2.6 [120].

$$MSE = \frac{1}{N} \sum (\hat{y} - y)^2 \tag{2.6}$$

In Equation 2.6, N represents the total number of data points, $\hat{y}$ represents the predicted output value, and y represents the true output value.

To have the MSE on the same scale as the predicted label values, use can be made of the Root Mean Squared Error, or shortened RMSE [120]. The RMSE can be calculated by making use of Equation 2.7 [120].

$$RMSE = \sqrt{MSE} \tag{2.7}$$

Another used metric is the Mean Absolute Percentage Error, or shortened MAPE [138]. The MAPE can be calculated by making use of Equation 2.8 [138].

$$MAPE = \frac{100\%}{N} \sum \left| \frac{\hat{y} - y}{y} \right| \tag{2.8}$$

In Equation 2.8, N represents the total number of data points, $\hat{y}$ represents the predicted output value, and y represents the true output value.

The metrics introduced above can be used to define the loss function. An example cost function is presented in Equation 2.9, where use is being made of the MSE as a loss function metric [96].

$$J(\theta) = \frac{1}{N} \sum_{i}^{N} \left(f(x_i|\theta) - y_i \right)^2 \tag{2.9}$$

In Equation 2.9, N represents the total number of data points, $f(x_i|\theta)$ represents the predicted output value, where x_i represents the input data, θ represents the parameters of the network (weight and biases) at the current epoch and y_i represents the true output value.

2.1.1.6. Neural Network

Neurons found in nature are connected to be able to learn, as was briefly explained in subsubsection 2.1.1.1. A single artificial neuron can only learn simple problems, such as a two-label classification problem or simple regression problems. However, if multiple artificial neurons are connected, more complicated relationships can be learned during the training phase [46]. If multiple artificial neurons are connected, the resulting network is called a neural network, or shortened, NN [46]. An NN consists of an input layer, an output layer and at least one hidden layer. Each layer can have as many artificial neurons as the problem requires, as long as the number of input features and output features corresponds with the problem. An example of a single hidden layer NN is presented in Figure 2.5.

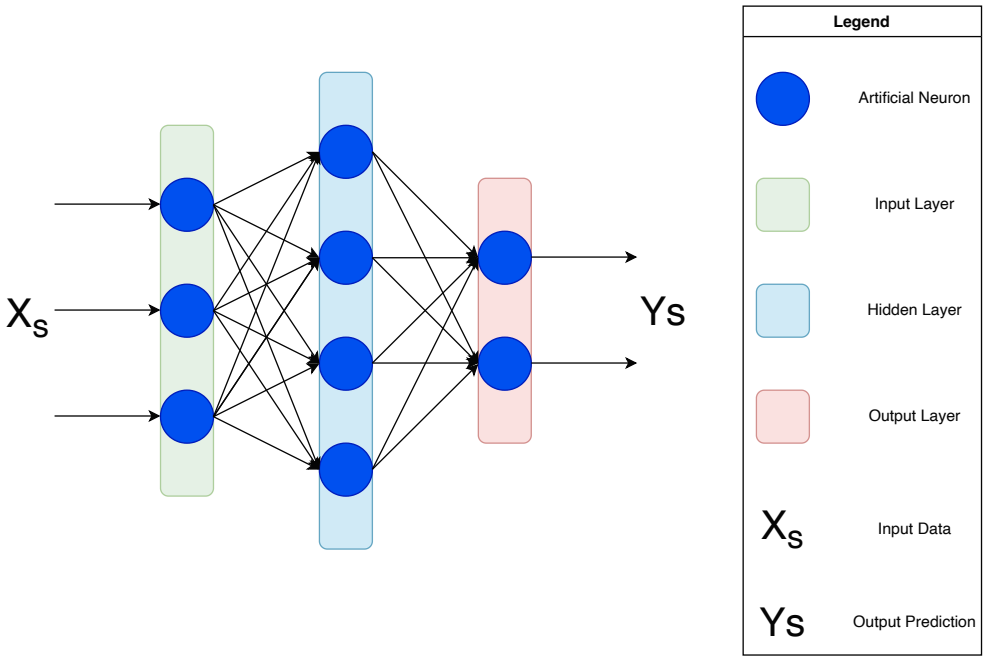


Figure 2.5: An illustration of an example artificial neural network with an input layer of 3 neurons, 1 hidden layer with 4 neurons and an output layer consisting of 2 neurons

It can be seen from Figure 2.5 that the connections between the individual artificial neurons are between the various layers, but there is no connection between the artificial neurons within the same layer. Each connection represents a certain weight term, which should be learned during the training process.

In case an NN has multiple hidden layers, it is called a deep NN, or shortened, a DNN [75]. It has been shown that single-layer feed-forward NNs are a class of universal approximators [51]. This means that a NN with a single hidden layer and a finite number of artificial neurons can approximate any function [51]. However, a single-layer NN which contains a large number of artificial neurons, requires a higher computational time to reach a certain accuracy, compared to a multi-layered NN reaching the same amount of accuracy [59]. Therefore, multi-layered NNs or DNNs are mostly used for machine learning operations [59].

2.1.1.7. Optimizer

In subsubsection 2.1.1.3, it was briefly described how the ML training process works. To recap, this works by updating the weights and biases based on a cost function. To determine how the weights and biases are being updated, a process called gradient descent [46] is being used. During this process, the gradient of the cost function concerning the network parameters is calculated. The gradient represents the direction of maximum change, meaning that, to minimise the cost function, the weights and biases should be updated in the reverse direction. So, if the gradient is positive, the values of the weights and biases should decrease, while the opposite is true if the gradient has a negative value [46]. For an NN, this process is performed for each layer and cross-tracked over the entire NN. The process of performing gradient calculations and cross-tracking over the entire NN is called back-propagation [29]. The standard equations to update the weights and biases for gradient descent are provided in Equation 2.10 [101].

$$\begin{aligned} W_{i+1} &= W_i - \eta \frac{\partial J\left(W\right)}{\partial W} \\ B_{i+1} &= B_i - \eta \frac{\partial J\left(W\right)}{\partial B} \end{aligned} \tag{2.10}$$

In Equation 2.10, $J\left(W\right)$ represents the cost function, W_i represents the weight of the artificial neuron, B_i represents the bias of the artificial neuron and η represents the learning rate. The learning rate is introduced to determine the step sizes at which a gradient will be calculated [101].

Unfortunately, gradient descent is not always able to reach the global minimum of the cost function [101]. In case there are local minima or plateaus, the gradient descent process will not be able to reach the global minimum. This can be seen in the illustration presented in Figure 2.6.

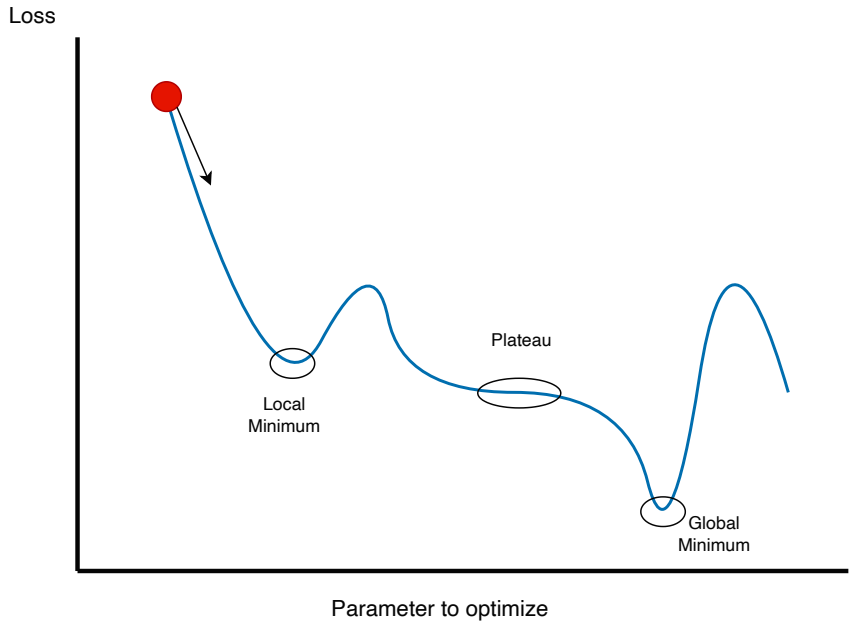


Figure 2.6: An illustration of the gradient descent algorithm, showing a local minimum, plateau and global minimum

In Figure 2.6, the red dot represents the cost function. The arrow pointing in the downward-right direction represents the gradient at that point. It can be seen that the dot will end up in a local minimum or at the plateau, but will currently not end up at the global minimum located to the right of the figure.

The learning rate has a very important role in obtaining the global minimum. The main goal of the learning rate is to control how much the weights of the NN are adjusted in the direction of the gradient during optimisation [101]. In case the learning rate is too large, the cost function will overshoot the minimum and will not be able to reach the global minimum value [101]. In case the learning rate is too small, it will take a long time before the cost function will be able to reach the minimum value [101]. The learning rate is also called a hyperparameter and can be tuned, making use of a cost function against an epoch graph [7]. An epoch is one full pass through the training data, during which the model calculates the loss for all data points and updates its weights accordingly [7]. An illustration of the impact of the learning rate, η, on the cost function is presented in Figure 2.7.

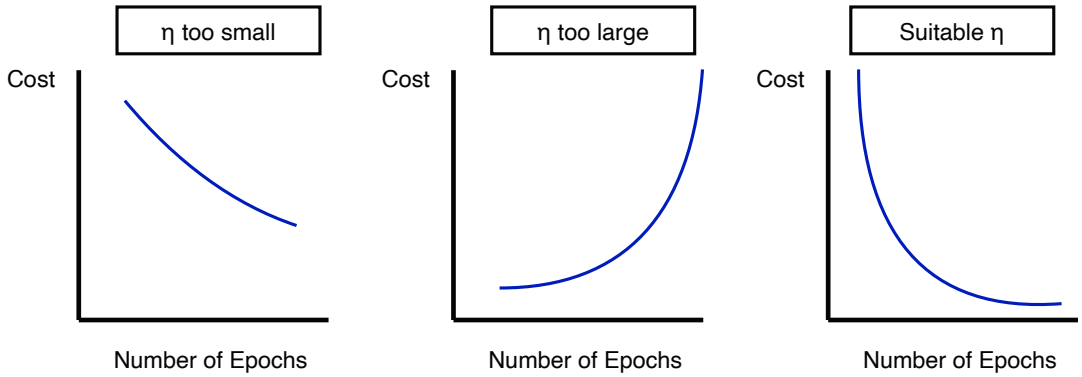
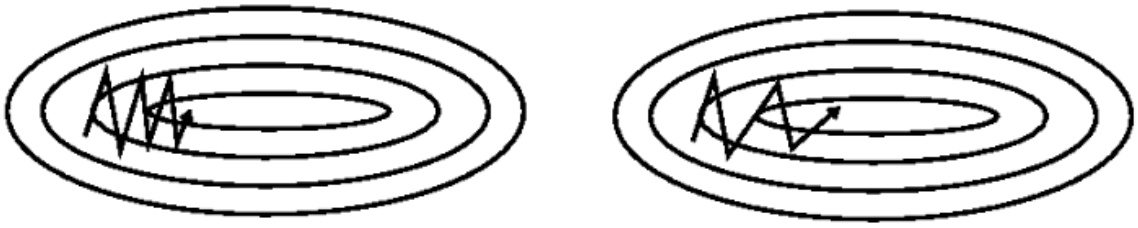


Figure 2.7: An illustration showing the influence of the learning rate η on the loss function and number of epochs

The weights and biases are also called parameters of the network [101] and are often represented by θ. This transforms Equation 2.10 into Equation 2.11.

$$\theta_{i+1} = \theta_i - \eta \frac{\partial J(\theta)}{\partial \theta} \tag{2.11}$$

The gradient descent algorithm has three different variants in the amount of data used to update the weights [101]. The vanilla gradient descent, also known as batch gradient descent, uses all data points to perform an update on the weights and biases [101]. A major downside of this variant is that it takes a lot of computational time and memory [101]. Stochastic gradient descent, on the other hand, uses only one data point to perform an update of the weights and biases [101]. To combine both worlds, mini-batch gradient descent takes n samples to update the weights and biases [101]. Overall, the most used gradient descent method is the mini-batch algorithm, varying the size of the data points used to perform an update on the weights and biases [101]. The gradient descent algorithm, as presented in Equation 2.11, suffers major downsides, such as the possibility of overshooting the global minimum or getting stuck at local minima or plateaus [101], as was presented in Figure 2.6. Besides this, the learning rate is currently set as a fixed value, which is not the ideal situation, as it would be beneficial to have smaller step sizes in case you get closer to the optimum, to prevent overshooting. To solve these issues, use is mostly made of the Adam optimisation algorithm during the training of NNs, where Adam is shortened for Adaptive Momentum Estimation [101]. This method makes use of adaptive learning rates and is a combination of the momentum method and the RMSprop method. The momentum method is a method which is used to accelerate the gradient descent in the relevant direction and dampen oscillations around local minima, of which an example is presented in Figure 2.8 [101].



(a) Example of the path of the gradient descent method without momentum [101]

(b) Example of the path of the gradient descent method with momentum [101]

Figure 2.8: An illustration of the gradient descent method with and without momentum added to the optimiser

To add momentum to the previously discussed gradient descent method, which was presented in Equation 2.11, a fraction of the update vector, which is represented as γ, of the last time step is added to the current update [101]. The formula of the gradient descent method with momentum is presented in Equation 2.12 [101].

$$\begin{aligned} v_i &= \gamma v_{i-1} + \eta \nabla_\theta J(\theta) \\ \theta &= \theta - v_i \end{aligned} \tag{2.12}$$

In Equation 2.12, v_i represents the update of the parameters of the network of the current time step, and v_{i-1} represents the update of the parameters of the network of the previous time step [101].

RMSprop is a method used to decrease the learning rate [101]. If both the Momentum and RMSprop methods are being combined, one will get the Adam method, combining momentum and an adaptive learning rate. The formula for the Adam method is presented in Equation 2.13 [101].

$$\begin{aligned} m_i &= \beta_1 m_{i-1} + (1-\beta_1) g_i \\ v_i &= \beta_2 v_{i-1} + (1-\beta_2) g_i^2 \\ \hat{m}_i &= \frac{m_i}{1-\beta_1^i} \\ \hat{v}_i &= \frac{v_i}{1-\beta_2^i} \\ \theta_{i+1} &= \theta_i - \frac{\eta \hat{m}_i}{\sqrt{\hat{v}_i} + \epsilon} \end{aligned} \tag{2.13}$$

In Equation 2.13, m_t and v_t represent estimates of the mean and the uncentered variance of the gradients, g_i represents the gradient of the parameters concerning the network, $\frac{\partial J(\theta)}{\partial \theta}$ [101]. For β_1, β_2 and ϵ, some recommended values are available, which are 0.9 for β_1, 0.999 for β_2 and $1 \cdot 10^{-8}$ for ϵ [101].

2.1.1.8. Automatic Differentiation

As discussed in subsubsection 2.1.1.7, NN parameter optimisation is performed by minimising the cost function. To do so, the gradient of the loss function over the network is obtained, making use of a process called backpropagation. An example calculation of a single artificial neuron will be presented to present the working principle, of which a visual representation is presented in Figure 2.9.

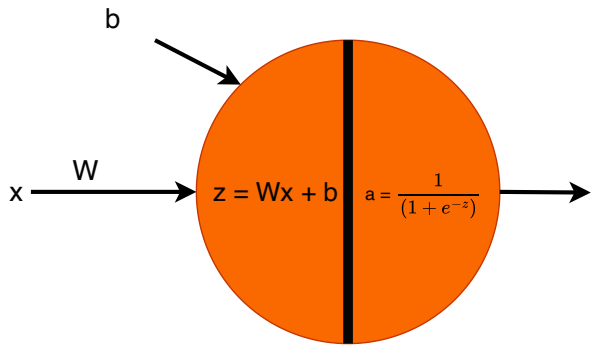


Figure 2.9: Example artificial neuron to demonstrate the backpropagation process

For this example, the MSE will be used as an error metric, as well as having a single input and output, to minimise the difference between the predicted output and the known output. To do so, the most optimal weight and bias need to be obtained, making use of Equation 2.10. To be able to obtain the gradients of the cost function concerning the network parameters, use will be made of the chain rule of calculus, as well as partial derivatives. A partial derivative is a derivative of a function with multiple variables, where one variable is taken as differentiable while the others are kept constant [95]. Making use of Figure 2.9, the following equations will be used to obtain the gradient of the cost function concerning the network parameters, which are presented in Equation 2.14.

$$\begin{aligned} J &= \frac{1}{2}(T-a)^2 \\ a &= \frac{1}{(1+e^{-z})} \\ z &= Wx + b \end{aligned} \tag{2.14}$$

In Equation 2.14, J represents the error measure function, T represents the true value, a represents the prediction given by the activation function, which for this example is the Sigmoid activation function, z represents the linear combination of the weight, input and bias, W represents the weight of the neuron, b represents the bias of the neuron and x represents the input of the neuron. To obtain the partial derivatives for the weight and bias, $\frac{\partial J}{\partial W}$ and $\frac{\partial J}{\partial b}$, use can be made of the chain rule, which for these two partial derivatives are presented in Equation 2.15.

$$\begin{aligned} \frac{\partial J}{\partial W} &= \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial W} \\ \frac{\partial J}{\partial b} &= \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial b} \end{aligned} \tag{2.15}$$

For the equations provided in Equation 2.14, the corresponding partial derivatives are provided in Equation 2.16.

$$\begin{aligned} \frac{\partial J}{\partial a} &= -(T-a) \\ \frac{\partial a}{\partial z} &= \frac{e^{-z}}{(1+e^{-z})^2} \\ \frac{\partial z}{\partial w} &= x \\ \frac{\partial z}{\partial b} &= 1 \end{aligned} \tag{2.16}$$

As can be seen from this example, various partial derivatives are being calculated during the backpropagation process to optimise the weights and biases to minimise the difference between the predicted output and the true output. When making use of PyTorch, a Python library for the development of NN, the various data points, such as inputs, outputs, weights and biases, are stored in tensors [1]. All operations performed on these tensors and the relationship between individual tensors are being stored in a so-called computational graph [124]. A computational graph is a tool which can be used to identify relationships between certain inputs and outputs and how they are connected [32]. In PyTorch, a directed acyclic graph (DAG) is constructed, which is a special type of computational graph which can not form closed loops [1, 32]. When a DAG is being made in PyTorch, each node represents a tensor, and the arrows connecting the nodes are the operations between the tensors. An example of a DAG of the function $f(x, y) = x^2 + y^3$ and how this is being stored in PyTorch is presented in Figure 2.10.

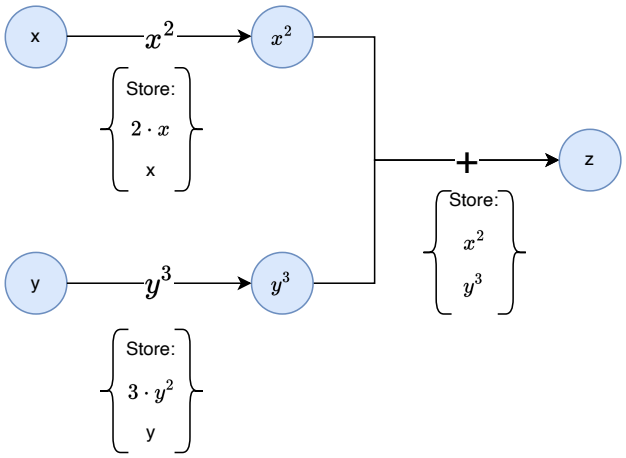


Figure 2.10: Example of a directed acyclic graph as being used in PyTorch operations

During the forward pass of an NN, which means that the output will be calculated by following the path of the input through the NN and performing all calculations to get to a final output, the computational graph is being constructed dynamically, while during the backwards propagation, the graph is being followed in reverse, being able to calculate the gradients making use of the chain rule [1]. This operation can be performed since, during the forward pass, each operation connecting the tensors is being stored [1]. This means that the computational graph stores which tensors are connected and by which operations, as well as the input which produced that specific result and the backwards function, or in other words, how the derivative of the output concerning its input needs to be computed [16]. Standard operations of derivatives are being stored as rules [16]. A large advantage of this system is that, rather than computing the derivatives numerically, each derivative is stored analytically, and the requested partial derivative can be obtained by making use of the chain rule during the backpropagation process through the network [16].

2.1.1.9. Problems During the Training Phase

The main goal of the ML training phase is to minimise the cost function as much as possible, as was previously discussed. However, if, for example, the MSE is taken as the metric for the cost function, the MSE can be decomposed into its bias-variance decomposition, which results in the formula as presented in Equation 2.17 [134].

$$E\left(y_0 - \hat{f}\left(x_0\right)\right)^2 = Var\left(\hat{f}\left(x_0\right)\right) + \left[Bias\left(\hat{f}\left(x_0\right)\right)\right]^2 + Var\left(\epsilon\right) \tag{2.17}$$

In Equation 2.17, y_0 represents the true value, $\hat{f}\left(x_0\right)$ represents the predicited value and ϵ represents the noise.

The decomposition of the MSE contains the following aspects [134]:

- The variance, which represents how much a model changes when retrained on new data. The variance is influenced by noise in training data, the chosen model and the size of the training data.
- The bias, which represents the error introduced by the approximation of the model. The bias is influenced by the number of features and the choice of the model.
- The irreducible error, which represents the inherent randomness of the world, also known as noise (ϵ)

In case the variance is high and the bias is low, the model is said to be overfitting the data. This means that once the model receives unseen new inputs, the model provides meaningless predictions, due to being trained too close to the training data [109]. In contrast, the model is said to be underfitting the data in case the variance is low but the bias is high, which results in the fact that the model is unable to provide sufficient accuracy in predicting the output of the training data and fails to predict outputs on general data [109]. An illustration of overfitting and underfitting is presented in Figure 2.11.

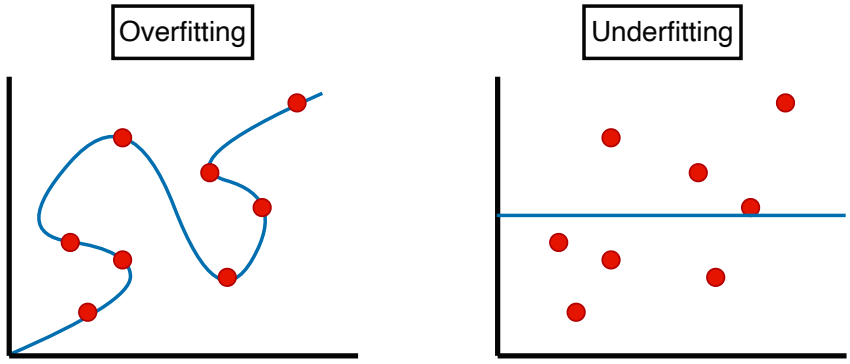


Figure 2.11: An illustration presenting overfitting and underfitting of a trained data-driven model on some data points

The main goal of ML is to train a model which has a good performance on general, new, unseen data. Therefore, overfitting and underfitting are undesirable outcomes of the training phase. To prevent this, various methods exist to prevent overfitting and underfitting. In general, the more data there is available to train the model, the better the model's performance will be on new, unseen data. Besides this, use can be made of regularisation methods.

2.1.1.10. Regularization

In subsubsection 2.1.1.9, it was discussed how a model could overfit the training data, resulting in a model which has unsatisfactory performance on general unseen new data. The model is said to overfit the training data in case the variance of the model is high, while the bias is low. To solve this issue, the bias has to be increased, while the variance should decrease [109]. This process involves reducing the complexity of the solution, or minimising the optimisation parameters (weights and biases of the artificial neurons) [109, 96]. The process of minimising the optimisation parameters is called regularisation [109]. To be able to regularise a model, an additional loss function should be added to the cost function. Various regularisation methods exist, of which the L2 regularisation method is presented in Equation 2.18 [96].

$$J(\theta) = \frac{1}{N}\sum_{i}^{N}\left(f\left(x_i|\theta\right) - y_i\right)^2 + \lambda||\theta||_2^2 \tag{2.18}$$

In Equation 2.18, λ represents a hyperparameter which controls the amount of regularisation, and, to ensure that the individual loss functions are of a similar order of magnitude [109]. Another method of regularisation is called dropout [109]. During the training phase, artificial neurons are temporarily

removed in each hidden layer of the network, such that the model will not rely heavily on certain artificial neurons in the network [109].

2.1.1.11. Data Scaling

When multiple input data, or features, are being used in an NN, it was found that when the magnitudes of the individual features differ widely, the possibility exists that the NN is not able to converge [107]. To circumvent this issue, the features could be scaled, such that they become of similar orders in magnitude [107]. Various scaling methods exist, where the most well-known is called normalisation or min-max scaling [107]. During this scaling method, the values of the features are being scaled such that they become in the range between [0,1] [107]. To do so, use is made of Equation 2.19 [107]. Normally, only the features are scaled, and the labels do not have to be scaled [107].

$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2.19}$$

A downside of this method is that, when the numerical values of certain features are small, while the numerical values of other features are large, the small-scaled features will be scaled to a numerical value of zero [107]. This means that, while changing these features during the training process, the model will not be able to learn the different inputs, as most of the time, the specific features will have a numerical value of zero. To circumvent this problem, use can be made of other scaling methods, such as logarithmic scaling. Making use of the log1p function, the natural logarithmic value of the input data, with the addition of the value one, will be calculated. This is useful, as previously explained that the min-max scaling process could make certain parts of the input data zero if they are small compared to other values. The log1p function, on the other hand, will make these values small but non-zero. The formula for the log1p is presented in Equation 2.20 [31].

$$\text{log1p}(x) = \log(1 + x) \tag{2.20}$$

In Equation 2.20, x represents the input data, while $\log$ represetns the natural logarithm. To clarify this, $\log(1) = \ln(1) = 0$.

2.1.1.12. Data Structure and Assessment of Performance

Data is a very important topic for ML, as this is the main source of finding the relationship between input and output of a model [21]. To train a model which performs well on general data, lots of data points are required [21]. Furthermore, the data needs to be processed upfront to the correct format, such that it can be processed by the NN [21]. Besides having lots of data available, there is generally another problem related to ML, namely, fairly assessing the performance of a trained model [21]. If a model has used all the available data during the training phase, it can no longer be tested for its performance on unseen new data sets, as these are not available [21]. Therefore, the available data should be divided upfront into test, training and validation data, to be able to fairly assess the performance of the developed model [21]. The reason why there are three different types of datasets is that the training dataset is used purely for training the model, and thereby obtaining the most optimal network parameters [21]. The validation data is used to check the performance of the network on general, unseen new data [21]. However, the validation data is also used to tune hyperparameters, such as the learning rate η and possible other hyperparameters, such as the regularisation parameter λ [21]. Therefore, it could be argued that, although the network does not directly make use of the validation data, the network is still being optimised using this data and therefore, has not been able to fairly assess the performance of the network [21]. Therefore, the third dataset, the test dataset, is used to be able to fairly test the performance of the model on completely unseen new data [21]. It is important to note that the available datasets are assumed to be a sufficient representation of the general population, meaning that if the model has been found to work sufficiently on the test dataset, it will have an acceptable performance when used on data outside the development dataset [21].

Various methods exist to split all the data into the three different datasets. The one-time split is the easiest method, and it only splits the data into three different groups [21] once. The leave-one-out cross-validation (LOO-CV) method only divides the data into a test and training group, where each

time a sample will be used as test data, the other samples will be used as training data [21]. After this, the next sample will be used and the model will be retrained on the other data, including the previous test data sample [21]. This will continue up until all the data samples have been used once as test data, while not being part of the training data, whereafter an estimate of its performance can be determined [21].

The method which is considered to provide the most reliable estimate of the performance of the model is called nested k-fold cross-validation [21]. During this process, the training set is first split into a train and a test fold during the outer cross-validation phase [21]. Next, on each outer train fold, a cross-validation is performed to tune the various hyperparameters [21]. The best hyperparameters found during this phase will be used to train the entire outer training fold, whereafter their performance will be evaluated using the outer test fold [21]. This step will be repeated for the other outer test folds, and this way, a model performance can be obtained using the mean error and the standard deviation [21]. A visual overview of this process is presented in Figure 2.12.

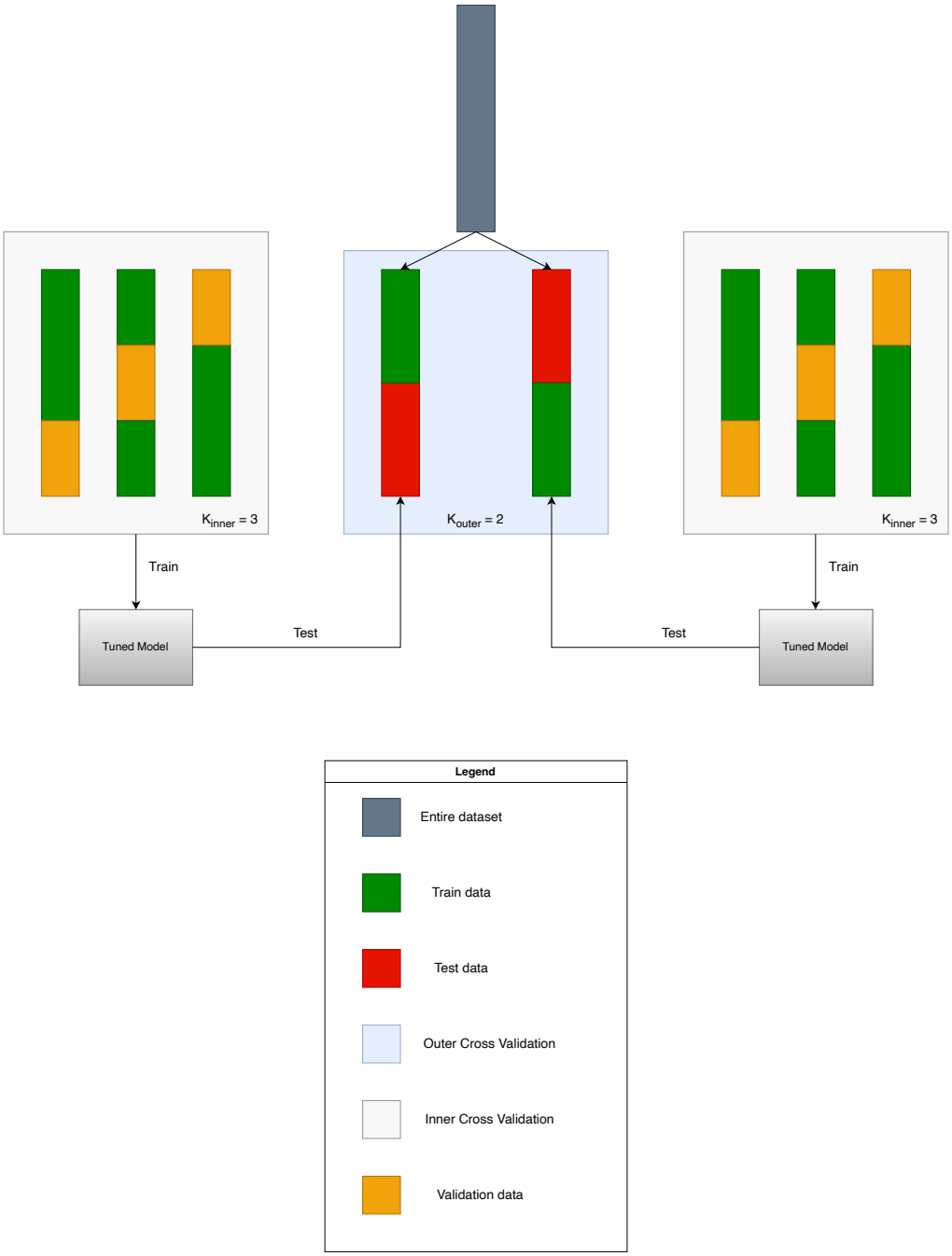


Figure 2.12: An illustration of nested k-fold cross validation, where k is set to 2 in the outer loop and 3 in the inner loop

Unfortunately, not all datasets are suitable for each dataset split method [21]. If the datasets are small or very specific, only certain ways of splitting the data can be used. For example, it is advised to make use of LOO-CV for small datasets, as this would provide the most sufficient estimation of the model's performance [21].

2.1.1.13. Design of a Neural Network Architecture

An important aspect of an NN is its architecture. The architecture of the NN refers to the overall structure of the network, which is better known as the number of neurons and how these neurons are connected [44]. In subsubsection 2.1.1.6, it was discussed that a single-layer NN with a finite number of artificial

neurons is capable of modelling every continuous function; in other words, a single-layer NN can act as a universal approximator [51]. However, this does not mean that a single-layer NN can succeed in the training process and generalise well [44]. It has empirically been shown that deep NNs, NNs with multiple hidden layers, are better at generalisation for a wide variety of tasks [44].

When defining an NN architecture, it is advised to start with two hidden layers with an equal number of artificial neurons [44]. After the architecture of the NN has been defined, overfitting on the training data points should be achieved, whereafter more layers can be added with the same number of neurons [44]. Furthermore, it is advised to use not more than five hidden layers [44]. For PINNs, it is advised to make use of 3-6 hidden layers, with 128-512 neurons per layer [127]. The main reason for this is that NNs which are too narrow and shallow are not capable of capturing complex non-linear functions, while a too wide and deep can be difficult to optimise [127].

2.1.1.14. Types of Neural Networks

As discussed in subsubsection 2.1.1.6, an NN consists of layers and neurons, which are connected [46]. However, different types of NN exist, and each has its advantages and preferred use cases. The most used NNs are examined and explained in the list below [44, 108]:

- Feed-forward Neural Networks, shortened FNN.
  These types of networks are mostly being used for regression and classification tasks [44]. The working principle behind these types of NN is that the inputs will go from the input layer through the neurons of each layer towards the output layer, generating an output [108]. In these types of networks, it is assumed that the inputs are independent of the outputs [108]. Another name for a multi-layer FNN is Multi-Layer Perceptron (MLP) [108].
- Convolutional Neural Network, shortened CNN.
  These types of networks are mainly being used for computer vision tasks, speech recognition and image classification [44, 108]. A CNN is also a type of FNN but has an additional processing step before the inputs enter the FNN [108]. In the processing step, a convolutional layer and a pooling layer are added before the FNN [108]. These layers can learn and recognise features from the input data, so that the input data does not require additional pre-processing steps [108]. The convolutional layer is used to extract distinct features from the input [108]. The higher convolutional layers can extract more abstract features, while the lower convolutional layers can detect basic features such as lines and textures [108]. The pooling layer is being used to sample down the input and change its dimensionality [108]. The pooling layer has as advantage that it makes the network more prone to noise and distortion of input data [108]. After the features have been extracted, they will be passed to the input layer of the FNN, whereafter it will predict and output in a similar manner as the FNN [108].
- Recurrent Neural Networks, shortened RNN.
  These types of networks are mostly being used for natural language processing, video classification and speech recognition [44, 108]. An RNN contains an internal memory, assuming that the inputs are dependent on the outputs [108]. This means that RNNs can capture sequential data and relationships [108]. An RNN works as a loop, meaning that the outputs will be fed back into the network as inputs [108]. A major downside of RNNs is their short-term memory, which means that RNNs are not able to obtain information over long sequences of data [108]. The main strength of RNNs is their ability to process short sequential data over time, making use of their ability to capture dependencies over time [108].
- Long-Short Term Memory, shortened LSTM.
  LSTMs are a variant of RNN, which solves the issue of RNNs not being able to capture long-term dependencies [108]. LSTMs have been proven to be more effective at retraining and using information over longer time sequences [108]. In LSTMs, the input of the current time step, as well as the output of the previous time step, are used as inputs to the network, which generates an output for the next time step [108]. Unfortunately, LSTM struggles with input data with no clear sequential data structure [108].

2.1.1.15. Verification and Validation for Machine Learning Algorithms and Neural Networks

Verification and validation are an important part of the design of a system [82]. This is not only important for physical systems, but also for software [82]. Verification is the process of checking if a product

adheres to its requirements or specifications, or, in other words, has the product been developed correctly [82]. Validation is the process of seeing if the right product has been created for an intended purpose [82]. For software simulations, verification is the step of seeing if the chosen simulation model represents the physical model [126]. This relates to investigating the effect of discretisation, the errors in calculations and if there are any errors in the computer program itself [126]. The validation process for software simulations is related to identifying if the simulation results accurately predict the physical problem [126]. This process investigates the effect of the assumptions made during the development of the model, how reliable the input data is, the accuracy of the prediction model and to what extent the model is validated [126].

For machine learning models, the model to provide predictions is obtained by finding the weights and biases of the NN, and is not provided up front. Therefore, the verification and validation process for this type of model will be different compared to standard simulation software. Verification of machine learning models is mostly being done by making use of sophisticated libraries, but also by changing the inputs to check their robustness [128]. Furthermore, providing standards checks to see if there are no coding mistakes visually remains very important [128]. An important aspect is to see how a model would behave to adversarial attacks [128]. An adversarial attack on an NN is a perturbation to the input signal which leads to incorrect predictions by the NN, while the original inputs would lead to correct predictions [128]. It should be checked if a trained NN is susceptible to this phenomenon. In case the NN is found to be susceptible to this, the NN should be retrained with adversarial data, making the model less prone to this phenomenon [128]. Validation of machine learning models for supervised learning is mostly based on checking for overfitting and underfitting [24]. Besides this, like for classical validation, the model inputs can be checked for unseen outputs of experimental data, assessing the accuracy of the machine learning model, of which the process was already explained in subsubsection 2.1.1.12 [24].

2.1.2. Physics-Informed Neural Networks

As discussed in subsubsection 2.1.1.12, data is very important for ML techniques. It was also discussed how lots of data is required to obtain models which perform sufficiently well on unseen new training data. However, in real-world applications, situations exist where a limited amount of data is available, resulting in an insufficient data-driven model. However, sometimes, prior knowledge about these types of situations exists, in the form of differential equations [99]. These differential equations can model the physical laws that govern the dynamics of a system and can be used as a regularisation term in the cost function to include this prior knowledge inside the ML training process [99]. This concept is known as a Physics-Informed Neural Network or shortened PINN, which enables the use of NN on smaller datasets, with a better generalisation behaviour [99]. PINNs are based on automatic differentiation, a process which was explained in subsubsection 2.1.1.8, to differentiate NNs concerning their inputs and model parameters [99]. PINNs add a constraint to the training process such that the model is not only being optimised on the observed data but also applies certain constraints in terms of symmetry, variances or conservation principles based on the physics laws in the cost function [99]. PINNs can be used for data-driven solutions, as well as for data-driven discovery [99]. The data-driven solution is similar to other ML processes, where a certain set of inputs will provide an output [99]. However, data-driven discovery can obtain certain unknown quantities of a differential equation by letting these quantities be parameters of the NN [99]. This way, the parameters will be used during the optimisation process of the NN and be learned similarly to the other network parameters (weights and biases) of the NN [99]. These types of PINNs are called Inverse PINNs, or shortened I-PINNs [99]. A major advantage of PINNs is that they are less prone to noise in data samples and that they can provide more accurate modelling capabilities for input data outside the training domain [99]. To set up a PINN, the cost function should have a general form as is presented in Equation 2.21 [99].

$$J(\theta) = L_{Data} + L_{Boundary} + L_{Physics} \tag{2.21}$$

In Equation 2.21, L_{Data} represents the loss function of the data points, $L_{Boundary}$ represents the loss introduced by imposing boundary conditions to the computational domain and $L_{Physics}$ represents the loss function which introduces the differential equations to the cost function [99]. It should be noted that not all of the individual loss terms as provided in Equation 2.21 are required for each problem. For

example, PINNs can also be used for training without experimental data, so without the data loss L_{Data}, thereby solely training on the imposed differential equations and their boundary conditions [99].

In subsubsection 2.1.1.5, the cost function making use of the MSE was introduced in Equation 2.9. This cost function can be transformed into a cost function seen in PINNs by adding the physics loss term to Equation 2.9. This results in a cost function as presented in Equation 2.22 [96].

$$J(\theta) = \frac{1}{N}\sum_{i}^{N}\left(f\left(x_i|\theta\right) - y_i\right)^2 + \lambda\frac{1}{M}\sum_{i}^{M}\left(g\left(x_i, f\left(x_i|\theta\right)\right)\right)^2 \tag{2.22}$$

It should be noted that the physics loss does not make use of the experimental data [99]. Instead, a set of collocation points is used as inputs to the NN to evaluate the physics loss over a certain computational domain [99]. The collocation points are points drawn along a certain pre-defined computational domain, which can be different from the domain of the experimental data set [99]. Making use of the collocation points as inputs, output predictions can be obtained, which can be used for the differentiation process of the NN and afterwards be compared to the differential equation of the physical model [99]. This way, the data loss term can be optimised [99].

PINNs are a form of hybrid modelling [60]. This means that a model is created making use of first principle models, such as physics laws, as well as a data-driven model, from a NN [60]. Unfortunately, there are some downsides with PINNs as well compared to the solely first principle models, such as finite difference schemes [74]. The main advantages of PINNs are that they are mesh-free, can be extended to the discovery of parameters, work well on mixed problems and are mostly an unsupervised process [74]. However, a major downside is its high computational time during the training phase compared to a purely data-driven model [74]. Furthermore, the PINN should be trained for every new situation, that is, for each new initial or boundary condition presented to the problem [74]. Besides this, PINNs tend to have a lower convergence rate compared to other methods due to competing terms in the cost function [74]. Lastly, PINNs are currently being used for simplified problems, but they have not been scaled to larger and more complex problems, meaning that PINNs are currently a proof of concept study [74].

Fortunately, some solutions are available for the aforementioned downsides. To decrease computational cost, additional input parameters could be added to the NN, which provide a general way of describing a certain problem [74]. Initially, this network will have a higher computational time compared to the standard network, but will decrease the required modelling time afterwards since the network does not have to be retrained for new situations [74]. This process of generalising a NN for a certain problem is called surrogate modelling [74]. Generalising input parameters of an NN for a certain problem is called conditioning of the network. The difference between a standard PINN and a conditioned PINN architecture is presented in Figure 2.13.

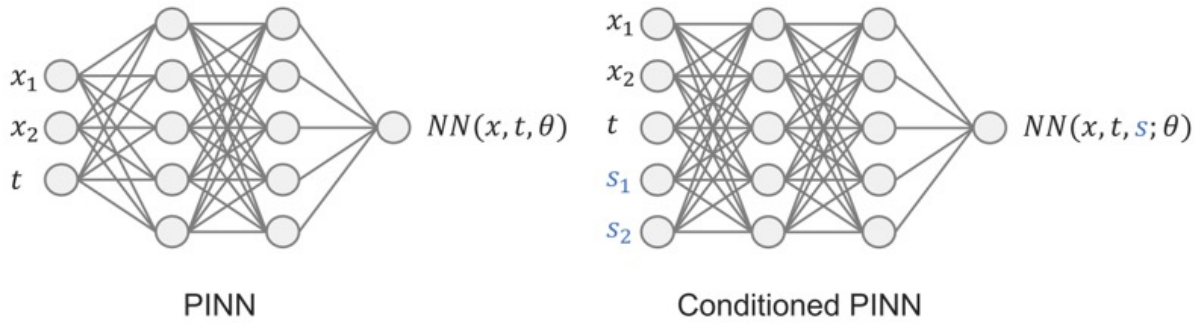


Figure 2.13: An illustration showing the differences in the network architecture of a standard PINN and a conditioned PINN for surrogate modelling [74]

To summarise, PINNs embed both data losses and physics-based losses, including boundary conditions, into their cost function [99]. While purely data-driven models use backpropagation solely to

compute gradients for updating network parameters, as explained in subsubsection 2.1.1.7, PINNs additionally enforce certain partial derivatives to satisfy governing physical equations [99]. This means the network is not only trained to predict function values but also to satisfy derivative constraints imposed by the underlying physical model [99].

2.2. Theory Behind Data Processing

In the real world, signals are considered to be continuous, meaning that a certain signal has a value at every instance in time. However, if a signal is being made digital, this continuous signal can not be stored as a continuous signal, since currently, no method exists to transform a fully continuous signal directly into a digital signal. To solve this issue, the continuous signal will be measured at certain fixed time intervals, meaning that only individual measurements are being stored from the original continuous signal. This results in a discrete signal, which is a signal consisting of individual time step values rather than a continuous measurement. An example of a continuous and discrete signal is presented in Figure 2.14.

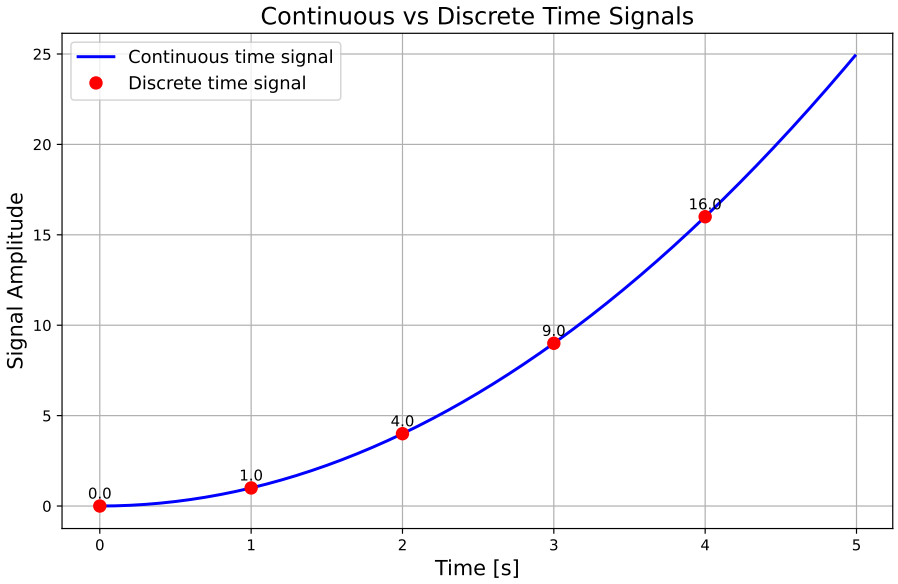


Figure 2.14: Example illustration of a continuous and discrete time signal

If a real-world signal is being measured, the signal has to be transformed into another form before it can be measured. The device which can perform such transformations is called a transducer. A transducer is a tool which can convert one type of signal into another [18]. For example, a pressure sensor can be used to transform the measured air pressure into an electrical signal. This electrical signal is still continuous since all the movements in pressure are being converted continuously. Of course, not all fluctuations in pressure can be transformed by a transducer, since this is dependent on the type of transducer and the way the transducer has been designed [18]. The electrical signal will be stored by measuring the values and converting them into the digital domain, such that they can be stored by a computer [18]. The transformation device converting the electronic signal into the digital domain is called an analogue-to-digital converter, or shortened, A/D converter [18]. The A/D converter measures the electrical signal at fixed time intervals and enables the storage of these values by a computer [18]. The measurements of the continuous signal are being taken a certain number of times per second, to convert the continuous signal into the discrete signal [18]. The rate at which the measurements are being taken is also known as the sampling frequency, or in other words, how many samples are being recorded and stored in 1 second [18]. To measure a signal with a certain frequency, the sampling frequency should be twice the highest frequency of the signal [18]. This frequency is called the Nyquist frequency [18].

2.2.1. Signal Noise Filtering

Real-world signal measurements do not only contain the ideal continuous signal in a discrete representation, but also noise. The noise introduced in this signal originates from various sources, such as electromagnetic interference and the A/D converter, but also from the environment itself [18]. To remove the noise, use can be made of various filtering methods. Certain filtering methods are more suitable for certain applications compared to others, which is dependent on the type and energy level of the noise.

2.2.1.1. Butterworth Filter

To remove high-frequency noise, use can be made of a digital Butterworth filter. The Butterworth filter can act as a low-pass, high-pass or band-pass filter, for which the order of decay and cut-off frequency can be set [18]. An illustration of a Butterworth low-pass filter is presented in Figure 2.15.

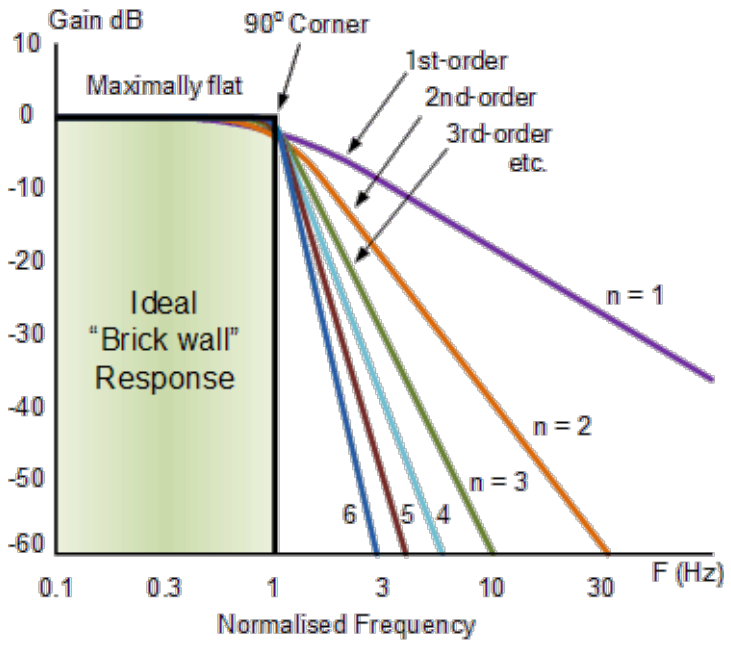


Figure 2.15: Example illustration of a low-pass Butterworth filter [35]

The order of decay represents how much the signal is being reduced per doubling of the frequency [18]. A first-order slope has an order of decay of 6 [dB] per octave, a second-order 12 [dB] per octave. To identify where to position the cut-off frequency and the order of the cut-off filter, use can be made of a Fast Fourier Transform, or shortened FFT. An FFT is a representation of the signal in the frequency domain, showcasing the energy levels of the signal per frequency [18]. The most important features of the measured continuous signal show a large energy peak in the FFT domain, while noise is normally much lower in energy levels [18]. An illustration of a time signal and the corresponding FFT is presented in Figure 2.16.

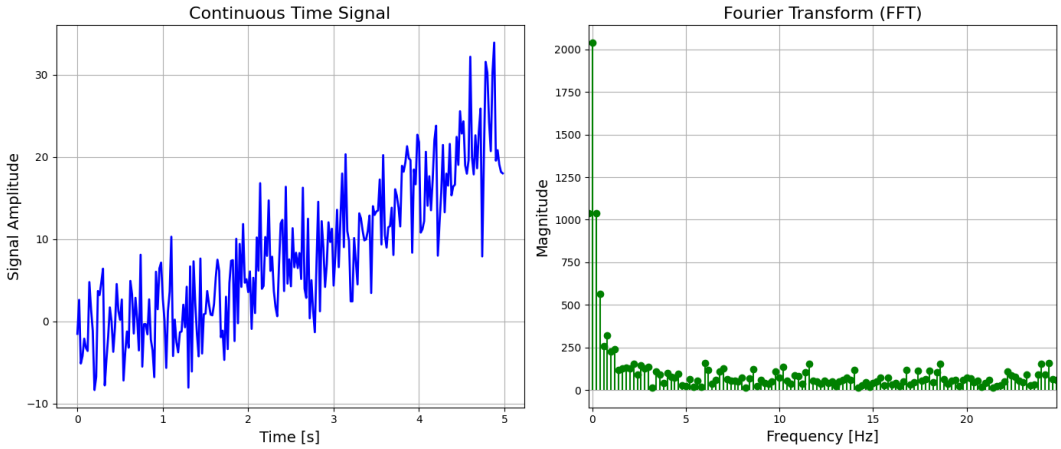


Figure 2.16: Example illustration of an FFT from time domain data including noise

A major downside of using a Butterworth filter is that using this type of filter could add delay to a signal, due to phase shifts introduced during the filtering process [18]. This becomes of interest if various measurements are linked to each other in terms of time, where a certain order and timing of events become important [18]. To solve this issue, use can be made of a linear phase Butterworth filter, which delays all the frequencies to an even amount, such that the overall signal is delayed and all events happen in the correct order and at the correct time [18]. By later removing this delay, the time at which a certain event happens is set at the original time. For low-pass filtering, use can be made of the FFT to determine the parameters for the Butterworth filter [18]. As previously explained, for this filter, an order and cut-off frequency need to be determined. Normally, an order of 3-6 is used for rocket engine test data [3]. Unfortunately, Low-pass filtering has a major downside, namely that important signals at higher frequencies with a relatively low energy level will be filtered out, removing important information from the test data [93].

2.2.1.2. Cross-Correlation Filtering

As was explained in subsubsection 2.2.1.1, a Butterworth filter also removes high-frequency components of the signal when positioning the cut-off frequency. However, some of these frequency components are important to the signal and should therefore not be removed. To be able to keep these components, the FFT of the data signal can be used to remove the unwanted frequencies introduced during the measurements from the experimental data, while keeping the important aspects of the signal. To do so, use could be made of a certain threshold function, filtering out all frequencies below the threshold and keeping all the frequencies above or at this threshold [93]. This way, frequencies containing important information of the experimental data will not be removed [93]. To determine the optimal threshold, use can be made of Spearman's or Pearson's correlation between the original signal and its difference with the filtered signal [93]. This process works as follows. First, a threshold will be positioned, and all the energy located at the frequencies below the threshold will be set to zero [93]. Next, the filtered signal will be subtracted from the original signal to obtain the noise signal [93]. After this, the correlation will be determined between the filtered signal and the removed noise signal [93]. Ideally, there should be no correlation between the filtered signal and the noise signal [93]. In case the correlation is high, this means that certain frequencies of the original data are present in the noise signal, meaning that the threshold is set too low [93]. To provide an example of this procedure, an example signal will be used, as presented in Figure 2.17. The example signal consists of an addition of 2 sine functions, with a frequency of 1 [Hz] and a frequency of 20 [Hz].

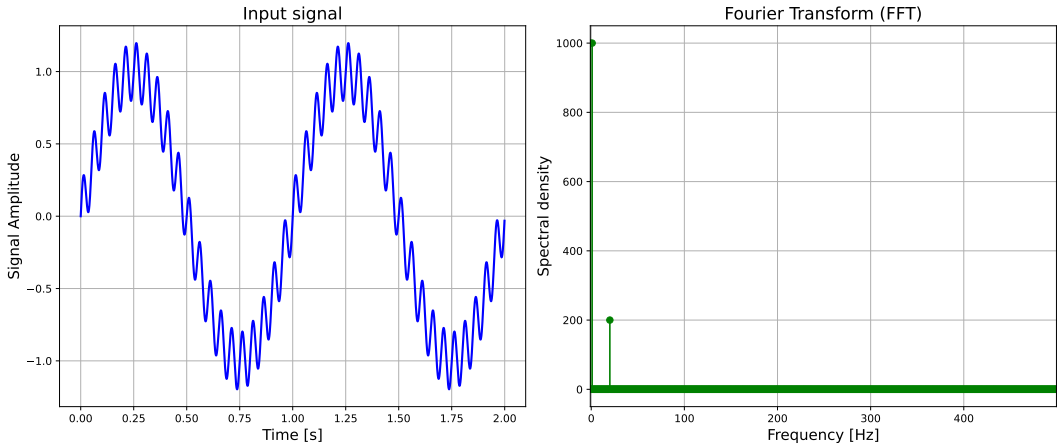


Figure 2.17: Example signal existing out of 2 sine functions, with a frequency of 1 [Hz] and 20 [Hz], in the time and frequency domain

Examining Figure 2.17, from the FFT it can be seen that the example signal indeed consists of a 1 [Hz] and 20 [Hz] component, as 2 energy peaks are visible in the FFT domain. If random noise is being added, as is presented in Figure 2.18, it can be seen that the FFT spectrum has obtained additional frequencies, and the signal exists out of more frequencies compared to the original signal, as presented in Figure 2.17.

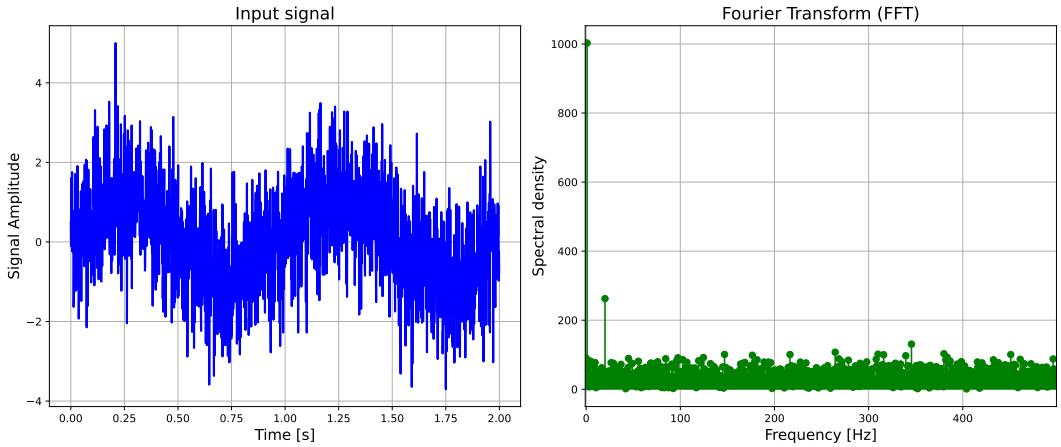


Figure 2.18: Example signal existing out of 2 sine functions, with a frequency of 1 [Hz] and 20 [Hz], and noise in the time and frequency domain

Comparing Figure 2.17 and Figure 2.18, it can be seen that there are still large energy peaks located at the frequencies of which the original signal exist out of. These peaks can be used to filter the signal, making use of a threshold. This works as follows. If a threshold is set at a certain magnitude, the frequencies which are above this threshold will not be removed, while setting the energy levels of the frequencies with a magnitude lower than the threshold to zero [93]. If this filtered signal is then compared to the noise signal, ideally, if the two signals are being subtracted from each other, only noise should remain. This means that there should be no correlation between the subtracted signal and the pre-filtered signal. To determine the correlation between the signals, use can be made of Spearman's or Pearson's correlation. The higher the correlation values, the more signal is still present in the noise signal. So, the threshold should be set at the smallest number possible. Performing the procedure as explained above on the example signal as presented in Figure 2.18, the following result will be obtained for the filtered signal, which is presented in Figure 2.19.

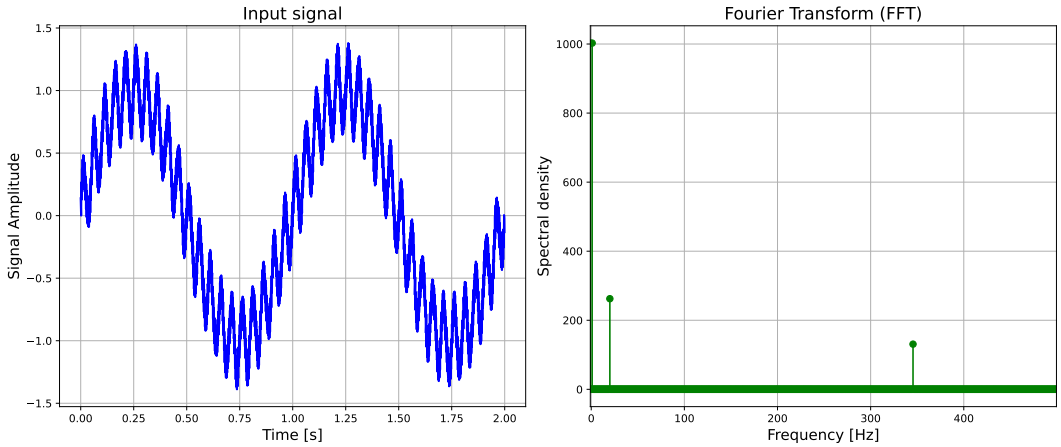


Figure 2.19: Filtered example signal existing out of 2 sine functions, with a frequency of 1 [Hz] and 20 [Hz], in the time and frequency domains

From Figure 2.19, it can be seen that the filtered algorithm using the threshold function based on the correlation almost eliminated all the noise, but still some noise components remain. Unfortunately, it is rarely possible to fully remove the noise, since it is unknown what the original underlying signal looks like [93]. This means that there will always be some parts of noise introduced into the signal. For this specific signal, changing Spearman's correlation with Pearson's correlation results in almost the same signal as presented in Figure 2.17, as is presented in Figure 2.20. Besides the remainder of the noise, the filtered signal also slightly deviates from the original signal, due to the filtering process.

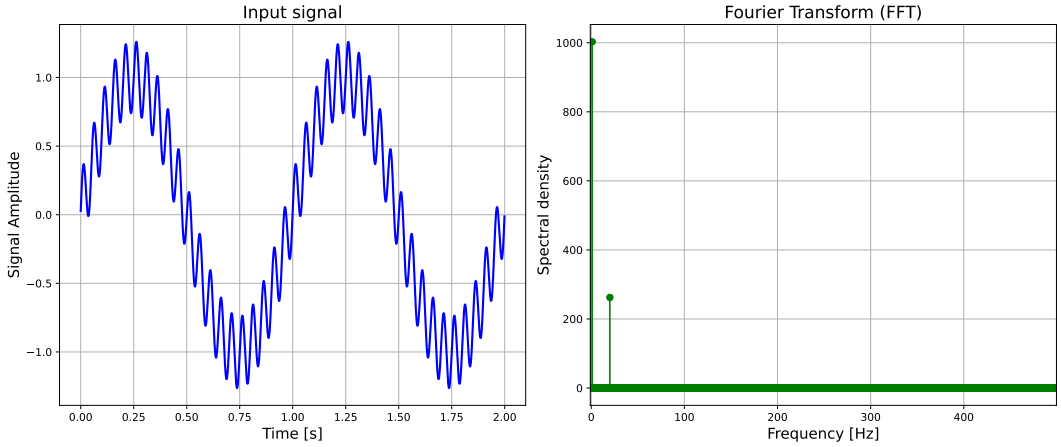


Figure 2.20: Filtered example signal existing out of 2 sinus functions in the time and frequency domains using Pearson's correlation

The decision to use either Pearson's or Spearman's correlation is dependent on how the signal behaves in the time domain. Pearson's correlation is mostly used for linear signals, while Spearman's correlation is mostly used for non-linear signals [100].

2.2.1.3. Savitzky-Golay Filtering

Another method to keep the high-frequency components of a signal is to make use of a Savitzky-Golay filter. This filtering method can be used to reduce high-frequency noise in a signal and reduce low-frequency signals, making use of smoothing and differentiation [41]. Besides this, the filtering method does not need to make use of an FFT to filter the data [41]. A Savitzky-Golay filter works as follows. For a given signal, the filter makes use of a certain number of points inside a window to fit a polynomial of a certain pre-defined order [41]. In the middle of the domain, a new value of a data point

is then estimated, making use of this fitted polynomial [41]. This process is then repeated by moving the window throughout the entire domain [41]. A graphical example of this process is presented in Figure 2.21.

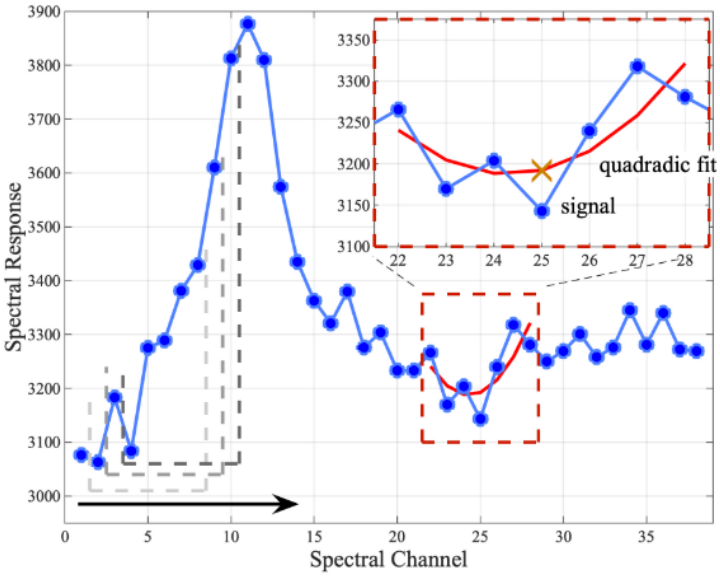


Figure 2.21: Example of the Savitzky-Golay filtering process, with a window length of 7 points [41]

The filter has the noise data as input, the window length and the polynomial order [41]. To obtain the optimal window length, it is first required to note that the window length should be larger than the polynomial order, as well as setting the window length equal to an odd number [41]. So, for example, if the polynomial order is 2, the window length should be at least 3 points. In general, the window length should be set in such a way that it can maintain important parts of the signal [41]. For example, if there are peaks in the signal, one could set the window length equal to the width of the peak to maintain the information of the peak [41]. The order of the polynomial has to be set according to the signal [41]. A low-order polynomial has to be set for linear signals with little variation in the signal, while higher-order polynomials could capture rapid variations in the signal, making them more suitable for these types of signals [41]. In general, it is advised to set the window length between 5 and 7 and the order to 2 or 3 [105].

2.3. Review of Statistics

When performing analysis, it is important to understand how generalisable a certain observation is. This means that the observation has not been observed by accident, but that, independent of the number of samples taken from a group, the same outcome would be observed. To determine if this is the case for a certain observation, statistics can be used to draw proper conclusions. To do so, firstly, it should be determined if a certain set of data follows a normal distribution. A normal distribution is a continuous probability distribution for random values [90]. This means that it will be determined how often a certain value can be expected when drawing random samples from a dataset [90]. The normal distribution can be represented using the probability density function, which can be seen in Figure 2.22. The middle value, μ, represents the mean of all the observed samples. σ represents the standard deviations, which is a measure of the amount of variation for a given mean [90]. The probability density function of a normal distribution has been made in such a way that 68% of the samples are within 1 standard deviation, 95% within 2 standard deviations and 99.7% within 3 standard deviations [90].

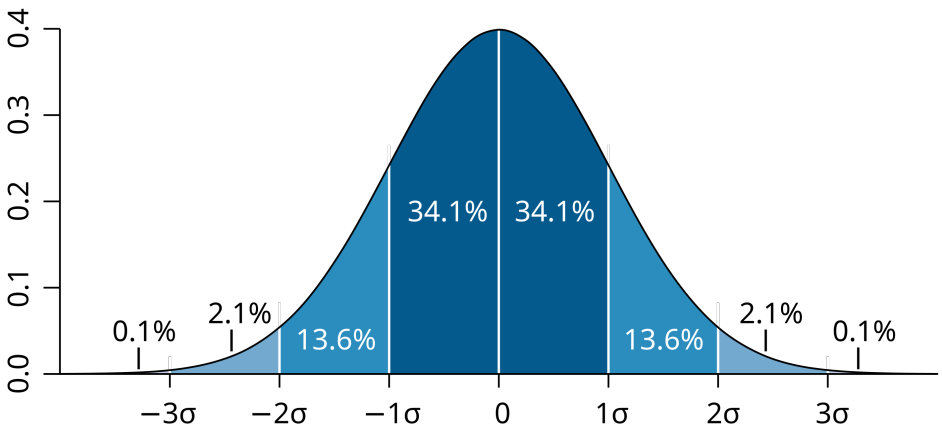


Figure 2.22: An illustration of a probability density function of a normal distribution [90]

In case the drawn samples from a group follow a normal distribution, use can be made of a variety of tests to draw relevant conclusions, also called a test statistic [111]. This works as follows. First, a certain null hypothesis, H_0, needs to be defined against which the specific test will be held [111]. After this, a certain threshold needs to be determined to know the value at which the null hypothesis will be rejected [111]. Then, a test will be performed, after which it will be determined if the null hypothesis will be rejected or not [111].

However, sometimes the generalised trend does not follow a normal distribution. To check if this is the case, the Shapiro-Wilk test [106] can be used. This metric tests if a certain drawn sample comes from a normal distribution [106]. If this is not the case, the null hypothesis will be rejected, and the sample does not belong to a normally distributed group [106].

To perform statistics on differences between values, use can be made of the paired t-test can be used as a test statistic in case the sample follows a normal distribution [94]. In case it was concluded that the sample does not follow a normal distribution, use can be made of the Wilcoxon signed-rank test as test statistic [136]. In case the null hypothesis is accepted, the claim is said to be statistically significant, while if the null hypothesis is rejected, the claim is not statistically significant [112].

2.4. Overview of Sources

Throughout the literature review, use has been made of various sources. To provide an overview of the sources and their domain, the sources for each part are summarised and presented in Table 2.1.

Table 2.1: Overview of sources and their usage in the literature review

Section	References
Artificial Intelligence	
Introduction	[133], [8]
Machine Learning	[66], [116], [25]
Artificial Neurons	[15], [132], [46], [33], [121]
Activation Function	[33], [4], [127]
The Learning Process	[46], [72]
Weight and Bias Initialization	[20]
Cost Function	[71], [120], [138], [96]
Neural Network	[46], [75], [51], [59]
Optimizer	[46], [29], [101], [7], [134], [109]
Automatic Differentiation	[95], [1], [124], [32], [16]
Problems During the Training Phase	[134], [109]
Regularization	[109], [96]
Data Scaling	[107], [127], [31]
Data Structure and Assessment of Performance	[21]
Design of a Neural Network Architecture	[44], [51], [127]
Types of Neural Networks	[46], [44], [108]
Verification and Validation for Machine Learning Algorithms and Neural Networks	[82], [126], [128], [24]
Physics-Informed Neural Networks	[99], [96], [60], [74]
Theory Behind Data Processing	
Introduction	[18]
Butterworth Filter	[18], [35], [3], [93]
Cross-Correlation Filtering	[93], [100]
Filtering Using Savitzky-Golay Filter	[41], [105]
Review of Statistics	
Introduction	[90], [111], [106], [94], [136], [112]

3

Developed Software Class

In section 2.1, it was explained what a PINN is, how it differs from standard NNs and what the main building blocks of PINNs and NNs are. At its core, NNs exist out of artificial neurons, which have a linear equation containing a weight and a bias, which will together with the input, after being multiplied by the weight and the bias being added to this, are passed on to an activation function to generate an output. This process can be implemented in code, making use of the Python coding language and additional libraries, which improve performance and decrease development time compared to coding the main building blocks and ideas from scratch. To decrease development time for the generation of PINNs and make code reusable, it was decided to make a special NN class which can quickly change network configurations and modify the loss functions in the main cost function. To do so, the PyTorch library is used, which is an open-source Python library which could be used for the development of deep learning applications [98].

After careful evaluation of the various types of NNs and how to design network architectures for data-driven NNs and PINNs, as described in subsubsection 2.1.1.13 and subsubsection 2.1.1.14, it was decided to make use of an FNN. This decision is based on the promising results which have been obtained by making use of these types of NN when developing PINNs [99, 12]. The general setup of an FNN was encoded into a class for enhancing development and iteration purposes. To do so, the NN class can easily change the number of input neurons, output neurons, hidden layers and number of neurons in these hidden layers, as well as the type of activation function, initialisation method and adapt the cost function by changing the amount or types of loss functions. A flow diagram of this NN class, including its inputs and outputs and internal structure, is presented in Figure 3.1

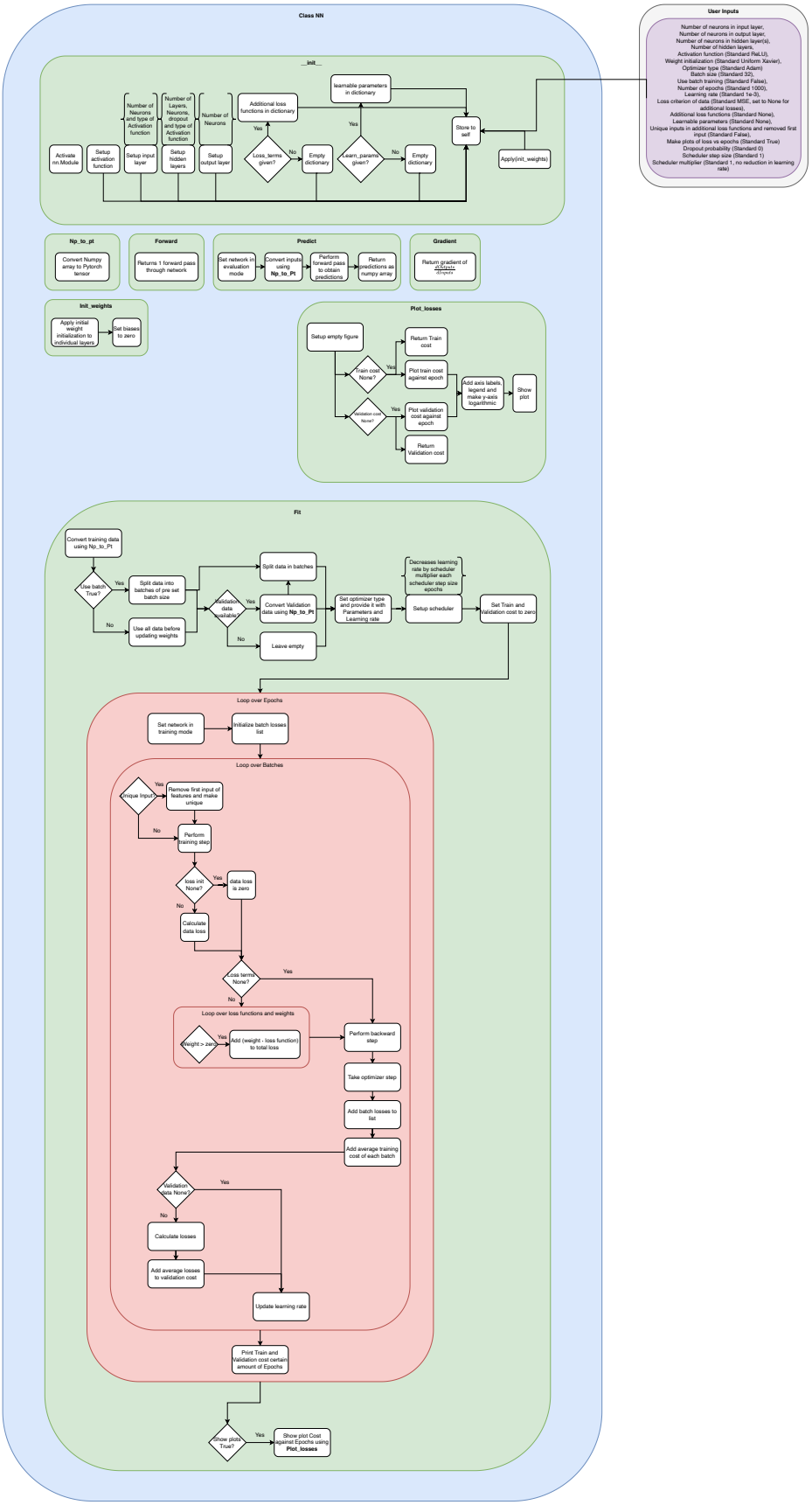


Figure 3.1: Block diagram of the developed class NN during this thesis

3.1. Explanation and Manual for the NN Class

Although Figure 3.1 provides an overview of the developed class for this thesis, it does not explain how to use it for the development of NNs or PINNs. This section will explain how to use the NN class and what its limitations are.

Firstly, it should be mentioned that this class can only be used by making use of Python as the coding language. The NN class can be imported by making use of the command as presented in Code 3.1. However, it should be made clear that this only works in case the Python file containing the NN class is stored in the same working directory. Besides the NN module, the gradient module will also be imported. This part is required to be able to develop PINNs in a later stage. Besides the NN_tools_thesis, other functions should also be imported, to be able to change activation functions, initialisation methods and optimisers, as well as define additional physics and boundary condition losses.

Code 3.1: Importing NN class in new Python File

```
from NN_tools_thesis import NN, gradient
import torch.nn as nn
import torch.optim as optim
import torch
```

After all tools are imported, an NN can be set up by calling the NN function. To show how this works with the setup of the class, the NN class will be initialised as an example NN, which is presented in Code 3.2.

Code 3.2: Example initialization of the NN class

```
NN_example = NN(N_INPUT, N_OUTPUT, N_HIDDEN, N_LAYERS, activation_fn=nn.ReLU, init_method=nn.
    init.xavier_uniform_, optimizer_type=optim.Adam, batch_size=32, use_batch=False, epochs
    =1000, lr=1e-3, loss_init=nn.MSELoss(), loss_terms=None, learn_params=None, unique_input=
    False, show_plots=True, dropout_prob=0, scheduler_step_size=1, scheduler_gamma=1)
```

The inputs of the NN function are listed below, together with an explanation of the definition for each input and the data types used.

- N_INPUT : int
  Number of input neurons of the Neural Network.
- N_OUTPUT : int
  Number of output neurons of the Neural Network.
- N_HIDDEN : int
  Amount of Neurons per hidden layer.
- N_LAYERS : int
  Number of hidden layers.
- activation_fn : nn.<activation_function>, optional
  Type of activation function used in the input and hidden layers. The default is nn.ReLU. The output layer makes use of a linear output. To be able to change this parameter, torch.nn should be imported.
- init_method : nn.<init_method>, optional
  Type of weight initialisation method. The default is nn.init.xavier_uniform_. To be able to change this parameter, torch.nn should be imported.
- optimizer_type: optim.<optimizer_type>, optional
  Type of optimiser used during the training process. The default is optim.Adam. To be able to change this parameter, torch.optim should be imported.
- batch_size : int, optional
  Batch size. The default value is set to 32.
- use_batch : Boolean, optional
  Define if you would like to use batch training. The default is False.
- epochs: int, optional
  The number of epochs used during training. The default value is set to 1000.

- lr: float, optional Learning rate. The default value is set to 1e-3.
- loss_init : nn.<optimizer_type>, optional
  Defines the type of loss metric used in the cost function for the data loss. If you would like to train the Neural Network without (experimental) data points, such that you only train on a physics loss, select None. The default is nn.MSELoss(). To be able to change this parameter, torch.nn should be imported.
- loss_terms : Dictionary, optional
  Insert additional loss terms, such as physics and boundary loss. The default is None. The data loss term is standard, embedded inside the NN class.
- learn_params : Dictionary, optional
  Insert learnable parameters into the network for parameter discovery. The default is None.
- unique_input : Boolean, optional
  Define if you would like to have unique inputs for the physics loss when using various inputs to the training loss. The default is False.
- show_plots : Boolean, optional
  Define if you would like to have a plot of the training and validation losses per epoch once the training process is finished. The default is True.
- dropout_prob : Int, optional
  Defines the probability of neuron dropout within the hidden layers during the training process. The default value is set to 0.
- scheduler_step_size: Int, optional
  Defines the scheduler step size during the training process. This defines how many epochs are required before the learning rate is decreased by a certain amount. The default value is set to 1.
- scheduler_gamma: Float, optional
  Defines how much the learning rate is decreased each step size by the scheduler. This works as follows: $lr_{(i+1)} = lr_{(i)} * \gamma$. The default value is set to 1, meaning that there is no decrease in learning rate during the training process.

Now the class has been initialised, a PINN can be developed making use of the NN class. To do so, a physics loss and/or boundary loss needs to be implemented. The setup of this process is presented in Code 3.3. It should be noted that used_inputs can be used to obtain the inputs of the class. This could be useful during the development of surrogate models.

Code 3.3: Example initialization of the NN class

```
def PINN_loss_example(NN_output: torch.nn.Module, used_inputs):

    # Obtain outputs from the network based on some inputs
    inputs = used_inputs

    current_PINN_prediction = NN_output(inputs)

    # Obtain derivative from the network
    d_output_d_input = gradient(output, input)

    # Physics or boundary loss implementation
    #
    #

    # Calculate loss using a certain metric (MSE in this case)
    loss = torch.mean((d_output_d_input - d_output_d_input_theory)**2)

    return loss
```

From Code 3.3, it can be seen that the inputs provided to the network as training data can also be used in the additional loss functions. However, this is not required, and newly generated inputs, such as longer domain lengths, can be created and used as well. The earlier-mentioned gradient module is used in this part of the code to obtain the derivative of a certain input and output. Lastly, the Torch module is required to specify a certain loss metric. Since tensors are being used, specifying these

operations in PyTorch is more efficient. Finally, the loss is returned, which will be used in the total cost function of the NN class, which is presented in Figure 3.1. To add this loss to the NN class, a dictionary is used. The setup of this dictionary is presented in Code 3.4.

Code 3.4: Setup of dictionary for loss functions to pass to the NN class

```
loss_functions_dict = {
    PINN_loss_example: 1
}
```

Firstly, from Code 3.4, it can be seen that, besides the name of the loss function, a number is specified as well. This number is used to be able to scale the individual loss in the total cost function, as was explained in subsection 2.1.2. This process is enabled such that the PINN will not only take the data loss or physics loss into account, but both. A possibility exists that, in case one of the individual loss terms is larger in magnitude compared to the other, only a single loss function will be optimised, rather than multiple losses.

Another use case of the class is that the class can be used as a discovery network or I-PINN. To do so, a certain parameter of a differential equation should be implemented as a parameter of the network. To do so, the learnable parameter should be passed to the network as a library as well, an operation which is presented in Code 3.5. Besides the parameter, the second value present in the library is the initial guess for the parameter, which is different compared to the library with loss functions passed to the network. The closer the specified value is to the real-world value, the better the network is able to obtain the most suitable value for the differential equations of the physical model, obtained from the provided experimental data. Therefore, it is advised to provide known values of similar problems in the literature as initial values in the library.

Code 3.5: Setup of dictionary for parameters for discovery to pass to the NN class

```
learnable_parameters_dict = {
    "X": 0.1
}
```

After this, these parameters can be used in the additional physics loss as is presented in Code 3.6.

Code 3.6: Example initialization of the NN class for parameter discovery

```
def PINN_loss_example_discovery(NN_output: torch.nn.Module, used_inputs):

    # Obtain outputs from the network based on some inputs and learnable parameter
    inputs = used_inputs

    current_PINN_prediction = NN_output(inputs)
    current_X = NN_output.X

    # Obtain derivative from the network
    d_output_d_input = gradient(output, input)

    # Physics or boundary loss implementation
    #
    #

    # Calculate loss using a certain metric (MSE in this case)
    loss = torch.mean((d_output_d_input - d_output_d_input_theory)**2)

    return loss
```

It should be noted that the loss functions and learnable parameters need to be set up before the initialisation of the NN class network, such that they can be passed as parameters to the network.

The NN class can also be used as a purely data-driven NN. This can be done by adding or leaving out additional terms in the loss functions, as was previously explained. However, although the NN class is quite flexible, there are some limitations. Using this class, only FNNs can be created, and not other network types. Besides this, the initialisation methods and optimiser can only be passed in standard form and can not be changed. This means that individual values can not be directly set. Lastly, the scheduler is set to a step learning rate, and can not be changed for other scheduler types.

Part I

Cooling Down of Objects Modelling

4

Introduction: Cooling Down of Objects

To test out the capabilities and identify if a PINN does improve computational time and data efficiency, providing similar accuracy compared to other models and if a PINN can be used as a surrogate modelling tool, a well-known test case could be used. This test case should be described by a certain differential equation (either partial or ordinary), and a set of experimental data points which can be used to train the PINN and perform validation [99]. To explore this entire cycle, it was decided to make use of a heat transfer problem, namely the cooling down of an object by convection. In aerospace, heat transfer is an important aspect in a wide variety of domains. Interesting domains to investigate are heat transfer of materials during re-entry, supersonic flight or spacecraft during space operations. For this part of the research, it was decided to investigate the cooling down of objects made of various materials by natural convection. The chosen materials are glass and clay cups, which will be heated up by making use of hot water. The heat transfer process can be described by making use of Newton's law of cooling [88], which describes the heat transfer problem by making use of a single ordinary differential equation. Rather than making use of dummy data, experimental data of this process will be measured, thereby meeting all the requirements to be able to generate a PINN model. To summarise, this part of the thesis will investigate the computational and data efficiency, as well as the accuracy of a PINN model compared to a numerical and purely data-driven model for a heat transfer problem. Furthermore, this part will investigate the capability of surrogate modelling of PINNs compared to other modelling methods.

To aid in answering the main research question as provided in chapter 1, this part of the report will focus on the following sub-research questions:

1. *To what extent can a hybrid model, using a Physics-Informed Neural Network, improve the accuracy and computational efficiency of an object's surface temperature predictions compared to a Numerical method?*
2. *To what extent can a hybrid model, using a Physics-Informed Neural Network, improve the accuracy and data efficiency of an object's surface temperature predictions compared to a purely Data-Driven method?*
3. *To what extent does treating the convective heat transfer coefficient as a learnable parameter in a Physics-Informed Neural Network affect the accuracy, data efficiency, and computational efficiency of an object's surface temperature predictions, compared to a Numerical and purely Data-Driven method?*
4. *To what extent can a Physics-Informed Neural Network accurately predict an object's surface temperature when presented with new design parameters (for example, object's geometry or ambient conditions) not seen during training, compared to a Numerical and purely Data-Driven method, when validated against experimental data? What are potential limitations?*

This part of the report is structured as follows. First, a brief literature review will be presented in chapter 5, to explain important concepts of heat transfer mechanisms and obtain the required physical models to describe the heat transfer mechanism related to convective heat transfer. Next, the process of setting up the experiment and obtaining and processing the experimental data related to this use case will be presented in chapter 6. After this, a numerical model will be presented in chapter 7. Next, the generated PINN will be discussed in chapter 8. Lastly, the setup of the tests to assess the performance of the different modelling methods and the results of these tests will be discussed in chapter 9.

5

Literature Review: Cooling Down of Objects

In this chapter, the required background information related to the cooling down of objects will be examined. First, a general overview of heat transfer mechanisms will be explained in section 5.1. After this, Newton's Law of Cooling, a differential equation which can be used to make predictions for objects which are cooling down due to natural convection, will be derived and explained in section 5.2. Lastly, an overview of the sources used during this part of the literature review will be presented in section 5.3.

5.1. Heat Transfer Mechanisms

The process of objects cooling down due to natural convection can be described by investigating heat transfer mechanisms [57]. Overall, there are three main types of heat transfer mechanisms, which are presented and explained in the list below [57]:

- Conduction
  When nearby atoms or molecules collide, heat is transferred from one to the other [57]. This process is called conduction, and can only happen when matter is in direct contact with each other [57]. In solids and liquids, conduction happens more easily compared to gases, since their particles are closer to each other [57].
- Convection
  Inside gases and liquids, large-scale movements of molecules force heat away, which is the main driving force of the convective heat transfer mechanism [57]. Overall, there are two main types of convection. Natural convection is convection due to the absence of external forces, where buoyancy is the driving factor [57]. This principle is based on the fact that higher temperatures will rise and colder temperatures will go down [57]. On the other hand, forced convection is due to an external force, such as a pump or fan, which forces the movement [57].
- Radiation
  Radiative heat transfer moves heat in the form of electromagnetic waves [57]. This type of heat transfer mechanism does not need a medium, meaning that it is the primary source of heat transfer in vacuum conditions [57].

In real-world operations, heat transfer mechanisms mostly appear in a combined fashion, meaning that multiple heat transfer mechanisms are active at the same time [57]. For experiments related to investigating an individual mechanism, it is important to limit the other mechanisms as much as possible, by, for example, isolating surfaces and preventing conductive heat transfer [57].

5.2. Newton's Law of Cooling

In 1701, Isaac Newton came up with a mathematical way of describing convective heat transfer, enabling a method to come up with predictions of future temperature states of matter due to convection [88]. In this section, Newton's Law of Cooling will be examined. First, the derivation will be examined in subsection 5.2.1. Next, assumptions and limitations of Newton's Law of Cooling will be explored in subsection 5.2.2. After this, the applicability of Newton's Law of Cooling will be examined in subsection 5.2.3, to investigate under which situations this equation can be used. Lastly, various constants for Newton's Law of Cooling will be provided, as well as various material properties for the to-be-conducted experiments in chapter 6.

5.2.1. Derivation of Newton's Law of Cooling

To derive Newton's cooling law, use can be made of the equations as presented in Equation 5.1 [88]. These equations present the rate of heat transfer out of a body and the change in internal energy of a lumped capacitance object [88].

$$\begin{aligned} \frac{dQ}{dt} &= hA\left(T\left(t\right) - T_{env}\right) \\ \frac{dU}{dt} &= mc_p\frac{T\left(t\right)}{dt} \end{aligned} \tag{5.1}$$

In Equation 5.1, Q represents the heat flux transferred out of the body, t represents time, h represents the heat transfer coefficient, A represents the heat transfer surface area, T represents the temperature of the objects surface, T_{env} represents the temperature of the environment surrounding the object, U represents the internal energy, m represents the mass of the object and c_p represents the specific heat capacity [88]. Making use of the first law of thermodynamics for the lumped body, Newton's Law of Cooling in differential form can be obtained, which is presented in Equation 5.2 [88].

$$\frac{dT\left(t\right)}{dt} = \frac{hA}{mc_p}\left(T_{env} - T\left(t\right)\right) \tag{5.2}$$

5.2.2. Assumptions and Limitations of Newton's Law of Cooling

When deriving Newton's Law of Cooling, various assumptions were made to simplify the analysis. The used assumptions are summarised in the list below [88]:

- The temperature difference is small
- The nature of the heat transfer mechanism remains constant
- Convection is the main driver of removing heat from the system
- The heat transfer coefficient is independent of the temperature difference between the object and the environment
- The object has a single uniform temperature
- The body has a constant heat capacity
- The environmental temperature surrounding the object is constant in time

5.2.3. Applicability of Newton's Law of Cooling

To identify if use can be made of Newton's Law of Cooling, the Biot number can be used [88]. The Biot number is the ratio of thermal resistance of conduction inside a body to the resistance of convection at the surface of the body [19]. If the Biot number is smaller than 0.1, Newton's Law of Cooling can be used, as it is expected that predictions made using this method have a small percentage of error compared to real-world observations [88]. This means that the heat conduction inside the object is much faster than the heat convection away from the surface of the object, resulting in negligible temperature gradients inside the object [88]. This leads to the fact that the assumption can be made that the object has a single uniform temperature [88]. To calculate the Biot number, use can be made of Equation 5.3 [19].

$$Bi = \frac{hL_c}{k_b} \tag{5.3}$$

In Equation 5.3, h represents the convective heat transfer coefficient, L_c represents the characteristic length, which is defined as $L_c = \frac{V_{body}}{A_{surface}}$ where V_{body} represents the volume of the objects body and $A_{surface}$ represents the surface area of the objects body and k_b represents the thermal conductivity of the body.

5.2.4. Constants of Various Materials for Newton's Law of Cooling

To obtain the Biot number and use Newton's Law of Cooling, use should be made of various constants. These constants are mostly material properties, meaning that the constants will differ for different material types [88]. In Table 5.1, the constants to obtain the Biot number and the temperature predictions, making use of Newton's Law of Cooling for various materials, are presented. It should be noted that the thermal conductivity increases with temperature, which positively affects the applicability of Newton's Law of Cooling, as the Biot number decreases.

Table 5.1: Constants for various materials to be used in Newton's Law of Cooling and Biot's Number calculations at 297 [K] for thermal conductivities and 298 [K] for specific heat capacity [119, 63, 47]

Quantity	Value	Unit
Glass		
c_p	0.84	[J/gK]
k_b	0.8	[W/mK]
Clay		
c_p	0.8	[J/gK]
k_b	0.8	[W/mK]
Air		
h (Natural Convection)	5	[W/m^{2}K]
h (Forced Convection)	100	[W/m^{2}K]

5.3. Overview of Sources

Throughout the literature review related to the cooling down of objects, use have been made of various sources. To provide an overview of the sources and their domain, the sources have been structured in Table 5.2.

Table 5.2: Overview of sources and their usage in the literature review

Section	References
Cooling Down of Objects	
Heat Transfer Mechanisms	[57]
Derivation of Newton's Law of Cooling	[88]
Assumptions and Limitations of Newton's Law of Cooling	[88]
Applicability of Newton's Law of Cooling	[88], [19]
Constants of Various Materials for Newton's Law of Cooling	[88], [119], [63], [47]

6

Experimental Setup and Data Processing: Cooling Down of Objects

Experimental data should be available to create and generate a PINN to model a heat transfer mechanism problem related to the cooling down of an object. Since Newton's Law of Cooling only applies to certain situations, it was decided to perform some specifically tailored experiments to obtain applicable data sets. As explained in subsection 5.2.3 and subsection 5.2.2, Newton's Law of Cooling is only applicable to situations related to problems where convection is the main driver of heat transfer, and the Biot number has a value which is less than 0.1. After careful evaluation of these requirements, it was decided to perform an experiment that tracked the temperature of various cups, which would cool down after hot water was placed inside them. This chapter is structured as follows. First, the setup of the experiment and the goals will be examined in section 6.1. Next, the obtained filtered experimental data will be presented in section 6.2. Lastly, the partitioning of the various obtained experimental data files for surrogate modelling of the PINN will be explained in section 6.3.

6.1. Setup and Explanation of the Experiment

To be able to perform the experiments related to the cooling down of a cup, it is important to adhere to the aforementioned requirements and limitations. Since convection should be the main driver of heat transfer to be applicable for Newton's Law of Cooling, the cups will be isolated from the top and bottom to make sure that conduction will not be a main driver of heat transfer and to make sure that heat transfer will mainly occur from the sides. Unfortunately, heat transfer due to radiation cannot be easily limited, and it was decided not to make use of preventive measures. The main goal of this part of the report is to examine whether PINNs are suitable for surrogate modelling, and how their performance relates in terms of computational and data efficiency as well as accuracy compared to other methods. To do so, experimental data of various types of cups will be recorded to see if the model can be used for surrogate modelling. For this experiment, it was decided to make use of cups existing out of 2 different material types, make use of 2 different ambient temperature conditions, make use of cups with 2 different diameters and 4 different lengths. To get an indication of whether Newton's Law of Cooling is applicable, use can be made of the Biot number, as was previously explained in subsection 5.2.3. If the Biot number is smaller than 0.1, it was found in subsection 5.2.3 that Newton's Law of Cooling would be sufficiently applicable for that specific situation. To determine if this would be the case for the various experiments, the Biot numbers of the to-be-used cups will be calculated. In Table 6.1, properties of the used cups are presented. Making use of these properties, the Biot number can be calculated by making use of Equation 5.3 and the constants provided in Table 5.1.

Table 6.1: Properties of the various cups used in the experiments

Quantity	Value	Unit
White cup		
Material	Clay	[-]
D	67.7 ± 0.2	[mm]
L	112.0 ± 0.2	[mm]
L_{eff}	72.8 ± 0.2	[mm]
m	268	[g]
Biot number	0.0255	[-]
Long 18563		
Material	Glass	[-]
D	65.4 ± 0.2	[mm]
L	159.2 ± 0.2	[mm]
L_{eff}	129.5 ± 0.2	[mm]
m	134	[g]
Biot number	0.0204	[-]
Medium 18562		
Material	Glass	[-]
D	65.2 ± 0.2	[mm]
L	119.2 ± 0.2	[mm]
L_{eff}	87.7 ± 0.2	[mm]
m	108	[g]
Biot number	0.0204	[-]
Short 18560		
Material	Glass	[-]
D	65.2 ± 0.2	[mm]
L	70.8 ± 0.2	[mm]
L_{eff}	41.6 ± 0.2	[mm]
m	68	[g]
Biot number	0.0204	[-]

From Table 6.1, it can be seen that all objects have Biot numbers less than 0.1, meaning that Newton's Cooling Law would be an applicable modelling method for the experimental test. The experimental procedure will be as follows. In total, 8 experiments will be conducted. For each object, there will be 2 measurements, namely for different ambient temperatures. To sum up, this means that the main varying factors for the experiments are:

- Material of the object
- Length of the object
- Diameter of the object
- Ambient temperature

The test procedure will be as follows. First, the object will be positioned on the test stand with a styrofoam pad on its bottom connected to it. Next, the temperature sensor will be positioned in the middle of the object's side and will be held in place making use of tape. After this, the setup will be tested to see if the ambient and temperature sensors provide sufficient readings and if everything works as intended. If this is the case, the measurement will start. Next, the hot water will be obtained by making use of a coffee machine and poured into the cup. Next, the top of the object will be sealed off by making use of the styrofoam lid. The measurement will stop if the temperature sensor has a value close to the ambient temperature sensor. It was decided to stop the test if the sensors output a temperature of approximately ±50% of the ambient temperature. The reason why this threshold was chosen was to decrease the individual measurement time per test, as some tests took as long as 3.5 hours. After the test was concluded, the styrofoam top and bottom will be removed from the cup, as well as the temperature sensor, whereafter the wet mass of the object will be determined, making use of

a scale. This measurement will be recorded, whereafter the recorded ambient and object temperature will be stored as a .csv file. Lastly, the experiment is concluded, and a new measurement sequence can be initiated.

It should be mentioned that although there was a stopping condition implemented for the test, this did not decrease the quality of the experiment. The relatively long time for each experiment provided a sufficient amount of experimental data for the training and performance estimation of the PINN and other methods. However, it should be noted that, since no data is obtained to reach a steady state situation, the performance of the various modelling methods in this regime can not be determined, meaning that any predictions made using this method need to be validated before using this model to make predictions inside this regime.

6.1.1. Required Equipment and Experimental Setup

During this experiment, various equipment and materials have been used. The used materials during the experiment are presented in Table 6.2. A schematic overview of the test setup is presented in Figure 6.1. It should be noted that the top and bottom of the cup are isolated from the environment, to reduce the conductive heat transfer mechanism, as well as to force the heat transfer mechanisms to only be present on the sides, as explained in section 6.1. In Figure 6.2, the electrical connections between the Arduino Uno and the sensors are presented.

Table 6.2: Materials used during the cooling down of objects experiments

ID	Amount	Object	Purpose
TE-01	1	White cup	Test object
TE-02	1	Long 18563 glass	Test object
TE-03	1	Medium 18562 glass	Test object
TE-04	1	Small 1860 glass	Test object
TE-05	1	Arduino Uno	Converts temperature sensor data from the surface of the object and data of the ambient temperature sensor from electrical signals into digital temperature signals
TE-06	1	Temperature sensor (DS18B20)	Positioned on the test object's surface to measure the temperature of the outside of the object, which provides the experimental data for the later-to-be-developed PINN. The datasheet of the DS18B20 sensor is presented in Appendix C.
TE-07	1	Temperature sensor (DHT 11)	Positioned away from the test object to measure the ambient temperature of the test environment. Required for Newton's Law of Cooling (see Equation 5.2). The datasheet of the DHT 11 sensor is presented in Appendix B.
TE-08	1	Scale	Measures test objects' dry mass before the test and wet mass after the test. Required for Newton's Law of Cooling (see Equation 5.2).
TE-09	1	Calliper	To be able to measure the objects' diameters and lengths.
TE-10	1	Camera	Take pictures of the test setup for the report.
TE-11	1	Laptop	Convert measured voltages from sensors into degrees Celsius and provide real-time measurements to check for deviations. Stores data after the test.
TE-12	1	Tape	Connect temperature sensor to test object.
TE-13	3	Styrofoam	Create top and bottom isolation pads for test objects to limit other heat transfer mechanisms, mainly conductive heat transfer.

In Table 6.2, it is mentioned that the wet mass is measured after the test has been performed, not before. The reason for this is that, since the hot water which will be poured into the cup to heat the cup will come from an external source, measuring the cup's weight and not starting the experiment yet will lead to a loss of valuable data in the higher temperature regions. Furthermore, if the object's wet mass is measured while the water is still hot and the top is not insulated, the mass will slightly decrease during the measurement due to evaporation. By measuring the wet mass after the test, the evaporated water could condense back into the object, leading to a more accurate measurement. It is expected that the impact of the evaporated water will be negligible, but measuring the wet mass after the test will decrease the likelihood of a large impact.

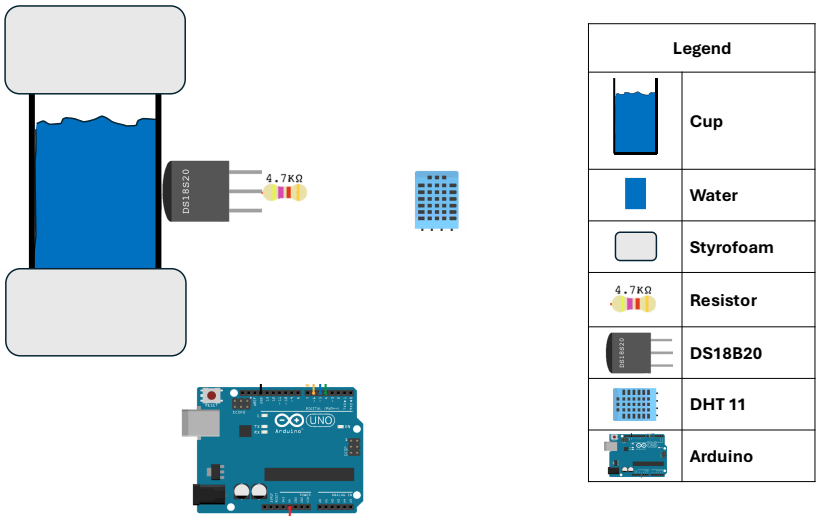


Figure 6.1: Schematic of the experimental setup used for the experiments related to the cooling down of objects

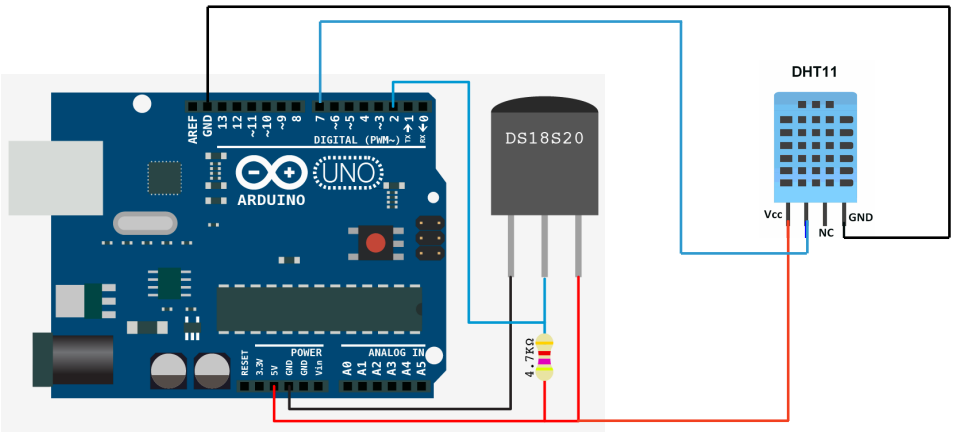


Figure 6.2: Overview of electrical connections between the Arduino Uno and used temperature sensors [14, 5]

6.1.2. Test Objectives and To-Be-Measured Quantities

In this section, the test objectives and to be measured quantities will be presented. The main goals of the experimental test are to measure the ambient temperature of the environment and the object's external temperature over time. Besides this, other measurements need to be conducted, such as measuring the wet and dry mass of the cups and their geometric values. To make sure all the objects of the experimental test campaign will be obtained, the required quantities which need to be measured are presented in Table 6.3 and the test objectives are presented in Table 6.4, to make sure they can be verified after the test campaign.

Table 6.3: To-be-measured quantities during the cooling down of objects experiments

ID		Measured Quantity	Method	Data Rate
Object Measurements Before Test (**OMBT**)				
OMBT-01		Outside diameter of White cup	Calliper	-
OMBT-02		Outside height of White cup	Calliper	-
OMBT-03		Dry mass of White cup	Scale	-
OMBT-04		Outside diameter of Long 18563 glass	Calliper	-
OMBT-05		Outside height of Long 18563 glass	Calliper	-
OMBT-06		Dry mass of Long 18563 glass	Scale	-
OMBT-07		Outside diameter of Medium 18562 glass	Calliper	-
OMBT-08		Outside height of Medium 18562 glass	Calliper	-
OMBT-09		Dry mass of Medium 18562 glass	Scale	-
OMBT-10		Outside Short 18560 glass	Calliper	-
OMBT-11		Outside height of Short 18560 glass	Calliper	-
OMBT-12		Dry mass of Short 18560 glass	Scale	-
Measurements During Test (**MDT**)				
MDT-01		Ambient temperature of test environment	DHT 11	1 [Hz]
MDT-02		Test object temperature from the side	DS18B20	1 [Hz]
Object Measurements After Test (**OMAT**)				
OMAT-01		Wet mass of White cup after the test	Scale	-
OMAT-02		Wet mass of Long 18563 glass glass after the test	Scale	-
OMAT-03		Wet mass of Medium 18562 glass after the test	Scale	-
OMAT-04		Wet mass of Short 18560 glass after the test	Scale	-

Table 6.4: Test objectives of the cooling down of objects experiments

ID		Objectives	Compliance Method
Primary test goal(s)			
PTG-01		Determine the temperature of the object over time	• Measure the temperature of the object on the outside of the object, where the temperature sensor is positioned in the middle of the object
PTG-02		Determine the average ambient temperature of the test environment during the test	• Measure the ambient temperature during the test • Average the measured temperatures after the test
PTG-03		Determine the wet mass of the test object	• Measure the object's mass after the test
Secondary test goal(s)			
STG-01		Document the test making use of a camera (movies and photos)	

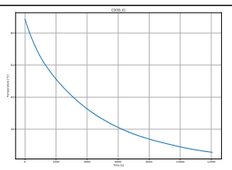
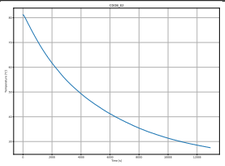
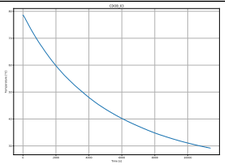
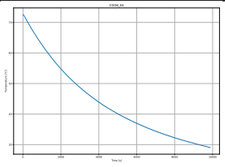
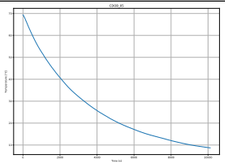
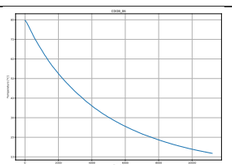
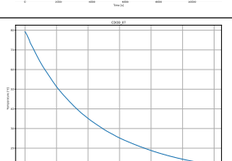
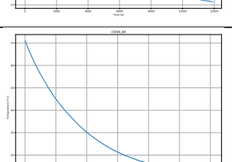
After all the tests were performed, it was concluded that all test objectives had been reached and no deviations had been found during the test campaign.

6.2. Cooling Down of Objects Experimental Data

After the experimental test campaign was conducted, the obtained experimental data needs to be processed and documented. Filtering of the experimental data files has been performed by making use of Savitzky-Golay filtering. To do so, the window length and polynomial order have been set in such a way that the mean and standard deviation of the filtered data are similar to the raw data files. The 8 filtered experimental data sets, including relevant information, are presented in Table 6.5. In this table, the data file name, description of the test, the window length and polynomial order for the Savitzky-Golay filter and the number of data points are presented. Furthermore, a plot providing the object's temperature over time is presented as well. Enlarged versions of the experimental data figures are presented in Appendix A, which can be accessed by pressing on the figure in the table [1].

[1] This is only possible in case this thesis is being viewed in a PDF viewer

Table 6.5: Cooling Down of Objects data overview

File name	Description		Window length	Polynomial order	Amount of Data Points	Object temperature figure
CDOB_01.csv	White clay cup with an average ambient temperature of 19.1 [°C] and a wet mass of 454 [g]	-	10	3	5555	
CDOB_02.csv	Long 18563 glass with an average ambient temperature of 19.8 [°C] and a wet mass of 552 [g]	-	10	3	6046	
CDOB_03.csv	Medium 18562 glass with an average ambient temperature of 20.7 [°C] and a wet mass of 402 [g]	-	10	3	5352	
CDOB_04.csv	Short 18650 glass with an average ambient temperature of 20.8 [°C] and a wet mass of 194 [g]	-	10	3	4622	
CDOB_05.csv	White clay cup with an average ambient temperature of 2.4 [°C] and a wet mass of 456 [g]	-	10	3	5430	
CDOB_06.csv	Long 18563 glass with an average ambient temperature of 5.6 [°C] and a wet mass of 520 [g]	-	10	3	5824	
CDOB_07.csv	Medium 18562 glass with an average ambient temperature of 4.3 [°C] and a wet mass of 376 [g]	-	10	3	6213	
CDOB_08.csv	Short 18560 glass with an average ambient temperature of 6.0 [°C] and a wet mass of 192 [g]	-	10	3	6694	

6.3. Data Partitioning Strategy Cooling Down of Objects

The first 3 sub-research questions presented in chapter 4 will be examined, making use of the LOO-CV method as explained in subsubsection 2.1.1.12. To recap, this means that 7 experimental data files will be used during the training of the models involving a data-driven component, while the single data set will not be used during the training phase and will be used as a test dataset to examine the performance of the model. To investigate the 4th research question related to surrogate modelling, the experimental data needs to be split up into various groups. The partitioning of the data into various groups should be done in such a way that, in the training data, certain input parameters from the test group are not present. For example, the experiments were performed making use of cups with 4 different lengths. To test the surrogate modelling capability of the model, during the training phase, the lengths present in the test data should be present in the training data of the model. It was decided to partition the experimental data files into 3 different groups. The partitioning of the groups is presented in the list below:

- Material
- Geometry
- Ambient temperature

It should be noted that certain experimental data sets belong to multiple groups. The various groups will be tested separately to be able to answer the research question related to surrogate modelling capability. The test data will be assessed such that all the test files have been used individually, and the average of these results will be obtained to be able to assess the performance of the model. For the first test, related to the difference in materials between the groups, the data will be split according to Table 6.6.

Table 6.6: Data partitioning strategy for the experimental data of the cooling down of objects experiments related to the various materials used

Clay	Glass
CDOB_01.csv	CDOB_02.csv
CDOB_05.csv	CDOB_03.csv
	CDOB_04.csv
	CDOB_06.csv
	CDOB_07.csv
	CDOB_08.csv

For the second test, which is related to geometry, the data will be split according to Table 6.7.

Table 6.7: Data partitioning strategy for the experimental data from the cooling down of objects experiments related to the various geometries used

Cup	Long	Medium	Short
CDOB_01.csv	CDOB_02.csv	CDOB_03.csv	CDOB_04.csv
CDOB_05.csv	CDOB_06.csv	CDOB_07.csv	CDOB_08.csv

For the last test, related to the ambient temperature, the data will be split according to Table 6.8

Table 6.8: Data partitioning strategy for the experimental data from the cooling down of objects experiments related to the various ambient temperature conditions

Room temperature	Outside temperature
CDOB_01.csv	CDOB_05.csv
CDOB_02.csv	CDOB_06.csv
CDOB_03.csv	CDOB_07.csv
CDOB_04.csv	CDOB_08.csv

During each test, the groups will be used at least once as test data sets. However, as more groups will be removed during the training process, multiple groups could be used as test data to assess the

performance of the model. This means that only a few data groups will be used as training data. This is a fully automated process, so no mistakes can be made during this process.

7

Numerical Model: Cooling Down of Objects

As previously discussed in subsection 2.1.2, a numerical model is required for a PINN to be able to train the network output derivatives. For the PINN, which will be generated for this part of the thesis, Newton's Law of Cooling will be used as underlying physical model for the numerical model. The developed numerical model will not only be used to train the PINN, but will also be used to be able to make a comparison between various modelling methods, such as the PINN. This chapter describes the development of the numerical model, making use of Newton's Law of Cooling as underlying physical model. First, the equations used, assumptions and limitations of the numerical model will be examined in section 7.1. Next, the setup of the numerical model will be explained in section 7.2. After this, the inputs of the model will be examined in section 7.3. Next, an example of the output predictions of the numerical model will be presented in section 7.4. Lastly, the verification process of the numerical model and the outcome will be presented in section 7.5.

7.1. Equations, Assumptions and Limitations of the Numerical Model

As described in chapter 5, the cooling down of an object due to convection as the main driving force can be described by making use of Newton's Law of Cooling. The differential equation which describes Newton's Law of Cooling has been derived in subsection 5.2.1 and, which resulted in Equation 5.2.

7.1.1. Equations Used in the Numerical Model

The main equation used for the to-be-developed numerical model is Newton's Law of Cooling as described in Equation 5.2. Transforming Equation 5.2 into a state equation, results in Equation 7.1.

$$\left[\dot{T}_{object} \right] = \left[\frac{hA}{mc_p} \left(T_{env} - T_{object} \right) \right] \tag{7.1}$$

7.1.2. Assumptions of the Numerical Model

When deriving Newton's Law of Cooling, several assumptions have been made to simplify the analysis. Most of the assumptions have already been stated in subsection 5.2.2. However, for the numerical model, there are some additional assumptions related to the input and outputs of the numerical model. The assumptions for the numerical model are presented in the list below [88]:

- Convection is the main heat transfer mechanism
- The object has a, single, uniform temperature
- There is a single form of convection, either forced or natural
- The mass of the object will remain constant
- The environmental temperature will remain constant

- The specific heat of the object will be independent of temperature

7.1.3. Limitations of the Numerical Model

From subsection 7.1.2, the assumptions made which are applicable to the numerical model were presented. It is important to note that Newton's Law of Cooling can only provide sufficiently accurate results in case the Biot number is below 0.1 [88]. Furthermore, it can only be used in case convection is the main form of heat transfer [88]. This should be taken into account when performing experiments or investigating the cooling down of objects. Furthermore, it should be noted that the numerical simulation has been tailored to only be used for cylindrical-shaped objects, meaning that providing other objects as input could result in unreliable results.

7.2. Setup of the Numerical Model

To provide an overview of how the model will provide predictions of the temperature over time making use of the state equation as presented in subsection 7.1.1, a flowchart has been generated, which is presented in Figure 7.1. The presented flowchart has later been implemented in a software code, making use of the Python coding language. The model makes use of a Runge-Kutta 5th-order integration scheme [104]. It should be noted that the output of the code is provided in degrees Celsius rather than Kelvin.

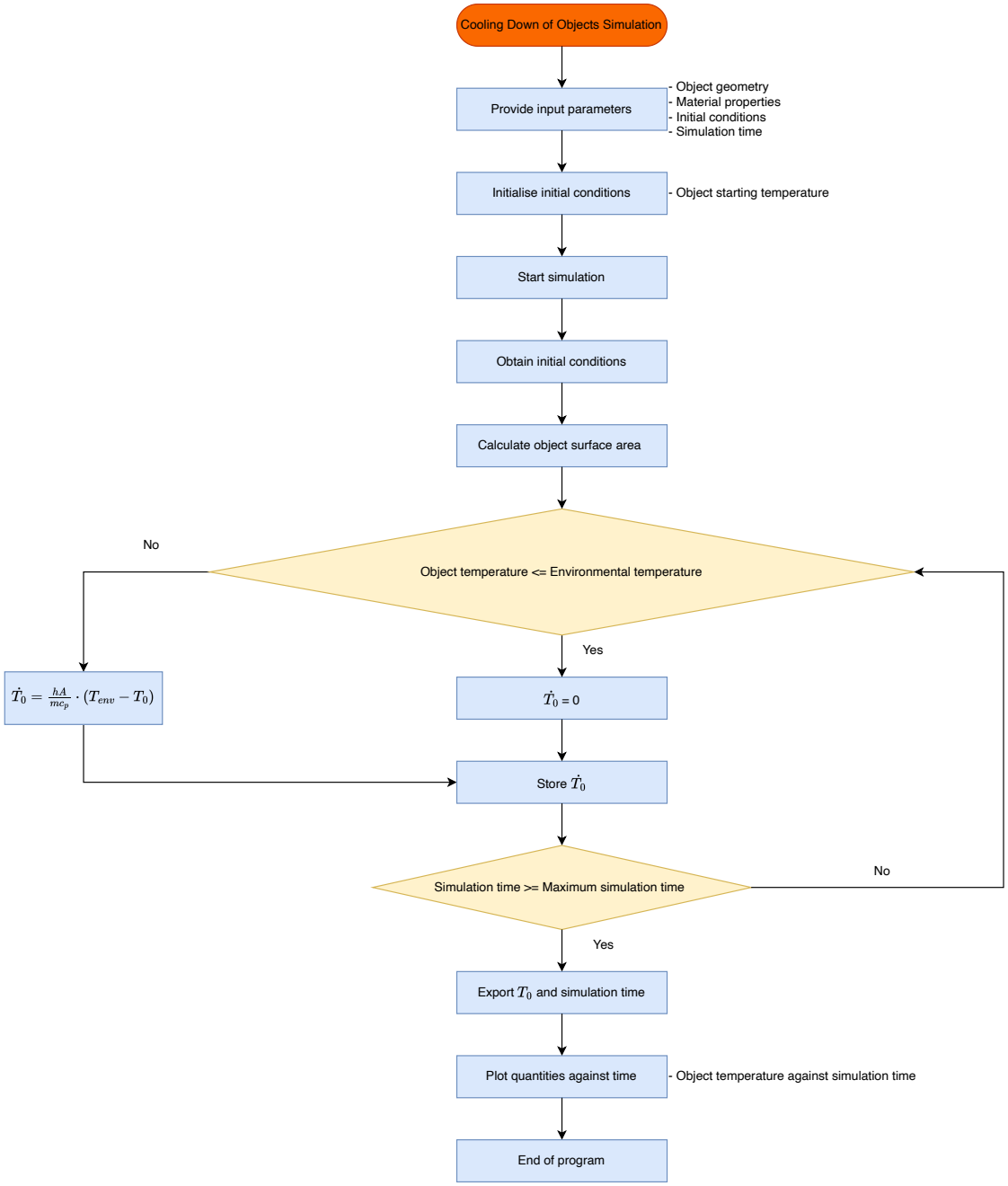


Figure 7.1: Flowchart of the numerical simulation model implemented in Python code for the cooling down of objects

7.3. Inputs of the Numerical Model

In order to obtain temperature predictions from the numerical simulation, inputs are required to specify the conditions of the simulation. For the inpiuts of every simulation, it is important to know the quantities and their required units. For the numerical simulation developed in this chapter, the inputs and their units are summarised in Table 7.1.

Table 7.1: Summary of inputs for the numerical simulation

Symbol	Meaning	Unit
Simulation Settings		
t	Simulation time	[s]
Constants		
h	Convective heat transfer coefficient	[W/m^{2}K]
Object Geometry		
L	Length of the object	[m]
r	Outer radius of the object	[m]
m	Wet mass of the object	[kg]
Material Properties		
c_p	Specific heat capacity of the object	[J/gK]
Environmental Conditions		
T_0	Starting temperature of the object	[°C]
T_{env}	Environmental temperature	[°C]

7.4. Results of the Numerical Model

As was previously discussed in section 7.1, the numerical model provides predictions of the temperature of the object over time, based on the provided inputs. An example of temperature predictions over time by the developed numerical simulation model is presented in Figure 7.2.

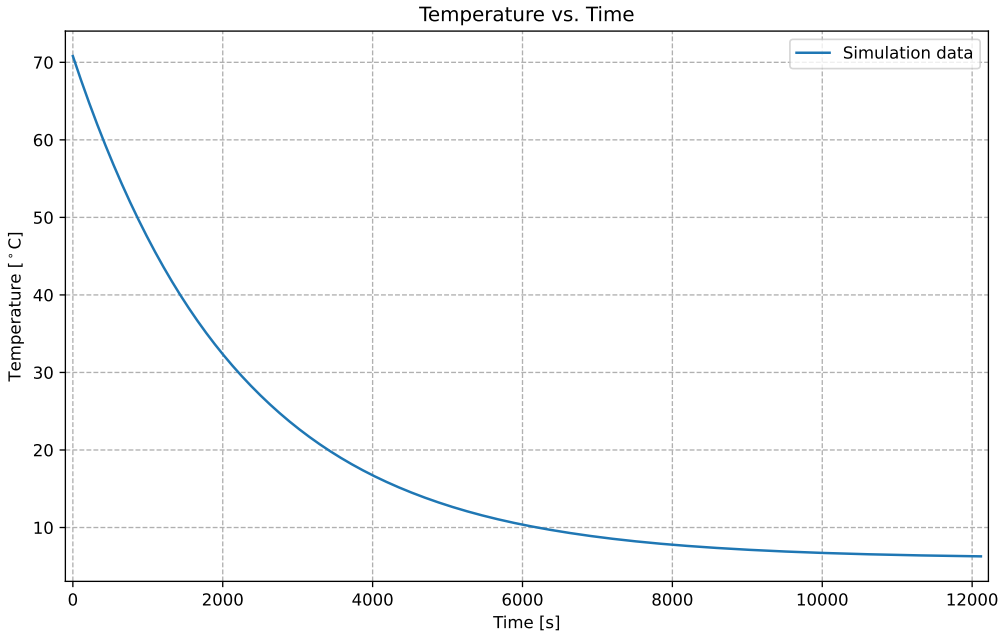


Figure 7.2: Example output predictions of the numerical model predicting the temperature over time making use of the developed numerical simulation model in Python code

7.5. Verification of the Numerical Model

In this chapter, a numerical simulation model has been developed to model Newton's Law of Cooling and providing temperature predictions over time, as presented in Equation 5.2. To estimate if the developed numerical model has been correctly implemented in the developed Python code, verification has to be performed, as was explained in subsubsection 2.1.1.15. The verification process for the developed numerical simulation model for the cooling down of objects was conducted as follows. First, after the design and implementation of the simulation for Newton's Law of Cooling in Python code was completed, the output of the simulation was obtained and inspected for flaws. After the expected results were obtained, the code was checked for the correct implementation of the used differential equation, as well as the units used as inputs. It should be noted that each time the inputs are updated, a verification process has to be performed to make sure that all the inputs have the correct dimensions.

After the process of inspecting the code for bugs was completed, it was concluded that the verification process was completed and there are currently no indications of implementation errors. To be able to compare the performance of the developed numerical simulation model, use was made of the analytical model to compare the results. To do so, an example input is provided to the simulation and compared against the analytical model, using the same inputs. The analytical model for Newton's Law of Cooling is presented in Equation 7.2 [88].

$$T(t) = T_{env} + (T_0 - T_{env})\, e^{-\left(\frac{hA}{mc_p}\right)t} \tag{7.2}$$

The results of the comparison between the numerical model and the analytical model are visually presented in Figure 7.3.

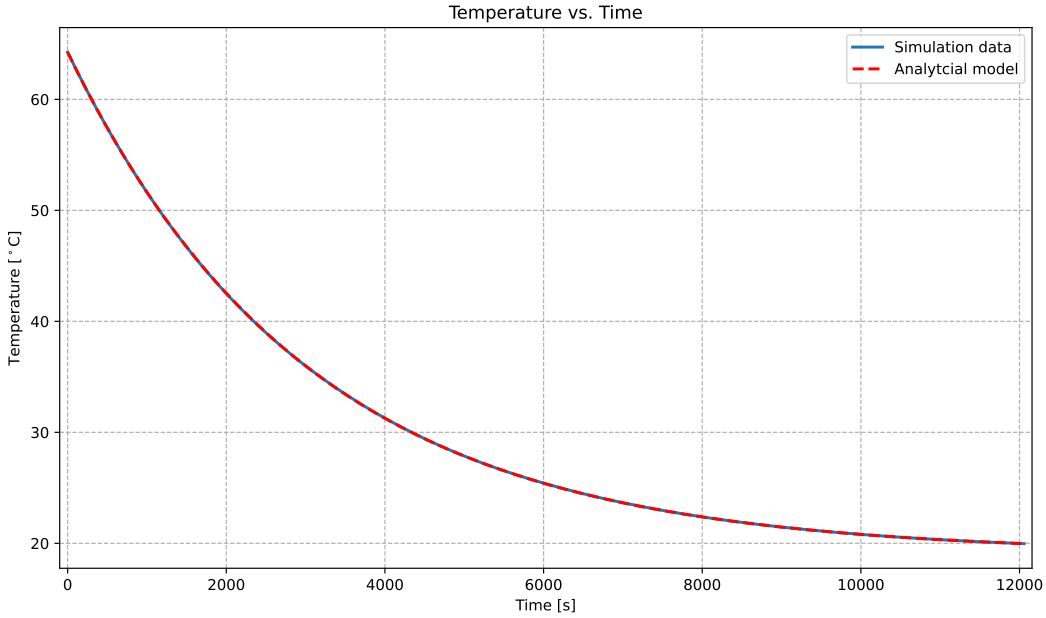


Figure 7.3: Visual overview of the verification output of the developed numerical simulation compared to the predictions of the analytical solution for the Cooling Down of Objects, making use of the same input conditions

From Figure 7.3, it can be seen that there is a good agreement between the temperature predictions of the numerical simulation compared to the analytical solution. Therefore, it was concluded that the numerical model is verified.

8

Physics-Informed Neural Network Model: Cooling Down of Objects

The main goal of this part of the research is to compare a PINN with a numerical and data-driven model in terms of computational efficiency, data efficiency, accuracy and surrogate modelling capability for the temperature predictions over time for the cooling down of objects. To do so, a PINN needs to be developed to be able to perform this comparison. This chapter will describe the development of the PINN, a description of the network architecture, as well as a strategy to develop the most optimal network architecture. Furthermore, an example output of the PINN is presented. In section 8.1, the network architecture of the PINN will be presented, as well as the implementation of the physics and boundary loss terms in Python. Furthermore, a flow chart of the PINN will be presented, as well as the inputs of the PINN. Lastly, an example output of the PINN model will be presented in section 8.2.

8.1. Setup of the Model

To generate a surrogate model of the PINN, as described in subsection 2.1.2, a conditioned PINN model should be developed, making use of inputs which generalise the modelling problem. To do so, it was decided to take some of the inputs of the numerical model developed in chapter 7 as the inputs for the PINN as well. The inputs of the PINN are summarised in Table 8.1.

Table 8.1: Summary of inputs used for the Cooling Down of Objects PINN model

Symbol	Meaning	Unit
Simulation Settings		
t	Simulation time	[s]
Object Geometry		
L	Length of the object	[m]
r	Outer radius of the object	[m]
m	Wet mass of the object	[kg]
Material Properties		
c_p	Specific heat capacity of the object	[J/gK]
Environmental Conditions		
T_0	Starting temperature of the object	[°C]
T_{env}	Environmental temperature	[°C]

Comparing the inputs for the PINN to the inputs of the numerical model from section 7.3, it can be seen that there is only 1 different input. For the PINN, it is assumed that the convective heat transfer coefficient will be constant for all problems and will therefore not be used as input for the surrogate model. So, to conclude, to make a surrogate PINN for this specific problem related to heat transfer

for the cooling down of objects and making predictions, 7 inputs are required. Since only the object's temperature over time will be predicted, only 1 output is required. This means that a total of 7 input neurons and 1 output neuron is required. In this research, it was found that the PINN needed 4 hidden layers containing 64 neurons per layer to provide sufficient results. The ReLU activation function is used for all the artificial neurons in the network. The initialisation method of the PINN model is set to uniform Xavier initialisation for its weights, and the biases are initialised to zero. An illustration of the final PINN architecture is presented in Figure 8.1.

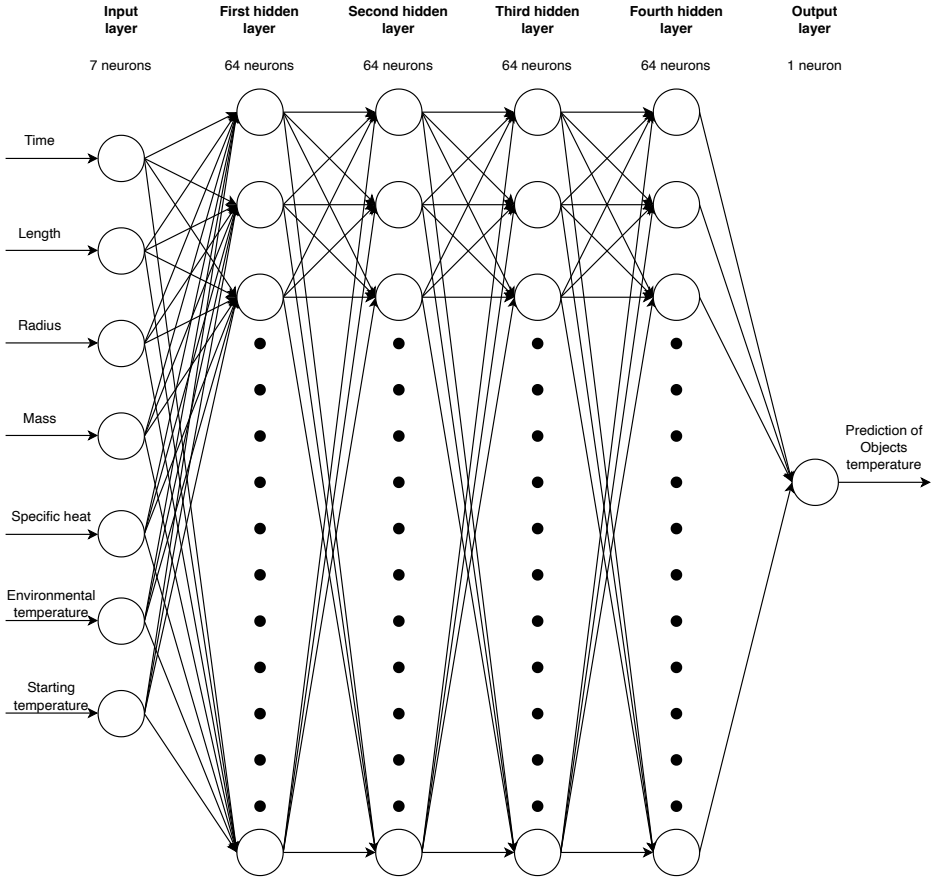


Figure 8.1: Neural Network architecture for the Physics-Informed Neural Network related to the modelling of the cooling down of objects

As explained in chapter 3, a specific physics and boundary condition need to be implemented. For this problem, the implementation of the physics loss function is presented in Code 8.1 and the boundary loss is presented in Code 8.2. In the physics loss, Newton's Law of Cooling is being implemented, making use of 500 collocation points in the computational domain between 0 and 15000 [s]. The weight in the dictionary of this loss function is set to 1. The boundary condition, which is being implemented, is a starting condition, which in this case is forcing the simulation to start at the starting temperature of the object, T_0. This makes use of a single collocation point at time $t = 0$ [s]. Similar to the physics loss, the boundary loss function also has a weight of 1.

Code 8.1: Implementation of the physics loss of Newton's Law of Cooling

```
1
2 def Newtons_cooling_law(NN_output: torch.nn.Module, used_inputs):
3     # Setup collocation points over domain
4     start, end = 0, 15e3
5     steps = 500
```

```
    time_inputs = torch.linspace(start+1000, end, int((steps/10) * 5)).view(-1, 1).
    requires_grad_(True)
    log_space1 = torch.logspace(start=torch.log10(torch.tensor(1e-16)),end=torch.log10(torch.
    tensor(start+1000)),steps=int((steps/10) * 5)).view(-1,1).requires_grad_(True)

    # Concatenate with the reversed version to get more density at both start and end
    Collocation_temps = torch.cat((log_space1, time_inputs), dim=0)

    # Reset total loss
    total_loss = 0

    for inputs in used_inputs:

        L = inputs[0]
        R = inputs[1]
        m = inputs[2]
        c_p = inputs[3]
        T_env = inputs[4]

        # Calculate area for formula
        A = 2 * np.pi * R * L

        # Repeat the constants for each time step (number of time points)
        constant_features_repeated = inputs.unsqueeze(0).repeat(Collocation_temps.size(0), 1)

        # Concatenate the constant features with the time feature along the second axis (
    columns)
        input_data = torch.cat((Collocation_temps, constant_features_repeated), dim=1)

        # Obtain predictions from the network (outputs)
        T_predicted = NN_output(input_data)

        # Obtain dT/dt_scaled
        dT_dt = gradient(T_predicted, input_data)[:,0].unsqueeze(-1) #* (1 / (1 +
    Collocation_temps))


        # Newton's Law of Cooling
        NLC = (((h * A) / (m * c_p * 1e3)) * (T_env - T_predicted)) - dT_dt

        # Determine loss for specific case and add to total loss
        total_loss += torch.mean((NLC)**2)

    return total_loss / len(used_inputs)
```

Code 8.2: Implementation of the boundary loss of Newton's Law of Cooling

```

def BC_start(NN_output: torch.nn.Module, used_inputs):
    # Setup input network (t = 0)
    time_start = torch.tensor(0.).view(-1,1).requires_grad_(True)

    # Reset total loss
    total_loss = 0

    for inputs in used_inputs:

        # Obtain starting temperature
        T_0 = inputs[5]

        # Repeat the constants for each time step (number of time points)
        constant_features_repeated = inputs.unsqueeze(0).repeat(time_start.size(0), 1)

        # Concatenate the constant features with the time feature along the second axis (
    columns)
        input_data = torch.cat((time_start, constant_features_repeated), dim=1)

        # Obtain input temperature prediction from network
        Temp_start = NN_output(input_data)

        # Determine loss for specific case and add to total loss
```

```
        total_loss += torch.mean((Temp_start - T_0)**2)

    return total_loss / len(used_inputs)
```

To be able to encode the PINN in Python, making use of the developed code class as presented in chapter 3, a flowchart was developed to provide a general overview of the coding structure. The flowchart for the developed PINN for the cooling down of objects is presented in Figure 8.2.

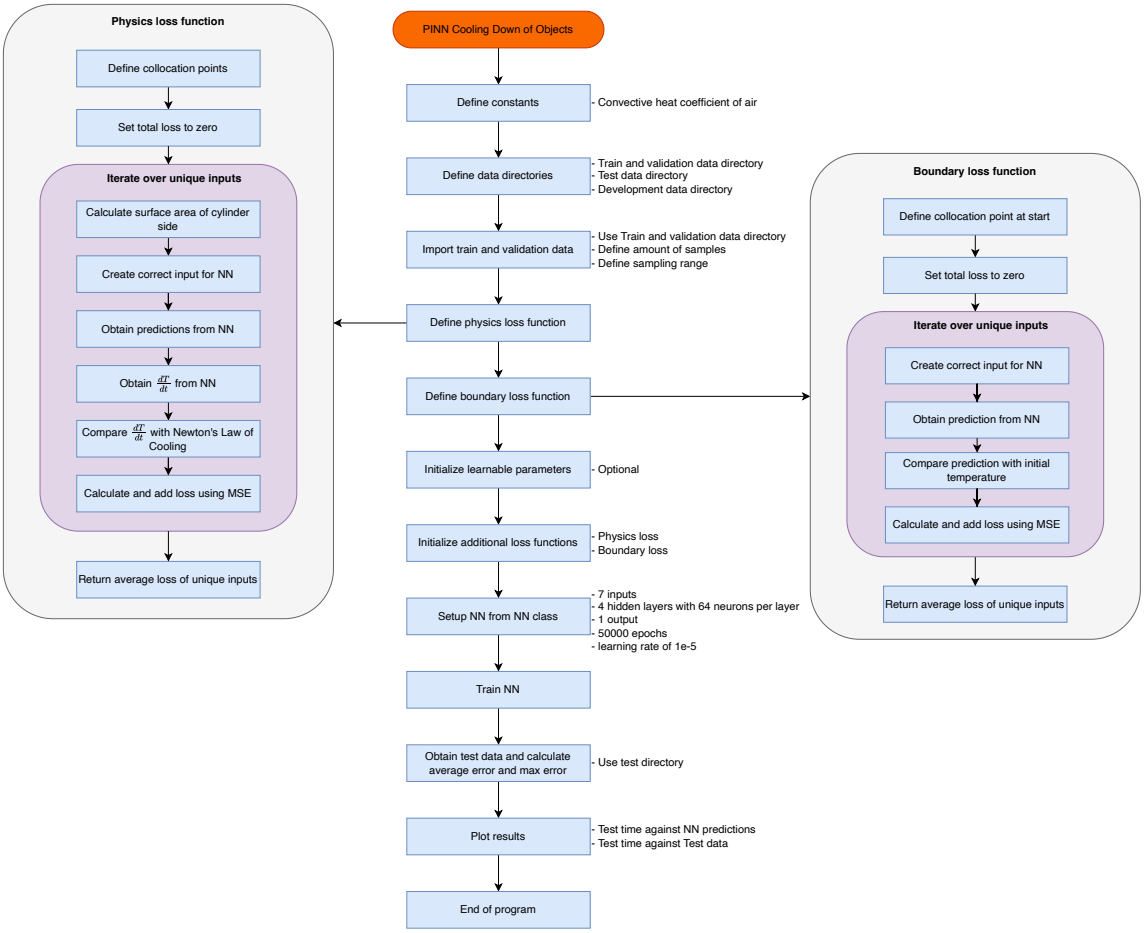


Figure 8.2: Flowchart of the Python code for the PINN program developed for the cooling down of objects

8.2. Results of the PINN Model

As was previously explained, the PINN generates a prediction of the temperature of an object over time. An example of a prediction is presented in Figure 8.3, together with the training data, test data and numerical model predictions.

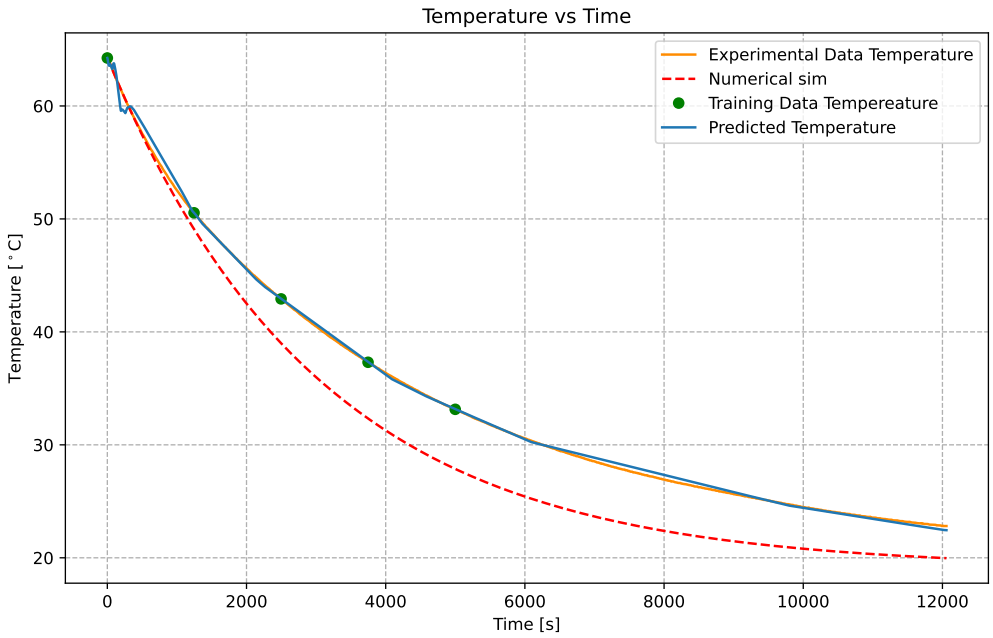


Figure 8.3: Example output of the PINN developed for the cooling down of objects, compared to its training data and the real experimental data

From Figure 8.3, it can be seen that only a few data points are provided, presented as the green dots in Figure 8.3, in the range of 0 to 5000 [s]. However, the PINN model makes predictions from 0 to approximately 12000 [s]. What can be seen is that the PINN model, represented as the blue line, has, besides some inconsistencies from 0 to 100 [s], a good agreement with the measured experimental data which is presented as the orange line, given the few amount of data points it has used during the training phase. Comparing the orange line to the numerical model, which was developed in chapter 7 and presented as the red dotted line in Figure 8.3, it can be seen that the experimental data has a slower decrease in temperature compared to the predictions of the numerical model. However, it should be noted that the output of the PINN model is dependent on the number of data points used, as well as the positioning of the data points within the domain. This also holds for the weight of the additional losses and the amount and position of collocation points. The inconsistencies of the PINN model, ranging from 0 to 100 [s], could be caused due to a wide variety of reasons. First of all, since there are not many data points, the data-driven part of the model will quickly be able to minimise its loss, which could lead to nonphysical behaviours of the model since the parameters of the NN will no longer be adapted. Besides this, the enforcement of the physics loss term and boundary loss term could lead to competing terms, meaning that the model will, at some point, be stuck optimising the various loss terms. Besides this, the learning rate and number of epochs also greatly influence the overall performance of the model. The presented example in Figure 8.3 has a total of 3500 epochs and a learning rate of $1 \cdot 10^{-4}$. Increasing or decreasing the number of epochs or the learning rate could lead to an unstable model and increase computational time. Therefore, it is important to balance accuracy and computational efficiency during the training process, since one will greatly influence the other.

9

Model Comparison: Cooling Down of Objects

After the numerical and PINN models have been developed, it is time to perform tests to compare them. To answer all research questions, a data-driven method was also developed, making use of the same network configuration as the PINN, as was explained in chapter 8, but without the physics and boundary loss functions in the cost function. This chapter will explain the performed comparison tests and examine the results. To be able to answer the first 3 research questions, the LOO-CV strategy will be applied to the experimental data sets obtained in section 6.2. Firstly, the models used in the comparison will be explained in section 9.1. After this, the general process and the results of the LOO-CV process will be explained in section 9.2. After this, the surrogate modelling capability of the various modelling methods will be examined, which will be explained, and the results will be presented in section 9.3.

9.1. Comparison Model

In this section, the models which will be compared will be explained. During this chapter, the developed PINN as described in chapter 8 will be compared against the Numerical model developed in chapter 7. Furthermore, these models will be compared against a purely Data-Driven model with the same network architecture as the PINN. Besides this, an I-PINN will be developed as well by adding the convective heat transfer coefficient as a network parameter to the PINN network, adapting this parameter to the training data. An initial value of 5 [W/m^{2}K] will be used as an initial guess for the convective heat transfer coefficient parameter. However, when the convective heat transfer coefficient is learned by the network, this value should also be adapted in the numerical simulation to provide a fair evaluation. Therefore, the last model which will be used during this comparison is the I-Numerical model, which has the same setup as the Numerical model, except for the convective heat transfer coefficient, which is provided from the I-PINN. All the models making use of a data-driven component will have the same network and learning rate, which will be set to $1 \cdot 10^{-5}$. The PINN and I-PINN will have 500 collocation points in total, in a computational domain between 0 and 15000 [s].

9.2. Leave-One-Out Cross Validation Comparison

For data-driven methods, including PINNs, a validation strategy should be chosen which will lead to a fair estimate in performance, as explained in subsubsection 2.1.1.15. For the Cooling Down of Objects, it was decided to make use of the LOO-CV method to be able to answer the first 3 research questions. To do so, the 8 experimental data sets will be split into 2 different groups. The first 7 experimental data tests were used to train the PINN, I-PINN and the purely Data-Driven method. For the training process, a certain number of data points were used, sampled over the domain, but similar to each method, meaning that for each validation run, the same training data was available for the PINN, I-PINN and Data-Driven model. The remaining test, which was not been part of the training data, will

be used to estimate the performance of the PINN, I-PINN, Data-Driven method, Numerical method and I-Numerical method. To do so, the experimental data of the remaining test in the test group was provided as input to the various methods, whereafter the performance was determined by taking the RMSE of the developed method and the experimental data. Besides this, the maximum error, as well as the training time and execution time, were also recorded, if applicable, for the selected method. This process was then repeated until all 8 experimental data files had served once as test data files. Lastly, the average of the RMSE and the average of the maximum error, training time and execution time were taken as final estimates for the individual model performance.

As previously explained, the RMSE and maximum error were taken as performance metrics for these experiments. The reasoning behind this is that the RMSE will give a fair performance metric regarding how well the model will work in general. However, if a small inconsistency between the model and the test data set is detected at a certain position, but the other prediction errors are rather small, the RMSE will average out, showing a relatively small error, while the shape of the prediction is not always followed directly. Therefore, to provide a metric of how well the shape is predicted, the maximum error is used. If both the RMSE and the maximum error are small, the model has a good prediction capability.

To be able to answer the research questions, the following experiments are being conducted. First, the modelling methods will be trained using 50 data points, whereafter the training time, execution time, average RMSE and average max error are assessed. Next, the data efficiency will be determined. This will be done by obtaining the number of data points to reach a certain accuracy. To obtain this accuracy threshold, use will be made of the maximum accuracy of the various sensors used during the experimental measurements, as was presented in chapter 6. From the datasheets in Appendix B and Appendix C, it can be found that the error margins of the sensor are ± 2 [°C] and ± 0.5 [°C], respectively. This means that the measured ambient temperature could differ by ± 2 [°C], and the object temperature could differ by ± 0.5 [°C]. The total expected error due to sensor accuracy can be calculated by making use of Equation 9.1 [52].

$$E_{total} = \sqrt{\sum E_{sensor}^2} \tag{9.1}$$

This results in a provided error of 2.06 [°C]. This number is chosen as a threshold for the data efficiency metric and to assess the accuracy. For each method, the accuracy for the specific number of data points needed to require this threshold is also presented. This accuracy threshold means that all performances of the model below this threshold in terms of maximum error and RMSE could be caused due to the deviation of the used sensors, meaning that below this threshold, the modelling methods are no longer distinguishable in terms of performance.

9.2.1. Computational Efficiency

To examine the computational efficiency, the performed test data of the LOO-CV process has been summarised in Figure 9.1 for the training data and Figure 9.3 for the execution time.

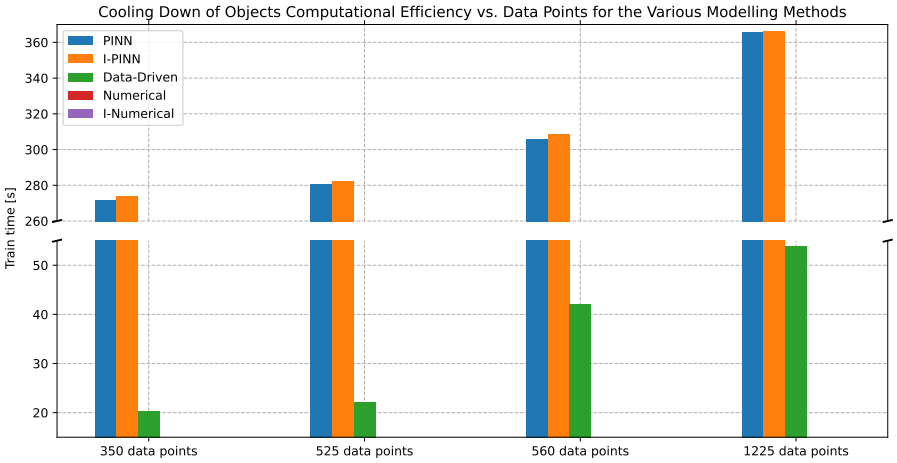


Figure 9.1: Bar chart presenting the train time against the number of data points for the various modelling methods

From Figure 9.1, it can be seen that the PINN and I-PINN have the largest training time for all the methods involving a data-driven aspect, ranging from 270 to 366 [s] for the PINN and 275-367 [s] for the I-PINN. Furthermore, it can be seen that the I-PINN has the largest training time. The most likely explanation for this observation is that the I-PINN has additional network parameters which also need to be varied to be able to find the most optimal solution. This changing of numbers takes time; therefore, introducing a slightly larger training time compared to the PINN. Looking at the differences between the Data-Driven method, PINN and I-PINN, it can be seen that the Data-Driven method has the smallest training time, ranging from 20-54 [s]. This observation can most likely be explained because the PINN and I-PINN need to perform additional calculations in the form of obtaining the derivative of the network at certain collocation points, calculating the theoretical derivative and comparing the two. This, most likely, introduces an increase in training time. When increasing the number of data points, it can be seen that the training time of all modelling methods involving a data-driven component increases. This is expected, as by introducing additional training data, the training time will increase since more values need to be compared to each other. To conclude, the PINN and I-PINN have the largest training time of the methods involving a data-driven component. The purely Data-Driven method has the smallest training time, while the I-PINN has the largest. Now, to determine the training time required to obtain the threshold set by the threshold, Figure 9.2 has been created.

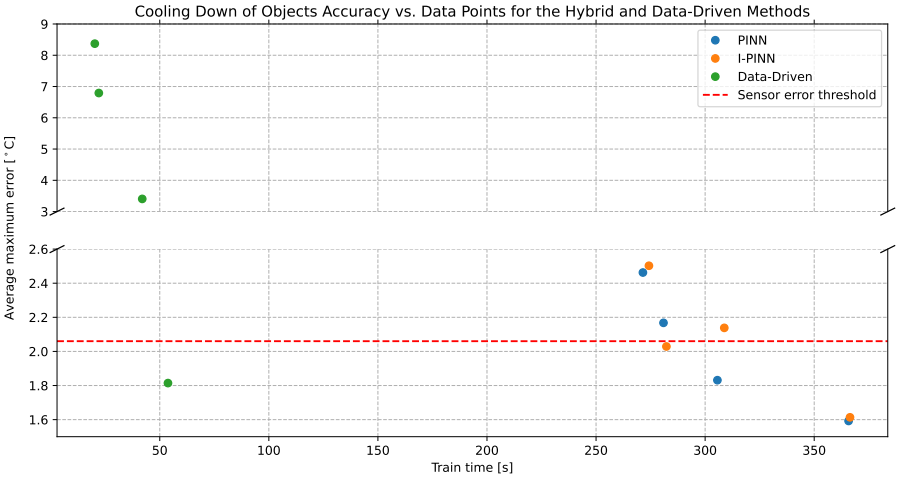


Figure 9.2: Graph presenting the average maximum error against the training time, including the maximum accuracy threshold obtained from the total sensor error

From Figure 9.2, it can be seen that a training time of approximately 305.6 [s] is required for the PINN and 53.8 [s] for the Data-Driven method to reach the threshold dictated by the sensors. However, about 282.3 [s] are required for the I-PINN. This means that the I-PINN has a slightly lower computational time compared to the PINN. Overall, the PINN and I-PINN have a training time which is about 5-6 times larger compared to a purely Data-Driven method to reach the same accuracy level.

However, since the training phase in principle only happens once, comparing the execution times of all the various modelling methods would be a better comparison to determine the computational efficiency. In Figure 9.3, the execution times for the various modelling methods are presented.

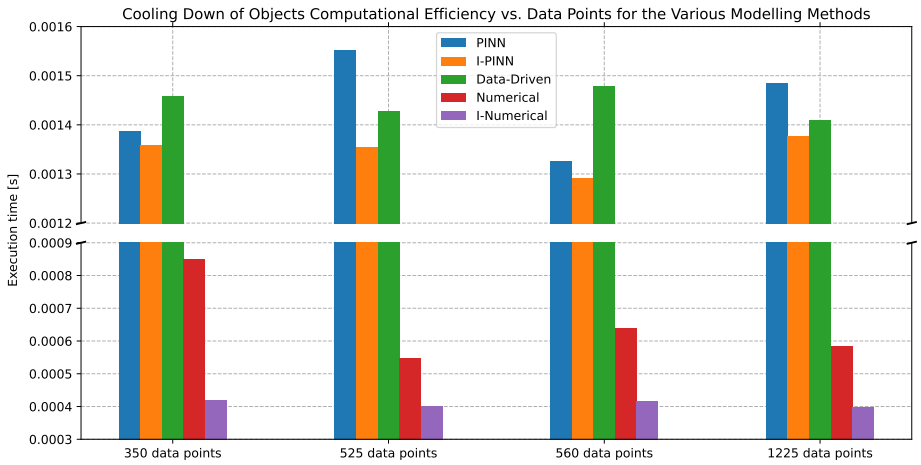


Figure 9.3: Bar chart presenting the execution time against the number of data points for the various modelling methods

From Figure 9.3, it can be seen that all methods involving a data-driven component have an only slightly higher execution time compared to the Numerical methods. Furthermore, it can be seen that the recorded execution times for the Numerical method are higher compared to the I-Numerical methods. This observation is most likely due to the variance in computational time due to workload and other tasks performed on the computer, since the execution time for the Numerical model should be independent of the number of data points. However, since this is not the case, only the order of magnitude can

be compared. Comparing the methods involving a data-driven component, the methods are relatively closely related to each other, ranging from 0.0013 to 0.0016 [s]. This observation is most likely caused because the network architecture of the methods is the same, and once trained, there is no difference between the PINN, I-PINN and Data-Driven method other than the weights and biases of the network. To conclude, the Numerical methods have only a slightly lower execution time compared to the methods involving a data-driven component. From this observation, it can be concluded that the PINN and I-PINN do not have a higher computational efficiency compared to a Numerical or purely Data-Driven method.

9.2.2. Data Efficiency

To examine the data efficiency of the various methods, use can be made of Figure 9.4, which shows the average maximum error of the various modelling methods against the number of data points used during the training process. It should be noted that the data efficiency is only applicable for the methods involving a data-driven component, since the Numerical and I-Numerical methods do not have data directly involved in their computational process.

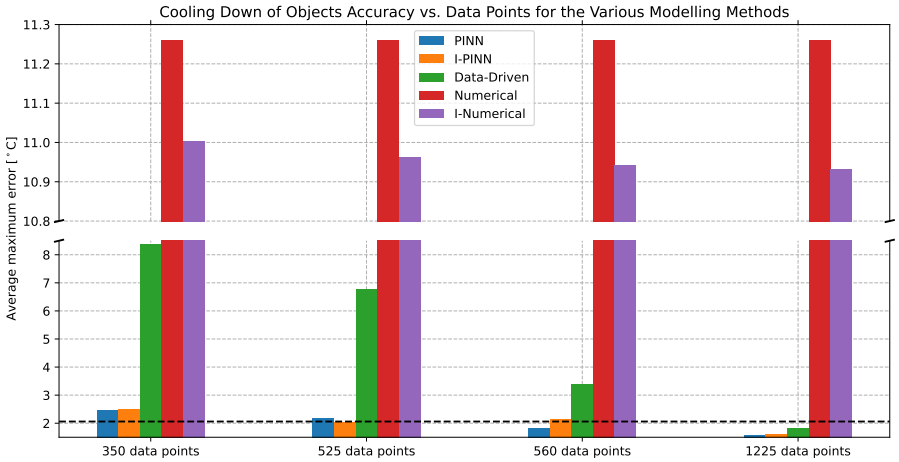


Figure 9.4: Bar chart presenting the average maximum error against the number of data points for the various modelling methods

From Figure 9.4, it can be seen that the average maximum error of the Numerical model is the largest, about 11.26 [°C]. After this, the I-Numerical method follows, with an average maximum error ranging between 10.93 and 11.0 [°C]. Now, although it was previously explained that the I-Numerical method does not have direct data-points involved, the I-PINN will provide the learned parameter based on the data, in this case, the convective heat coefficient, to the model, which in this case results in a lower error compared to the numerical model. However, none of the numerical models can reach the threshold of 2.06 [°C]. The methods involving a data-driven component, however, can reach this threshold. The I-PINN reaches below the threshold of 2.06 [°C] using 525 data points, with an average maximum error of about 2.03 [°C], while the PINN only needs 35 data points more, 560 data points to reach the threshold, with an average maximum error of about 1.83 [°C]. The purely Data-Driven method, although decreasing rapidly using more data points, is only able to reach below the threshold when making use of 1225 data points, having an average maximum error of about 1.81 [°C]. This means that, by making use of a PINN and I-PINN, a data reduction of 2-3 times can be obtained compared to a purely Data-Driven method, reaching the same order of accuracy.

9.2.3. Accuracy

Lastly, the accuracy of the various methods will be examined before the surrogate-ness of the various methods is discussed. The average RMSE of the various methods is presented in Figure 9.5 against the number of data points.

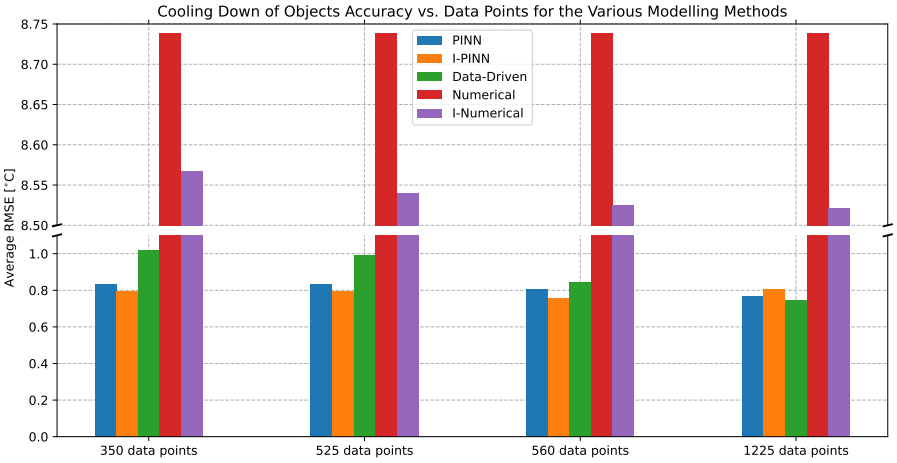


Figure 9.5: Bar chart presenting the average RMSE against data points for the various modelling methods

From Figure 9.5, it can be seen that the average RMSE is the highest for the Numerical method, with a value of about 8.74 [°C]. The I-Numerical model is slightly lower, with an average RMSE ranging between 8.52 to 8.57 [°C]. All the methods, including a data-driven component, have an average RMSE of below 1 [°C] after using 525 data points, while the PINN and I-PINN only need 350 data points to reach this type of accuracy. The PINN has an average RMSE ranging from 0.77 to 0.83 [°C], and the I-PINN has an average RMSE ranging from 0.76 to 0.81 [°C]. The purely Data-Driven method on the other hand, has an average RMSE ranging between 0.74 to 1.02 [°C] However, as could be seen from Figure 9.4, while the maximum error for the PINN and I-PINN are 2.46 and 2.5 [°C] for 350 data points, respectively, the maximum error for the purely Data-Driven method is about 8.37 [°C] using the same amount of data points. This means that the average RMSE alone is not a good metric to assess the overall performance of the model in terms of accuracy. However, including the maximum error, it can be concluded that the PINN and I-PINN have the highest accuracies of all the methods, having an average RMSE below 1 [°C] and maximum error below the determined threshold of 2.06 [°C] after making use of 560 data points during the training phase. The purely Data-Driven method is only able to reach this type of accuracy using 1225 data points during the training phase, while both the Numerical and I-Numerical methods are not able to reach this order of accuracy.

When performing statistical analysis, it was concluded that all the methods which had a data-driven component performed better compared to the numerical methods. However, it was found that the differences in errors between the methods which involve a data-driven component are not statistically significant, meaning that it can not be said that one method is better than the other in terms of average RMSE. For average maximum error, on the other hand, the differences are statistically significant between both the PINN and I-PINN compared to the fully Data-Driven method. However, the differences between the PINN and I-PINN are not statistically significant, meaning that both methods could be better models. In conclusion, the PINN and I-PINN do statistically prove to perform better in terms of the average maximum error between the fully Data-Driven method and numerical methods. For the average RMSE, this conclusion is not supported by statistics, but the models incorporating a data-driven component do perform better compared to the numerical models, a result which is found to be statistically significant.

9.2.4. Data Table LOO-CV

The detailed numbers presented in the previous subsections for the LOO-CV tests are provided in Table 9.1, together with all the other measured parameters during the tests for the LOO-CV process.

Table 9.1: Comparison of Cooling Down of Objects: Computational Efficiency, Data Efficiency, and Accuracy across the various used methods

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Computational Efficiency					
Train Time (Avg. Max Error of 2.06< [°C]) [s]	305.626051	282.251095	53.758441	x	x
Train Time (using 350 data points) [s]	271.478126	274.237702	20.219290	x	x
Train Time (using 525 data points) [s]	80.913093	282.251095	22.0621702	x	x
Train Time (using 560 data points) [s]	305.626051	308.781220	41.965435	x	x
Train Time (using 1225 data points) [s]	365.766499	366.408989	53.758441	x	x
Execution Time (Avg. Max Error of 2.062< [°C]) [s]	0.001326	0.001354	0.001409	x	x
Execution Time (350 data points) [s]	0.001388	0.001359	0.001458	0.000850	0.000420
Execution Time (525 data points) [s]	0.001552	0.001354	0.001428	0.000549	0.000403
Execution Time (560 data points) [s]	0.001326	0.001291	0.001479	0.000640	0.000418
Execution Time (1225 data points) [s]	0.001485	0.001376	0.001409	0.000585	0.000398
Data Efficiency					
Data Points for Avg. Max Error of 2.062< [°C]	560	525	1225	x	x
Accuracy					
Avg. RMSE (using 350 data points) [°C]	0.832647	0.796478	1.019728	8.738852	8.566384
Avg. Max Error (using 350 data points) [°C]	2.462597	2.502383	8.370630	11.259344	11.003159
Avg. RMSE (using 525 data points) [°C]	0.832672	0.797339	0.990161	8.738852	8.539941
Avg. Max Error (using 525 data points) [°C]	2.167577	2.028928	6.791827	11.259344	10.961561
Avg. RMSE (using 560 data points) [°C]	0.804193	0.758122	0.842010	8.738852	8.525232
Avg. Max Error (using 560 data points) [°C]	1.831345	2.138103	3.405330	11.259344	10.941790
Avg. RMSE (using 1225 data points) [°C]	0.766988	0.807348	0.744099	8.738852	8.521178
Avg. Max Error (using 1225 data points) [°C]	1.592232	1.613586	1.814330	11.259344	10.932451

9.3. Surrogate Modelling Comparison

From section 9.2, it could be concluded that PINNs and I-PINNs show promising results related to increasing accuracy and data-efficiency compared to a Numerical and a purely Data-Driven method. However, the Numerical method is still superior in terms of computational efficiency. After the first comparison test was conducted, the next test examining the surrogate-ness of the various modelling methods will be performed, as explained in section 6.3. To do so, each method containing a data-driven component will be trained on a certain data group, whereafter the performance will be assessed over the groups which are not being used, and the average RMSE and maximum error will be computed. After this, more groups will be removed from the training data, and each time, the performance metrics will be assessed. This way, it can be concluded how and if the model can be made surrogate and what their limitations would be.

During the training phase, 40 data points per data file will be used. Each method will be compared against a baseline LOO-CV, which has been performed with 280 data points in total. The training process has the same network parameters and loss functions as described in section 9.2.

9.3.1. Material Group

Firstly, the material groups will be varied to investigate the surrogate modelling possibility of the various models. To do so, the average RMSE and maximum error are presented against the baseline LOO-CV. In Figure 9.6, the results for the average RMSE are presented, and in Figure 9.7, the results for the maximum error are presented.

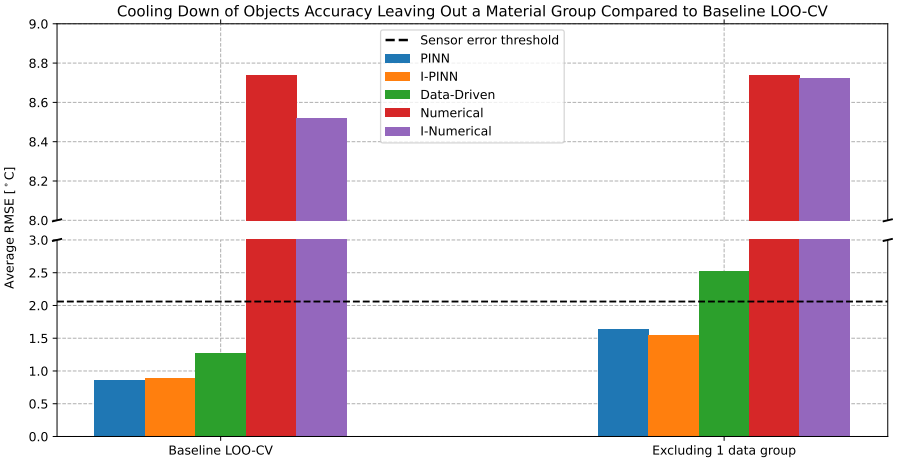


Figure 9.6: Bar chart presenting the average RMSE against various data partitioning strategies for the materials group for the various modelling methods

From Figure 9.6, it can be seen that leaving out 1 of the two material groups during training doubles the average RMSE of the methods containing a data-driven component. Still, the PINN and I-PINN have an average RMSE below the threshold of 2.06 [°C], meaning that these models could still be considered accurate in terms of average RMSE. The I-Numerical method has only a slight increase in average RMSE. However, to be able to draw a proper conclusion, the maximum error should also be investigated, which is presented in Figure 9.7.

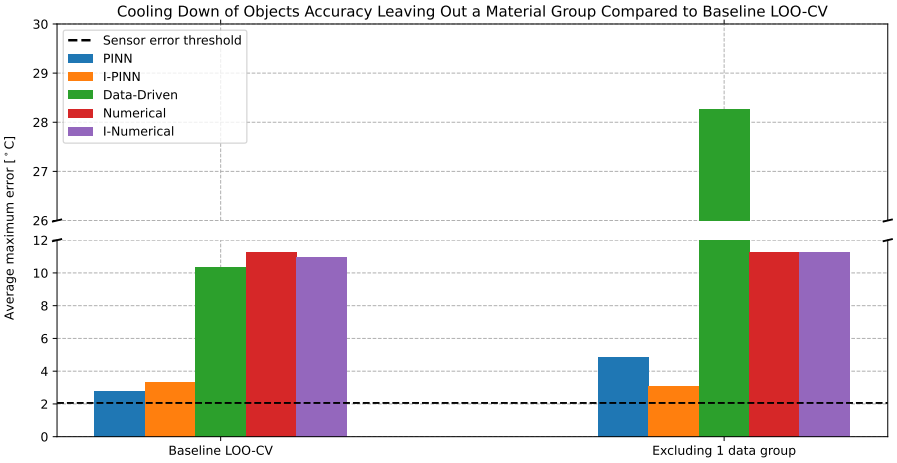


Figure 9.7: Bar chart presenting the average maximum error against various data partitioning strategies for the materials group for the various modelling methods

Examining Figure 9.7, it can be seen that the differences between the baseline LOO-CV and the training without 1 data group are very similar for the I-PINN method. For the PINN, the error doubles, which is similar to the purely Data-Driven method. However, where the purely Data-Driven method behaves similarly to the Numerical methods for the baseline, excluding a data group during training now results in the fact that the purely Data-Driven method is the most inaccurate compared to the other methods in terms of average maximum error. The PINN and I-PINN are the most accurate modelling methods, even when leaving out a data materials group during the training phase, the I-PINN is still close to the baseline.

To conclude, taking into account both the average RMSE and the average maximum error, the I-PINN show the most satisfactory behaviour for surrogate modelling compared to the standard baseline LOO-CV, with a maximum error comparable to the results of the LOO-CV for 280 data points, as can be seen in Figure 9.4. The PINN method, although performing better than the Numerical methods, does not provide a sufficient surrogate modelling capability since the maximum error is about twice the set threshold. However, comparing the PINN and I-PINN to the Numerical and purely Data-Driven methods shows that these methods are better at surrogate modelling in terms of accuracy.

9.3.2. Geometry Group

After the material groups, the geometry groups will be varied to investigate the surrogate modelling capabilities of the various modelling methods. To do so, the average RMSE and average maximum error will again be used to assess the performance. In total, the datasets can be split up into 4 different geometry groups, as was presented in section 6.3, meaning that 1 to 3 groups can be excluded from the training process and used to assess the performance. In Figure 9.8, the average RMSE of the baseline LOO-CV and the 3 excluding possibilities are presented for all the various modelling methods, while in Figure 9.9, the same analysis is performed for the average maximum error.

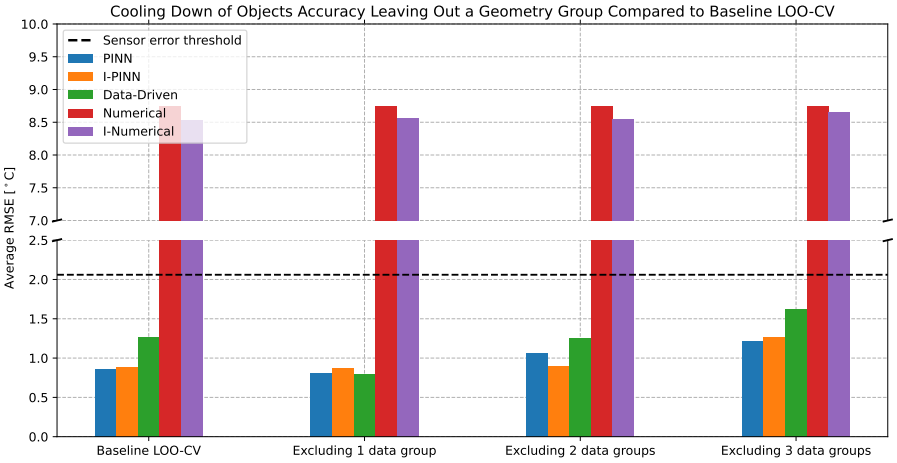


Figure 9.8: Bar chart presenting the average RMSE against various data partitioning strategies for the geometry groups for the various modelling methods

From Figure 9.8, it can be seen that compared to the baseline LOO-CV, the exclusion of 1 geometry group will have similar results compared to the baseline, even a slight increase in performance for the purely data-driven method. However, since the observations are under the provided threshold, all the modelling methods could have performed equally well. This also holds for the exclusion of 2 and 3 geometry groups, although it can be seen that the average RMSE increases for the modelling methods including a data-driven component, almost doubling the average RMSE for the PINN and I-PINN when excluding 3 geometry groups during the training process. However, a proper conclusion can only be drawn once the methods are compared, including the average maximum error, which is presented in Figure 9.9.

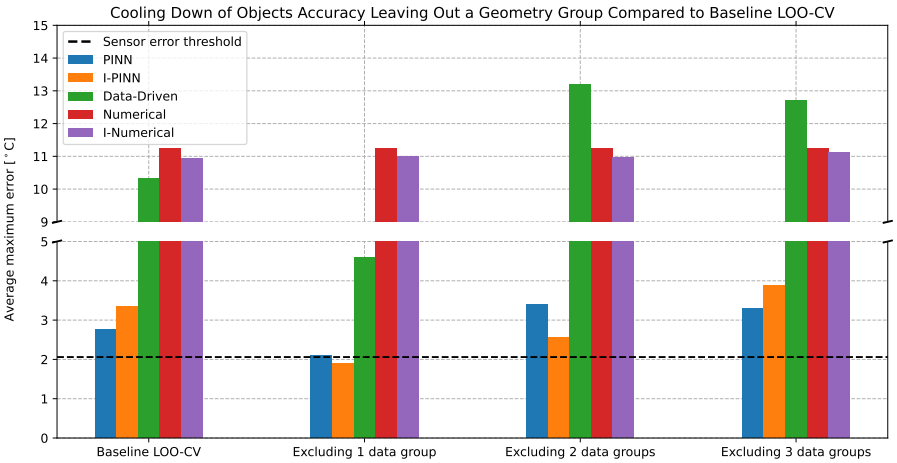


Figure 9.9: Bar chart presenting the average maximum error against various data partitioning strategies for the geometry groups for the various modelling methods

From Figure 9.9, it can be seen that sometimes, the models including a data-driven component will perform better compared to the baseline LOO-CV, sometimes even reaching below the threshold. This is the case for the PINN and I-PINN if 1 of the geometry groups is excluded during the training process. However, if more geometry groups are excluded, the maximum error increases for all modelling methods except the numerical models. This is expected since the data-component of the models will not be

able to capture the general trends, but it can be seen that compared to a purely Data-Driven method, the PINN and I-PINN do increase accuracy for surrogate modelling, only having a slight increase in average maximum error when excluding 3 geometry groups during the training phase compared to the baseline LOO-CV case.

To conclude, when taking into account both the average RMSE and the average maximum error, it can be seen that the PINN and I-PINN can reach satisfactory performance when excluding 1 geometry group. However, when excluding more geometry groups from the training phase, the methods will go above the threshold. This is expected since the more data is available to methods involving a data-driven component, the better the model can perform. Nevertheless, comparing the baseline LOO-CV with the exclusion of 3 geometry groups during the training phase only shows a slight decrease in accuracy, between 0.4 to 0.5 [°C]. Besides this, it can be seen that compared to the purely Data-Driven method and Numerical methods, the PINN and I-PINN do have a better capability in terms of accuracy for surrogate modelling when leaving out geometry groups.

9.3.3. Ambient Temperature

Lastly, the ambient temperature groups will be varied to draw a conclusion related to the surrogate modelling of the various modelling methods. To do so, the average RMSE and average maximum error will again be used, excluding an ambient temperature group from the training process, while comparing the results to a standard baseline LOO-CV. In Figure 9.10, the average RMSE are presented for the various modelling methods for the baseline LOO-CV and the exclusion of 1 ambient temperature group, while in Figure 9.11, the same analysis is performed for the average maximum error.

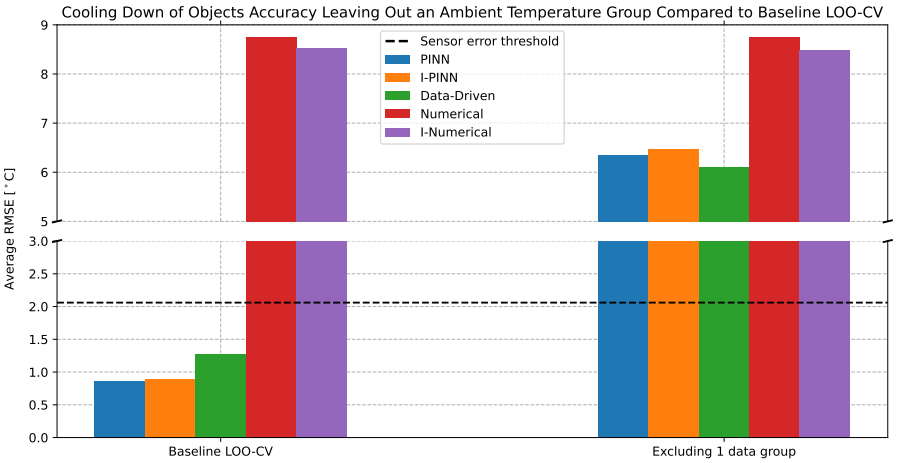


Figure 9.10: Bar chart presenting the average RMSE against various data partitioning strategies for the ambient temperature groups for the various modelling methods

From Figure 9.10, it can be seen that the average RMSE for the baseline LOO-CV is lower compared to the situation where 1 ambient temperature data group is excluded, with a decrease in accuracy of about 4.8 to 5.6 [°C] for the models containing a data-driven component. The models containing a data-driven component do still have a higher accuracy in terms of average RMSE compared to the Numerical methods. The purely Data-Driven method has a slightly higher accuracy in terms of average RMSE compared to the PINN and I-PINN methods, but these results are not statistically significant. All modelling methods fail to obtain an accuracy below or close to the threshold set by the sensor's accuracy.

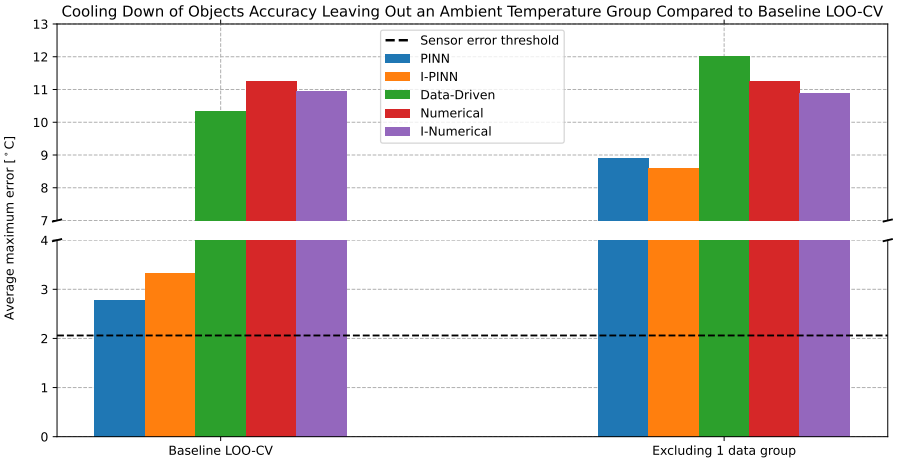


Figure 9.11: Bar chart presenting the average maximum error against various data partitioning strategies for the ambient temperature groups for the various modelling methods

From Figure 9.11, it can be seen that, as with the average RMSE, excluding 1 ambient temperature group decreases the accuracy of the average maximum error as compared to the baseline LOO-CV. However, it can be seen that compared to the purely Data-Driven and Numerical methods, the PINN and I-PINN do have a slightly higher accuracy in terms of average maximum error. However, all methods fail to reach the threshold set by the sensor's accuracy.

To conclude, excluding 1 ambient temperature group during the training phase shows that the various modelling methods are not able to provide accurate predictions. Although showing a slight improvement for the PINN and I-PINN compared to the Numerical methods, none of the models can provide accurate predictions in terms of average RMSE and average maximum error. The most likely explanation for these observations originates from 2 parts:

- The data-driven component of the modelling methods
  From Figure 9.10 and Figure 9.11, it can be seen that the purely Data-Driven model is not able to capture the general trend when leaving out an ambient temperature group during the training phase. The most likely reason for this is that the model does not know how to respond to other ambient temperatures, meaning that the model will only learn the relation with smaller temperature differences. To solve this, a wider range of temperatures should be introduced during the training phase to ensure the model can capture this relationship. Since a data-driven component is also part of the PINN and I-PINN, this part of the model becomes the limiting factor in terms of the performance of these modelling methods.
- The limitations of the numerical model and boundary conditions
  Although the PINN and I-PINN show some improvement compared to the purely Data-Driven and Numerical models, as can be seen from Figure 9.10 and Figure 9.11, the model is still not able to provide sufficiently accurate predictions. Previously, it was discussed that the data-driven component of the two modelling methods could potentially decrease the performance. However, another likely cause is the limitation of the numerical model and boundary conditions. Although the ambient temperature is incorporated in the numerical model, as was presented in Equation 7.1, it is not explicitly stated how the model should behave for varying ambient conditions, as the ambient temperature is only a small part of the physical model. A possible way to enforce this into the model is to add a far-field boundary condition, which enforces the predictions of the far field to be equal to the ambient temperature.

To conclude this section, there are strong indications that the PINN and I-PINN can be used as surrogate modelling methods. A PINN and I-PINN also provide the most accurate predictions compared to a purely Data-Driven model and Numerical models. However, it was found that the ability to act as a

surrogate model is dependent on how well the numerical model describes the physical model and if sufficient boundary conditions are enforced.

9.3.4. Data Table Surrogate Modelling

In this subsection, the data used and presented in section 9.3 are presented in more detail in Table 9.2.

Table 9.2: Comparison of Cooling Down of Objects for surrogate modelling: Computational Efficiency, Data Efficiency, and Accuracy across the various used methods

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Computational Efficiency					
Train Time (using 280 data points per file) [s]	268.171536	267.904055	18.903370	x	x
Train Time (using 40 data points per file, excluding 1 group material) [s]	124.803468	126.877200	16.988167	x	x
Train Time (using 40 data points per file, excluding 1 group geometry) [s]	244.431576	247.703592	19.492556	x	x
Train Time (using 40 data points per file, excluding 2 groups geometry) [s]	168.733814	166.879868	17.890594	x	x
Train Time (using 40 data points per file, excluding 3 groups geometry) [s]	91.838533	92.350542	16.34273	x	x
Train Time (using 40 data points per file, excluding 1 group ambient temperature) [s]	166.003013	169.100671	18.407101	x	x
Execution Time (using 280 data points per file) [s]	0.001307	0.001329	0.001440	0.000534	0.000402
Execution Time (using 40 data points per file, excluding 1 group material) [s]	0.001500	0.001326	0.001331	0.000543	0.000397
Execution Time (using 40 data points per file, excluding 1 group geometry) [s]	0.001567	0.001405	0.001475	0.000484	0.000400
Execution Time (using 40 data points per file, excluding 2 groups geometry) [s]	0.001513	0.001463	0.001459	0.000550	0.000426
Execution Time (using 40 data points per file, excluding 3 groups geometry) [s]	0.001465	0.001361	0.001356	0.000469	0.000392
Execution Time (using 40 data points per file, excluding 1 group ambient temperature) [s]	0.001566	0.001500	0.001448	0.000478	0.000394
Accuracy					
Avg. RMSE (using 280 data points per file) [°C]	0.858620	0.888685	1.27002	8.738852	8.521627
Avg. Max Error (using 280 data points per file) [°C]	2.772079	3.336031	10.325869	11.259344	10.937781
Avg. RMSE (using 40 data points per file, excluding 1 group of materials) [°C]	1.644811	1.552817	2.527845	8.738852	8.724829

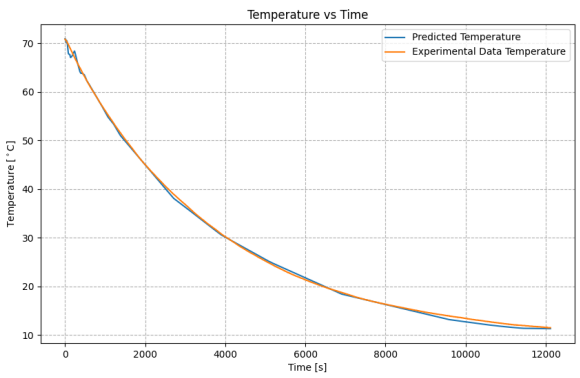
Continued on next page

Table 9.2 – continued from previous page

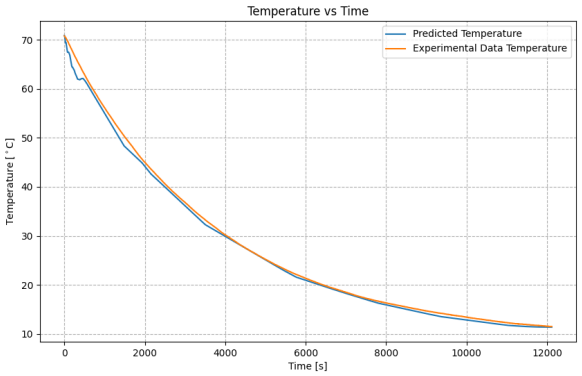
Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Avg. Max Error (using 40 data points per file, excluding 1 group of materials) [°C]	4.872826	3.102285	28.255650	11.259344	11.256333
Avg. RMSE (using 40 data points per file, excluding 1 group of geometry) [°C]	0.809077	0.866008	0.789851	8.738852	8.56446
Avg. Max Error (using 40 data points per file, excluding 1 group of geometry) [°C]	2.090795	1.9034276	4.595336	11.259344	11.001467
Avg. RMSE (using 40 data points per file, excluding 2 groups of geometry) [°C]	1.061543	0.900006	1.249411	8.738852	8.547767
Avg. Max Error (using 40 data points per file, excluding 2 groups of geometry) [°C]	3.391532	2.559428	13.193352	11.259344	10.977850
Avg. RMSE (using 40 data points per file, excluding 3 groups of geometry) [°C]	1.209055	1.271043	1.619660	8.738852	8.648927
Avg. Max Error (using 40 data points per file, excluding 3 groups of geometry) [°C]	3.292621	3.882659	12.718592	11.259344	11.128553
Avg. RMSE (using 40 data points per file, excluding 1 group of ambient temperature) [°C]	6.349252	6.457102	6.10709	8.738852	8.480991
Avg. Max Error (using 40 data points per file, excluding 1 group of ambient temperature) [°C]	8.907026	8.592429	12.014208	11.259344	10.875362

9.3.5. Visual Comparison Example

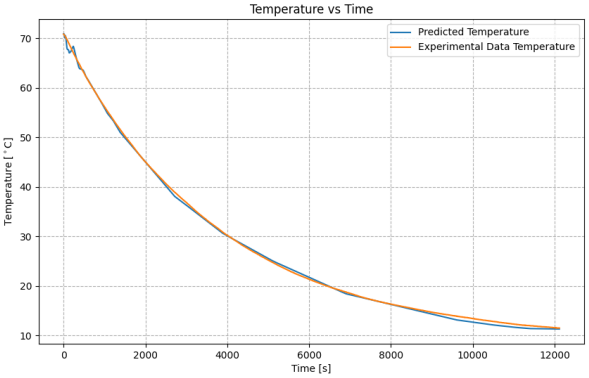
After comparing the various modelling methods using numbers, a visual comparison of the various methods will be presented. Using the results obtained in subsection 9.3.1, various graphs of predictions for the temperature against a test data set were generated. For this example, it was decided to compare the baseline LOO-CV trained using 280 data points against the models containing a data-driven component, which were being trained on a single materials group, namely the ”Clay” group, using 80 data points. This result will then be compared against the experimental data file CDOB_08.csv, a data file from the ”Glass” group. This visual overview of the various methods is presented in Figure 9.12. On the left of these figures, the baseline LOO-CV figures are presented, while on the right, the models trained on only the ”Clay” materials group are presented.



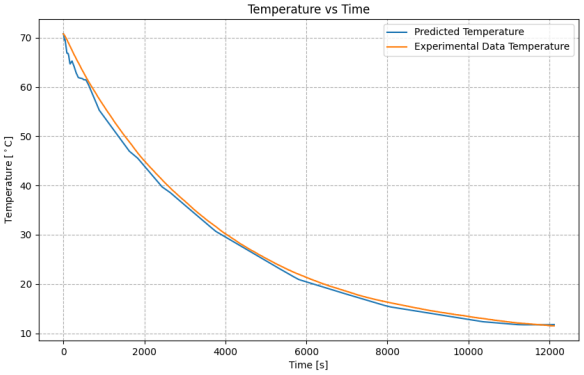
(a) PINN trained on 280 data points with CDOB_08.csv as test file. The RMSE is 0.436310 [°C], the maximum error is 1.852916 [°C], and the training time is 286.055203 [s]



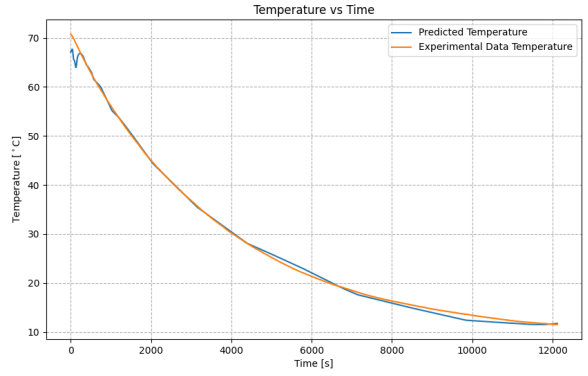
(b) PINN trained on 80 data points using only the materials group, with CDOB_08.csv as test file. The RMSE is 0.797915 [°C], the maximum error is 3.602167 [°C], and the training time is 90.009498 [s]



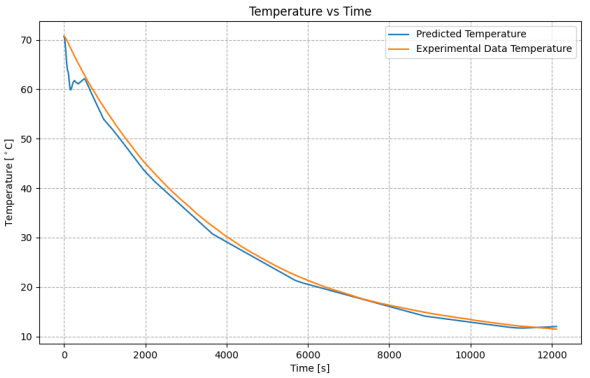
(c) I-PINN trained on 280 data points with CDOB_08.csv as test file. The RMSE is 0.443503 [°C], the maximum error is 1.853183 [°C], and the training time is 277.165709 [s]



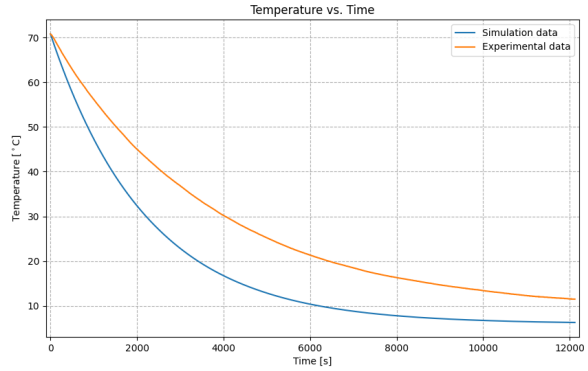
(d) I-PINN trained on 80 data points using only the materials group, with CDOB_08.csv as test file. The RMSE is 0.955689 [°C], the maximum error is 3.565083 [°C], and the training time is 91.217376 [s]



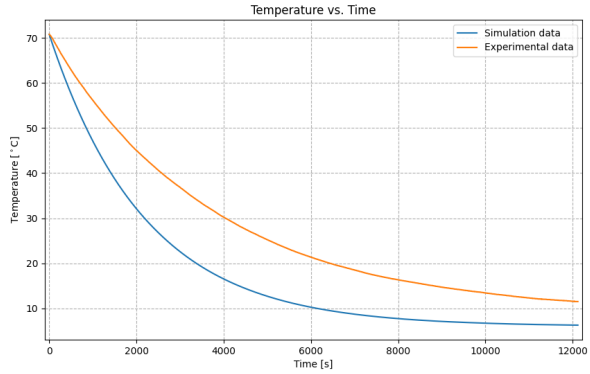
(e) Data-driven method trained on 280 data points with CDOB_08.csv as test file. The RMSE is 0.713778 [°C], the maximum error is 4.977436 [°C], and the training time is 19.202922 [s]



(f) Data-driven trained on 80 data points using only the materials group, with CDOB_08.csv as test file. The RMSE is 1.326335 [°C], the maximum error is 8.608839 [°C], and the training time is 16.790578 [s]

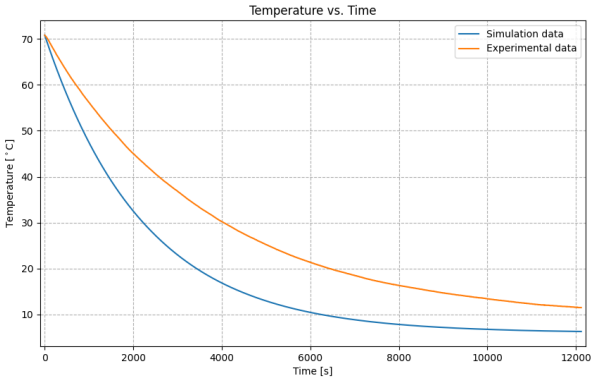


(g) Numerical method with CDOB_08.csv as test file. The RMSE is 8.268517 [°C], and the maximum error is 10.824917 [°C]

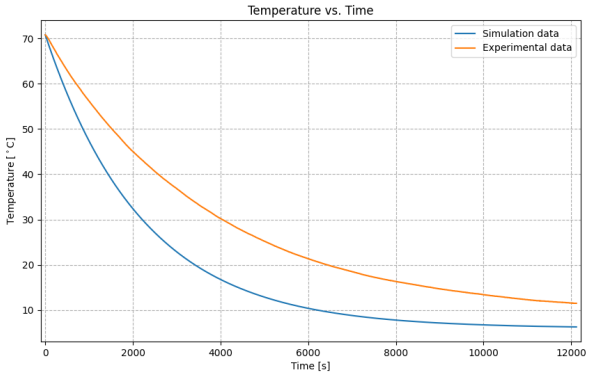


(h) Numerical method with CDOB_08.csv as test file. The RMSE is 8.268517 [°C], and the maximum error is 10.824917 [°C]

Figure 9.12: Comparison between a trained PINN, I-PINN, Data-driven, Numerical and I-Numerical method against a test file for 2 types of training methods (Part 1/2)



(i) I-Numerical method with CDOB_08.csv as test file. The RMSE is 8.194104 [°C], and the maximum error is 10.687941 [°C]



(j) I-Numerical method with CDOB_08.csv as test file. The RMSE is 8.257232 [°C], and the maximum error is 10.804127 [°C]

Figure 9.12: Comparison between a trained PINN, I-PINN, Data-driven, Numerical and I-Numerical method against a test file for 2 types of training methods (Part 2/2)

From Figure 9.12, it can be seen that for the baseline LOO-CV example, the PINN and I-PINN methods, which are presented in Figure 9.12a and Figure 9.12c, more closely resemble the experimental test data compared to the purely Data-Driven method as presented in Figure 9.12e and Numerical methods presented in Figure 9.12g and Figure 9.12i in terms of maximum error. To clarify, the test file CDOB_08.csv was not included during the training phase of the models. The main discrepancy between the PINN, I-PINN and the purely Data-Driven method lies in the fact that the purely Data-Driven method presented in Figure 9.12e seems to fail to find the correct starting position. The PINN and I-PINN presented in Figure 9.12a and Figure 9.12c can find the starting position but have some fluctuations at the start of their respective domains. After evaluation, the most likely cause of this is due to the conflicting data-driven part of the model and the incorporated physical model. This will largely smooth out the curves, but it could be possible that in certain regions, the data-driven part is more dictating compared to the physical model. This could be potentially solved by increasing the weight of the physics loss function. Another possible explanation for this phenomenon is that certain data points were present during training, which slightly overfitted the data. Both of the Numerical models presented in Figure 9.12g and Figure 9.12i provide a lower temperature prediction compared to the experimental test data. This showcases that the models containing a data-driven part are providing more accurate predictions compared to the Numerical methods, with the difference that the PINN and I-PINN can correctly predict the starting position, while the purely Data-Driven method is not able to provide this prediction.

Focusing on the training on the ”Clay” material figures on the right of Figure 9.12, it can be seen that all the methods involving a data-driven component seem to start at the correct position. Besides some small offset at the start of the domain for the PINN and I-PINN as presented in Figure 9.12b and Figure 9.12d, these modelling methods overall seem to provide a good agreement between their predictions and experimental test data. It should be noted, however, that the I-PINN presented in Figure 9.12d does have a slight upward trend at the end of the domain, while this is not the case for the PINN presented in Figure 9.12b. The purely Data-Driven method presented in Figure 9.12f has a similar prediction capability, showing a good agreement with the experimental test data. However, at the start of the domain, there is a large discrepancy between the experimental data and the predictions. The most likely explanation for this observation is that there were not enough data points present in this regime during the training phase to be able to get a good generalisation. At the end of the domain, the same upward trend as spotted for the I-PINN presented in Figure 9.12b can be seen, which most likely has a similar explanation. Both Numerical models are presented in Figure 9.12h and Figure 9.12j, likewise to the baseline LOO-CV case, underestimating the temperature predictions.

Part II

Computational Fluid Dynamics Modelling

10

Introduction: Computational Fluid Dynamics

In Part I, it was shown that using a PINN as a modelling method for the prediction of the temperature of an object over time has an increased accuracy and data-efficiency compared to purely Data-Driven and Numerical methods. Besides this, a PINN has a better modelling capability when making predictions outside the training data domain, making it a better modelling tool when data is scarce. To provide better prediction capabilities outside the training domain compared to a purely Data-Driven method, a PINN not only learn to predict values during the training phase, but also the derivative at this position. The derivative of the PINN is calculated analytically rather than numerically, as was explained in subsubsection 2.1.1.8. This has the advantage that the accuracy of the computer dictates the accuracy of the derivative. While PINNs act as a hybrid modelling method, meaning that they make use of both a Data-Driven and Numerical part, it is interesting to investigate the performance of a PINN, not making use of data during the training phase, solely relying on the underlying physical model. This way, the need for a computational mesh is not required, which eliminates truncation errors, as was explained in subsection 2.1.2. This could be a great advantage to domains where simulations are being performed using computational meshes, such as structural dynamics and fluid dynamics.

In this part of the report, the meshless ability of PINNs will be examined for the fluid dynamics domain, creating CFD simulations in the PINN architecture. To do so, first, a PINN model will be developed, without the use of experimental data, thereby training the PINN model solely on solving the underlying physical model and the provided boundary conditions. Normally, to validate CFD simulation, experimental data is used to compare the outputs of the numerical model to the real world. The experimental data can have various forms, such as pressure values, velocity values or forces and moments. After the comparison between the developed model and the experimental data has been completed, the model needs, most likely to be refined and tuned to be able to make model predictions which are more accurate compared to the experimental data. This tuning of the model can have various forms, such as refining the computational mesh, updating certain parameters or constants, such as flow velocities and boundary conditions. However, currently, there is no direct way to add the measured experimental data into the simulations to update the simulations for better prediction capabilities and, possibly, to show processes which can not be measured or are hard to measure on the object for traditional numerical methods. A PINN, on the other hand, can, besides learning the equations describing the underlying physical model and predicting values based on these equations, refine this prediction by incorporating measured experimental data. Besides the possibility for improved accuracy, computational efficiency and the decreasing time in pre-processing of CFD simulations by removing the need for mesh generations, adding experimental data to these types of simulations could open up a new interesting opportunity for CFD simulations. Therefore, the incorporation of experimental data in PINNs will also be investigated in this part of the report.

In this part of the report, the following sub-research questions will be answered to aid in answering the main research question as provided in chapter 1:

1. *To what extent can Physics-Informed Neural Networks serve as a general framework for solving the governing equations in Computational Fluid Dynamics without relying on data using shallow neural networks, compared to a traditional Numerical method?*
2. *To what extent can a hybrid model, using a Physics-Informed Neural Network without relying on data, improve the accuracy of a Computational Fluid Dynamics simulation compared to a traditional Numerical method when validated against experimental data for the section lift coefficient?*
3. *To what extent can a hybrid model using a Physics-Informed Neural Network, informed on experimental data of the pressure coefficient, improve the accuracy of a Computational Fluid Dynamics simulation compared to a traditional Numerical method when validated against experimental data for the section lift coefficient?*

This part of the report is structured as follows. First, a literature review will be presented in chapter 11 to obtain missing information regarding CFD modelling and a physical model which could be used for the PINN model. Next, experimental data, which later on will be used to train the PINN and used to compare the various CFD modelling methods, will be presented and discussed in chapter 12. After this, the setup of the PINN model will be explained in chapter 13. Lastly, the performances of the various CFD modelling methods will be examined in chapter 14.

11

Literature Review: Computational Fluid Dynamics

To perform simulations on problems related to the interaction between fluids and objects, Computational Fluid Dynamics simulations (CFD) [11] can be used. These types of models simulate the behaviour of a fluid based on governing equations of the fluid, making use of numerical methods [27]. The fluid's flow, heat transfer, mass transfer, chemical reactions and other related phenomena can be predicted by these types of simulations [11]. Although the term fluids is being used, CFD simulations can be performed on liquids, as well as gases [27]. CFD simulations could provide an in-depth insight into the interactions between fluids and objects, which could sometimes lead to more insights compared to experiments [11]. However, it should be noted that there is no simulation which could substitute a real-world test since unforeseen interactions between the real world and the object could lead to poor simulation results [82]. Furthermore, a simulation should always be validated by making use of experimental data before using a simulation to make important design decisions [82].

In this chapter, background information on CFD simulations will be provided. First, the governing equations describing fluid dynamics will be examined in section 11.1. Next, the various discretisation schemes used to solve the governing equations will be explained in section 11.2. After this, the design of computational meshes will be examined in section 11.3. Next, various types of boundary conditions used to enforce certain conditions at the boundaries of the computational domain will be discussed in section 11.4. After this, typical constants which are used for validation for aerodynamics simulations will be examined in section 11.5. Next, the current state-of-the-art for CFD simulations and PINNs will be examined in section 11.6, from which gaps in the literature will be explored. Lastly, an overview of the sources used during this chapter will be provided in section 11.7.

11.1. Navier-Stokes Equations

The governing equations describing the behaviour of fluids and their interactions with objects are the Navier-Stokes equations [86]. These equations are the backbone of all CFD simulation tools [87]. The Navier-Stokes equations describe the relation between the velocity, pressure, density and temperature of a moving fluid [86]. The core of the Navier-Stokes equations consists of three main equations, a continuity equation, a momentum equation and an energy equation. The core Navier-Stokes equations in vector form for a 3-dimensional unsteady flow are provided in Equation 11.1 [86]. The continuity equation describes the conservation of mass of the fluid, making sure that no mass can be created or destroyed [86]. The momentum equation describes how the momentum of the fluid is conserved [87]. This is effectively a rewritten version of Newton's second law of motion applied to a fluid element [87]. The energy equation describes the conservation of energy within the system and is effectively the applied version of the first law of thermodynamics [86, 131]. The energy equation is mainly important for problems involving a type of heat transfer mechanism [131], which means that the energy equation is not always required to be able to perform a CFD simulation.

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 \\ \rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \right) &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \mathbf{f} \\ \frac{\partial (\rho E_t)}{\partial t} + \nabla \cdot (\rho E_t \mathbf{u}) &= -\nabla \cdot \mathbf{q} + \nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{u}) - p(\nabla \cdot \mathbf{u}) \end{aligned} \tag{11.1}$$

In Equation 11.1, ρ represents the density of the fluid, $\mathbf{u}$ represents the velocity vector, p represents the pressure, t represents the time, μ represents the dynamic viscosity, $\boldsymbol{\tau}$ represents the viscous stress tensor, $\mathbf{f}$ represents the external forces applied to the fluid, E_t represents the total energy of the system, T represents the temperature, k represents the thermal conductivity and $\mathbf{q}$ represents the heat flux vector [87, 86, 131].

The equations as presented in Equation 11.1 are not the only equations considered when solving CFD equations. Mostly, a state equation is added to the set of equations to better grasp the underlying relationships [87, 86]. Furthermore, depending on the problem, other equations are being added [86, 87, 131]. These equations are being added such that the problem will be fully described.

As can be seen from Equation 11.1, quantities within the Navier-Stokes equations are linked together and, therefore, are hard to solve analytically [86]. To solve this issue, numerical schemes and boundary conditions have to be introduced, and a set of simplifications can be introduced, depending on the type of problem. For example, when modelling a 2D subsonic problem, an additional set of assumptions can be introduced [12]. These assumptions are provided in the list below:

- The simulation is steady-state
- The simulation models an incompressible flow
- The simulation describes a two-dimensional flow
- The simulation models an isothermal process

When introducing these additional assumptions to the core Navier-Stokes equations, Equation 11.1 reduces to Equation 11.2 [12].

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0 \\ (\mathbf{u} \cdot \nabla)\mathbf{u} &= -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} \end{aligned} \tag{11.2}$$

In Equation 11.2, ν represents the kinematic viscosity of the fluid. Equation 11.2 can now be used for equations describing an incompressible, steady-state, two-dimensional flow.

11.2. Discretization Schemes

As mentioned in section 11.1, the Navier-Stokes equations can not be solved analytically. Therefore, use needs to be made of a numerical method to solve the differential equation representation of the underlying physical model [27]. In CFD simulation, use can be made of a discretisation scheme, which is a way of rewriting the governing equations such that they can be solved numerically [27]. In general, there are three different discretisation schemes, namely the Finite Element Method (FEM), the Finite Volume Method (FVM) and the Finite Difference Method (FDM) [27]. A visual comparison of the various discretisation schemes is presented in Figure 11.1. In the upcoming sections, the various discretisation schemes will be examined in more detail.

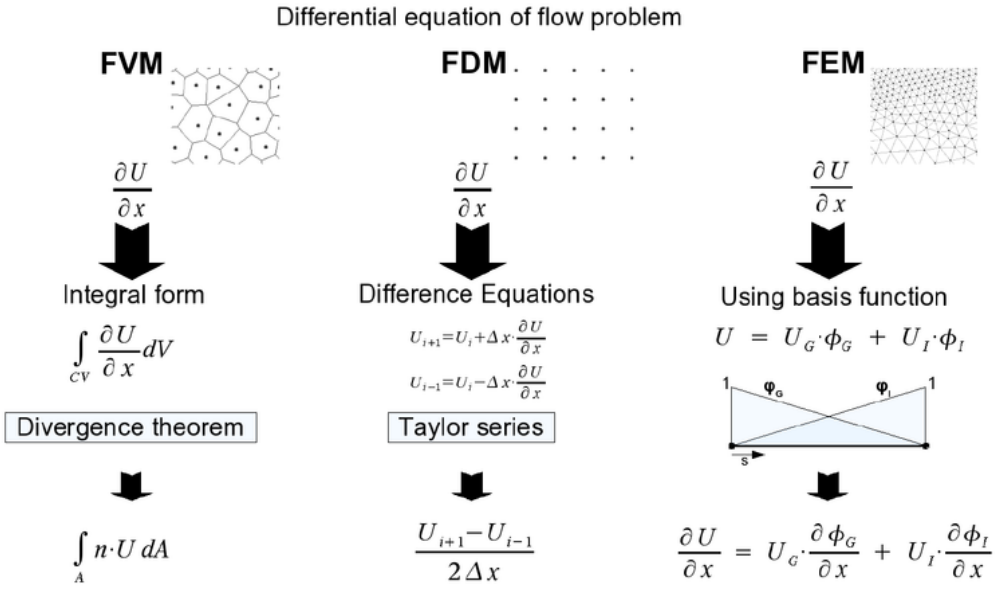


Figure 11.1: Visual comparison between the discretization schemes FVM, FDM, FEM [40]

11.2.1. Finite Element Method

FEM is mostly used in structural dynamics for simulations obtaining stresses, strains and other important material behaviours when applied under a certain load. The FEM method works as follows. First, the domain is split up into various small sections, whereafter the equations are solved on each element, making use of interpolation [38]. The advantages of FEM are that by making use of this method, complex geometries can be accurately represented and FEM is very flexible [38]. The downside, however, is that FEM is much more computationally demanding compared to FVM and FDM [38].

11.2.2. Finite Volume Method

FVM is a discretisation scheme mostly used in CFD simulations and works as follows [27]. Rather than splitting the domain into various sections as with the FEM method, the computational domain is being split up into various small control volumes when making use of FVM [39]. After this process has been completed, the differential equations are solved by integration over each control volume, ensuring that the fluxes in and out of the individual control volumes adhere to the various conservation laws for mass, momentum and energy [39]. The advantages of FVM are that it is very suitable for unstructured grids and that it works better for conservation equations, such as mass and momentum [39]. The major downsides of FVM are that it is more computationally demanding and is harder to implement for complex geometries [39].

11.2.3. Finite Difference Method

Rather than splitting the domain into sections or control volumes, FDM works by dividing the domain into discrete points using a mesh. After this process has been completed, the governing equations are solved by calculating the embedded derivatives in the equations by making use of the differences between values of the discrete points of the mesh [37]. Depending on the direction of the simulation, use can be made of a forward, backwards or central difference scheme [37]. The biggest advantage of this method is that it is relatively simple to use on structured grids [37]. However, that is also its major downside since it is harder to use with complex geometries [37].

As previously mentioned, FDM makes use of either forward, backwards or central difference schemes [37]. These schemes approximate derivatives by making use of differences between values [37]. The definition of a derivative is provided in Equation 11.3 [58].

$$\frac{d}{dx}f(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h} \tag{11.3}$$

In Equation 11.3, $f(x)$ represents the function evaluated at point x, $f(x+h)$ represents the function evaluated at $x+h$ and h represents a small step size in the x variable [58]. The definition of the derivative as presented in Equation 11.3 can be approximated by the Taylor series in three ways, namely the forward difference method, the backwards difference method and the central difference method. The forward difference method estimates the derivative by using the next point in the mesh and the step size between the current step and the next step, which is presented in Equation 11.4 [58].

$$\frac{d}{dx}f(x_i) = \frac{f_{i+1}-f_i}{h} + \mathcal{O}(h) \tag{11.4}$$

The backwards difference method works similarly to the forward difference method but does not use the next point to evaluate the derivative, but the previous point to calculate, as is presented in Equation 11.5 [58].

$$\frac{d}{dx}f(x_i) = \frac{f_i-f_{i-1}}{h} + \mathcal{O}(h) \tag{11.5}$$

The central difference method, on the other hand, uses the previous and the next point to estimate the derivative, as is presented in Equation 11.6 [58].

$$\frac{d}{dx}f(x_i) = \frac{f_{i+1}-f_{i-1}}{2h} + \mathcal{O}(h^2) \tag{11.6}$$

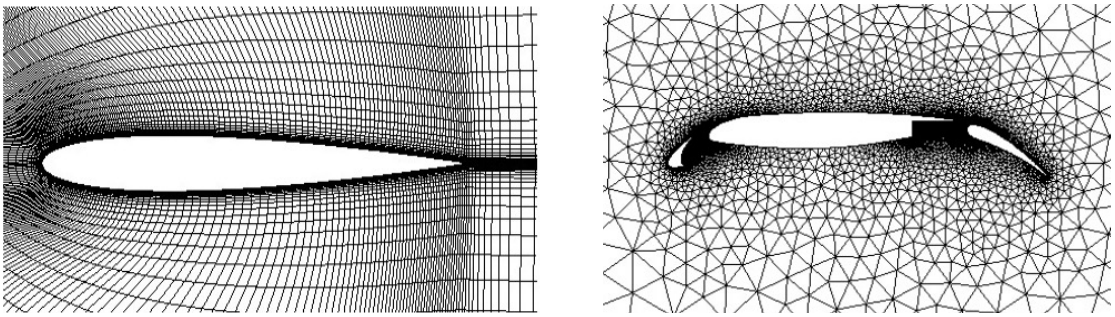
All of the provided methods have a major downside, namely that they estimate the derivative and can not provide an exact value of the derivative. This means that some sort of error is introduced when obtaining the estimation of the derivative [58]. This error will become smaller by decreasing the step size [58]. However, the accuracy is fixed due to a set step size, dictated by the distance of discrete points on the computational mesh by the finite difference method [58]. This means that to obtain a higher accuracy, a finer computational mesh is required, thereby increasing computational time [58]. In other words, to decrease the truncation error, the distance between the individual points in the computational mesh needs to be reduced. To do so, the mesh needs to be redesigned, making use of smaller step sizes. This increases computational time. In general, the error for each method is expressed in its order and is first order ($\mathcal{O}(h)$) for the first derivative using the forward and backward difference method, while it is second order ($\mathcal{O}(h^2)$) for the central difference method [58].

The choice of which differential scheme to use depends on the type of problem and available information. If only the current and previous values of a certain quantity are available, use will be made of a backwards scheme, also known as an implicit numerical method [58]. If, on the other hand, the current and future values are known, one could make use of the forward scheme, which is also known as an explicit numerical scheme [58]. The central difference scheme is being used when future and previous values are known, which has the advantage that a higher order of accuracy can be obtained [58].

11.3. Mesh Design

As explained in section 11.2, the Navier-Stokes equations provided in section 11.1 need to be discretised to be able to solve the governing differential equations. The discretisation includes splitting up the domain into a finite number of lines, cells or points, depending on the type of discretisation scheme used. The conversion from a continuous geometry and domain to a discretisation version is called mesh generation [102]. In general, two types of meshes exist, namely, structured and unstructured meshes. A structured mesh makes use of rectangles (2D) and hexahedra (3D) to split the domain and geometry, while an unstructured mesh makes use of all sorts of geometric shapes, mainly triangles (2D) and tetrahedra (3D) [49]. An example of a structured and unstructured domain and geometry is presented in Figure 11.2. The reason to choose a structured or an unstructured mesh depends on the

type of problem and the available resources [102]. For example, an unstructured grid is more suitable for complex geometries but is computationally more expensive [49].



(a) Structured mesh [49]

(b) Unstructured mesh [49]

Figure 11.2: Example of a structured and unstructured mesh

It is important to consider that the closer one gets to the object, the smaller the computational grid should be [102]. The reason for this is that if the cells are large and close to the object, the interaction between the object and the fluid will not be modelled correctly, since the calculation will be performed too far from the object, leading to underestimating of the effects of the objects influence to the flow and incorrect modelling of the boundary conditions [49, 102]. When one has completed the generation of a mesh and obtained some results, a mesh sensitivity study should be performed [102]. During a mesh sensitivity study, the mesh will be made finer or coarser to investigate if the grid influences the result. In other words, it is investigated whether decreasing the step size between the points or volumes influences the solution. In case this is true, one should redo the mesh and aim for a finer mesh [102] since the results should be independent of the mesh at some point to have reliable simulation results [102].

11.4. Boundary Conditions

When making use of a numerical model to solve the governing equations as described in for CFD simulations section 11.1, a set of boundary conditions at the boundaries of the computational domain and objects are required to provide the solver with a set of conditions which has to be met [50]. In general, the following boundary conditions exist [50]:

- Dirichlet boundary conditions
  This type of boundary condition imposes a certain fixed value on a quantity at a boundary of the domain [50].
- Neumann boundary conditions
  This type of boundary condition imposes a gradient of a certain quantity at a boundary of the domain [50].
- Robin boundary conditions
  This type of boundary condition is a mix between Dirichlet and Neumann boundary conditions, which means that at these boundaries, both types will occur [50].
- Periodic boundary conditions
  This type of boundary imposes a fixed value to a certain surface, which should be the same at the two opposite boundaries [50].

The previously mentioned boundary conditions will be positioned at the boundaries of the domain, as well as at the boundaries between the object and the fluid [50]. The reason for this is that this will help the solver to understand there is an object present in the flow of the domain, and how the flow should interact with it [50]. An example of a boundary condition at the surface of an object is the no-slip condition. This boundary condition is mostly opposed between the surface of an object of interest and the fluid, to ensure that the velocities in x- and y-directions are zero at the object's boundaries, since no fluid can go through the object [50]. This boundary condition consists of a Dirichlet boundary

condition for the velocity and a Neumann boundary condition for the pressure [50]. It should be noted that boundary conditions are an approximation of reality. This means that, to limit nonphysical effects, boundary conditions should be imposed as far as possible from the domain of interest [50].

11.5. Important Aerodynamic Constants

Aerodynamics is a domain of fluid dynamics which studies the motion of air when being affected by solid objects such as the wings of an airplane [6]. In general, there are 4 different types of domains for which aerodynamic simulations can be performed, based on the velocity regime of the flow [64]. To identify to which regime a certain flow belongs, use can be made of the Mach number, which can be obtained by making use of Equation 11.7 [65].

$$M = \frac{V}{\sqrt{\gamma R T}} \tag{11.7}$$

In Equation 11.7, M represents the Mach number, V represents the velocity of the flow, γ represents the specific heat ratio of the gas, R represents the specific gas constant and T represents the static air temperature of the flow [65]. The different flow regimes, which are being identified based on their Mach number, are listed below [64]:

- Subsonic flow ($M < 1$)
- Sonic flow ($M = 1$)
- Supersonic flow ($1 < M < 5$)
- Hypersonic flow ($M > 5$)

For the various flow regimes presented in the list, simplifications can be made to the governing equations. For example, if the flow has a Mach number below 0.3, the flow is approximately incompressible, meaning that the density of the air can be considered to be constant [26]. If the flow becomes sonic, having a Mach number of 1, a shockwave can occur [26]. Besides the different velocity regimes, another important quantity of the flow is whether the flow can be considered laminar or turbulent. Laminar flows are flows where no vortices are present, while in turbulent flows, vortices are present in the flow. To identify if a flow is laminar or turbulent, use can be made of the Reynolds number, which can be obtained by making use of Equation 11.8 [135].

$$Re = \frac{VL}{\nu} \tag{11.8}$$

In Equation 11.8, Re represents the Reynolds number, V represents the characteristic velocity of the fluid, L represents the characteristic length scale and ν represents the kinematic viscosity of the fluid [135]. When performing CFD simulations, it is important to set the conditions for the Mach and Reynolds numbers equal to the ones recorded during experiments in, for example, a wind tunnel. When the Reynolds number is below 2300 [-], the flow can be considered laminar, while if the Reynolds number is above 4000 [-], the flow is considered to be turbulent.

To be able to make an airplane fly, an upwards force is required [34]. This upward force is called the ”lift” force. The lift force is generated due to the difference in flow velocity between the upper and lower parts of the airplane’s wings, causing a difference in pressure between the upper and lower parts of the airplane’s wings, which results in a resulting upward force [34]. Airplane wings have different shapes because a certain wing shape has a certain performance [9]. The specific shape of the cross-section of a wing is called an airfoil [9]. To determine the general performance of an airfoil, as measured in a wind tunnel, the sectioned lift coefficient can be calculated by using the formula as presented in Equation 11.9 [62].

$$C_l = \frac{L'}{\frac{1}{2}\rho V^2 c} \tag{11.9}$$

In Equation 11.9, C_l represents the section lift coefficient for the airfoil, ρ represents the density of the flow, V represents the flow's velocity, c represents the chord length of the airfoil and L' represents the lift force per unit span of wing generated by the airfoil [62]. In wind tunnels, the lift force is obtained using scales, measuring the upward force [62]. While the lift coefficient is an important parameter for the airfoil, the pressure distribution along the airfoil is also of interest. To determine how the pressure is behaving on the surface of the airfoil, small holes in the airfoil model can be made, which will be connected to pressure sensors, which can then measure the pressure on the surface of the airfoil [129]. An example of a setup using pressure tabs in the airfoil can be seen in Figure 11.3.

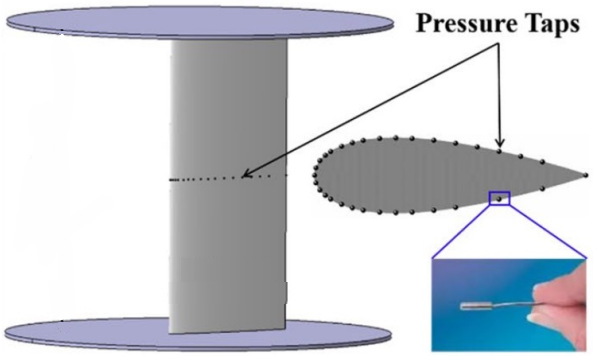


Figure 11.3: Example of pressure tabs to measure the static pressure at fixed positions on the airfoil [129]

To be able to generalise the measured pressure distribution, meaning that the distribution is independent of certain flow characteristics such as density and velocity, use can be made of the pressure coefficient, which can be obtained by making use of Equation 11.10 [97].

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty V_\infty^2} \tag{11.10}$$

In Equation 11.10, p represents the static pressure measured at the pressured tab, p_∞ represents the static pressure of the freestream flow, ρ_∞ represents the density of the freestream flow and V_∞ represents the velocity of the freestream flow [97]. Since the pressure difference between the upper and lower parts of the airfoil causes a lift force [34], there is a conversion formula from the pressure coefficient to the section lift coefficient as presented in Equation 11.9. To do so, use can be made of an approximation, which is presented in Equation 11.11 [97].

$$C_l = \frac{1}{x_{\mathsf{TE}} - x_{\mathsf{LE}}} \int_{x_{\mathsf{LE}}}^{x_{\mathsf{TE}}} \left(C_{p_l}(x) - C_{p_u}(x)\right)\, dx \tag{11.11}$$

In Equation 11.11, C_l represents the section lift coefficient, x_{TE} represents the x-coordinate of the trailing edge, x_{LE} represents the x-coordinate of the leading edge, C_{p_u} represents the pressure coefficient on the upper part of the airfoil, C_{p_l} represents the pressure coefficient of the lower part of the airfoil and x represents the x-coordinate of the airfoil [97]. However, it should be noted that Equation 11.11 is an oversimplified way to obtain the section lift coefficient from the pressure coefficient distribution over an airfoil [2]. The reason for this is that in Equation 11.11, it is assumed that all the pressure components are acting in the normal direction, resulting in a normal force only. However, at angles of attack higher than 5 [°], this assumption is no longer valid [2]. The additional error introduced due to this approximation can be approximated by making use of Equation 11.12 [2].

$$Error \approx 0.1 \sin(\alpha) \tag{11.12}$$

In Equation 11.12, α is the angle of attack, which is the angle between the incoming airflow and the airfoil's chord line [10, 2]. Making use of Equation 11.12, it can be found that the introduced error will

be between 0 and 0.87% for angles of attack between 0-5 [°], reaching to 1.74% at an angle of attack of 10 [°] and 2.59% for an angle of attack of 15 [°].

11.6. Current Status of Machine Learning in CFD Simulations

As previously explained in subsection 2.1.2, PINNs have physical models embedded in their loss functions, meaning that PINNs can be used without experimental data [12]. This has the benefit that it can be used for CFD simulations, something which has recently been explored. It was shown that it is possible to develop a PINN without the need for experimental or simulation data from an external source, obtaining sufficient accuracy compared to traditional CFD solvers [12]. Furthermore, it was found that these types of simulations become more accurate by increasing the number of hidden layers in the NN, whereafter increasing the number of collocation points in the domain also leads to more accurate results compared to traditional simulations [12]. The best architecture currently found for these types of simulations is FNN, with a lot of hidden layers, ranging from 15-25 [12, 53]. It was found that the improvement in accuracy is the least when introducing additional neurons in the hidden layers [12]. However, sufficient prediction capability of PINNs for CFD simulations, making use of shallow NN, has not been found in the literature.

Compared to a traditional CFD simulation method, it was found that PINNs are 5-10 times more efficient in terms of memory usage, at the cost of taking about 3 times as long to perform the simulation, given dedicated hardware to perform the simulation on [12]. However, it was found that while for relatively simple cases, PINNs take a longer time to train compared to standard simulations, increasing the complexity of the problem does not lead to an increase in computational time for the PINN. However, this does increase computational time for the traditional method [12]. Most importantly, it was found that there should be a sufficient number of physical models present in the loss functions to be able to fully describe the problem. It was found that the PINN lacked accuracy in pressure terms since these were not directly described [12].

Comparing the simulated data in terms of drag coefficient, it was found that PINNs perform reasonably well, having errors within 10% compared to the traditional CFD methods [53]. Furthermore, it was found that the discrepancies in terms of counters in the pressure and velocity plots and, therefore, the difference in the drag coefficients are mainly due to boundary conditions and how they are being enforced. Where traditional methods enforce boundary conditions as hard constraints, PINNs trade boundary conditions as potential optimisation points, trying to reduce the error to zero [53]. Therefore, these conditions will never be fully enforced, leading to discrepancies between the simulation domains [53].

Currently, PINNs are not meant as a replacement for traditional methods but as a potential new addition to CFD simulations [53]. The main advantage of PINNs compared to traditional methods is that a PINN makes use of a meshless approach, meaning that time-consuming computational mesh design is not required [53]. Besides this, PINNs could incorporate experimental data, leading to more insights and accuracy compared to traditional CFD simulations, something which has not been broadly explored as of now. Besides this, PINNs have another advantage compared to traditional CFD methods. For optimisation purposes, PINNs can be directly used without the need for iterations and multiple simulations. For example, it was shown that the shape of turbomachinery blades can be optimised by making use of PINNs [125, 54]. This was done by utilising the gradients obtained during the training process of the PINN and using this in an optimisation function, for example, lift-to-drag ratio, to change the design parameters and optimise these parameters [125]. Besides low-speed applications, PINNs also show good agreement in the high-speed applications when compared to traditional solvers [68]. PINNs do not show, likewise to their low-speed counterparts, improvements in accuracy compared to traditional CFD simulation solvers [68]. However, the fact that PINNs can be used to obtain certain parameters exceeds the possibilities of current tools, which need sophisticated software loops to perform optimisation, while this can be incorporated into the PINNs framework [68].

11.7. Overview of Sources

Throughout the literature review, use has been made of various sources. To provide an overview of the sources and their domain, the sources have been summarised in Table 11.1.

Table 11.1: Overview of Sources and Their Usage in the Literature Review

Section	References
Computational Fluid Dynamics	
Introduction	[11], [27], [82]
Navier-Stokes Equations	[86], [87], [131], [12]
Discretization Schemes	[27], [40]
Finite Element Method	[38]
Finite Volume Method	[39]
Finite Difference Method	[37], [58]
Mesh Design	[102], [49]
Boundary Conditions	[50]
Important Aerodynamic Constants	[6], [64], [65], [26], [135], [34], [9], [62], [129], [97], [2], [10]
Current Status of Machine Learning in Computational Fluid Dynamics Simulations	[12], [53], [125], [68], [54]

12

Experimental Data Processing: Computational Fluid Dynamics

To be able to answer the research questions, experimental data is required to train the PINN and compare the PINN's performance against predictions from the CFD simulation models. After careful evaluation, it was decided to compare the section lift coefficient at 3 different angles of attack of the NACA 0012 airfoil, at a flow condition of $M = 0.3$, at a Reynolds number of $6 \cdot 10^6$ [-]. At the 3 angles of attack, the pressure coefficients at fixed positions on the airfoil have been recorded, which can later be used as inputs for the PINN model. This chapter will describe the experimental data sets for the section lift coefficient and pressure coefficient distribution around the airfoil. First, some background information about the NACA 0012 airfoil will be presented in section 12.1. Next, the recorded pressure coefficient distribution corresponding to the 3 different angles of attack will be presented in section 12.2. Lastly, the recorded section lift coefficients for various angles of attack will be presented in section 12.3.

12.1. NACA 0012 Airfoil

The presented experimental data in this chapter were measured making use of the NACA 0012 airfoil [61]. The presented experimental data is normally used for the validation of CFD simulations [123]. The NACA 0012 is an airfoil developed by the National Advisory Committee for Aeronautics, shortened NACA [77]. Between 1920 and 1940, NACA created a set of thoroughly tested airfoils [77]. To provide a systematic classification for these airfoils, a 4-digit number classification system was introduced, representing the geometry properties of a specific airfoil [77]. The numbering system provided an easy way to obtain certain airfoil characteristics [77].

The NACA 4-digit code works the following. The first digit describes the maximum camber as a percentage of the chord length [77]. The second digit describes the distance of the maximum camber, measured from the airfoil's leading edge, in distances of tenths of the chord [77]. The last two digits describe the maximum thickness of the airfoil as a percentage of the chord length [77]. A visual example of the meaning of all the digits is presented in Figure 12.1.

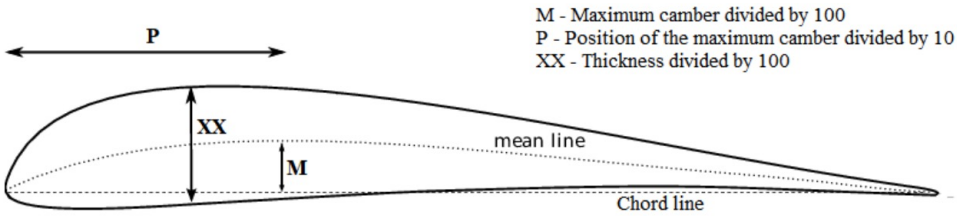


Figure 12.1: Visual explanation of the NACA 4-digit airfoil coding system, presented as NACA PMXX [17]

The experimental data was obtained making use of a NACA 0012 airfoil. Using the numbering system, this means that the airfoil does not have camber and the thickness is 12% of the chord length. The NACA 0012 airfoil and its discrete representation, which is used during the CFD and PINN simulations, is presented in Figure 12.2.

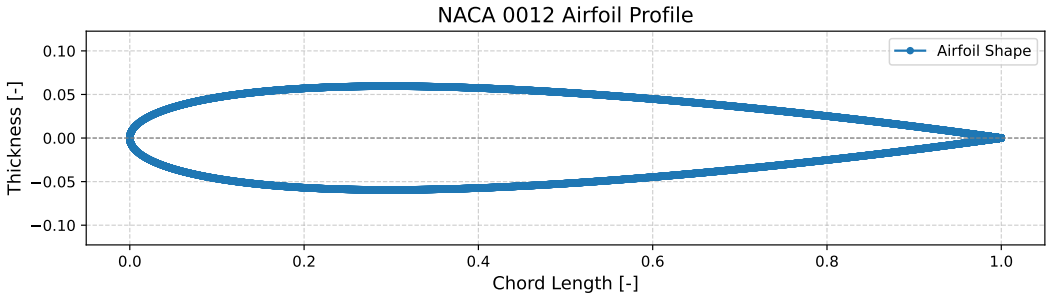


Figure 12.2: NACA 0012 discrete representation making use of various coordinates used in the CFD and PINN simulations.

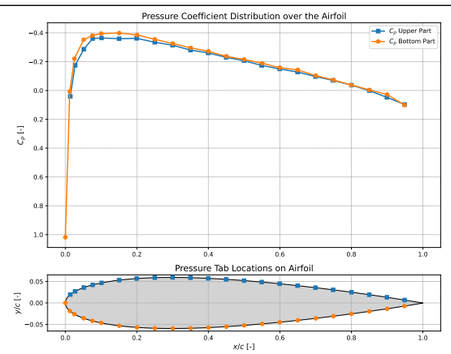
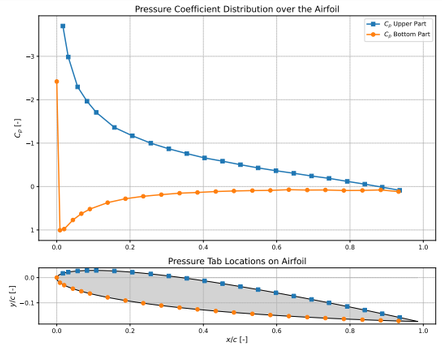
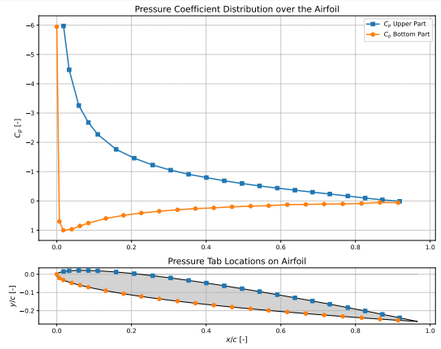
From Figure 12.2, it can be seen that sufficient coordinates are present to represent the airfoil profile. Furthermore, it can be seen that the chord length is scaled to 1 and that the maximum thickness is indeed 12% of the chord length, which should be the case according to the previously explained classification system.

12.2. Experimental Pressure Coefficient Distribution

Since the NACA 0012 is a well recorded airfoil, experimental data is publicly available for use in research or design [123, 61]. Making use of this data, the pressure coefficient distribution for various angles of attack has been recorded. This has been done by measuring the static pressure at small pressure tabs within the airfoil, as visually presented in Figure 11.3. The various recorded pressure distributions, including additional information for each experiment, are presented in Table 12.1. It should be noted that the individual data points on the C_p plot correspond to the pressure tap locations on the airfoil displayed directly below. For instance, the colored markers (orange for the lower surface and blue for the upper surface) on the airfoil indicate the exact positions where the C_p values were measured. Enlarged versions of the experimental data figures are presented in Appendix D, which can be accessed by pressing on the figure in the table [1].

[1] This is only possible in case this thesis is being viewed in a PDF viewer

Table 12.1: Pressure Coefficient Distribution over the NACA 0012 airfoil data overview

File name	Description		Amount of Data Points	C_p distribution over airfoil figure
Cp_alpha_0169_Re6.txt	C_p distribution around the NACA 0012 airfoil section, at $\alpha = 0.0169\ [°]$, $Re = 6 \cdot 10^6\ [-]$ and $M = 0.3\ [-]$	-	45	
Cp_alpha_10_0254_Re6.txt	C_p distribution around the NACA 0012 airfoil section, at $\alpha = 10.0254\ [°]$, $Re = 6 \cdot 10^6\ [-]$ and $M = 0.3\ [-]$	-	45	
Cp_alpha_15_0299_Re6.txt	C_p distribution around the NACA 0012 airfoil section, at $\alpha = 15.0299\ [°]$, $Re = 6 \cdot 10^6\ [-]$ and $M = 0.3\ [-]$	-	45	

12.3. Experimental Lift Curve of Section Lift Coefficient

As explained in section 12.2, the NACA 0012 is a well recorded airfoil and experimental data is publicly available for use in research or design [123, 61]. Making use of this data, the section lift curve of the NACA 0012 airfoil, at a Reynolds number of $6 \cdot 10^6$ [-] and a Mach number of 0.3 [-], can be generated [61]. The generated section lift curve, making use of the experimental data, is presented in Figure 12.3.

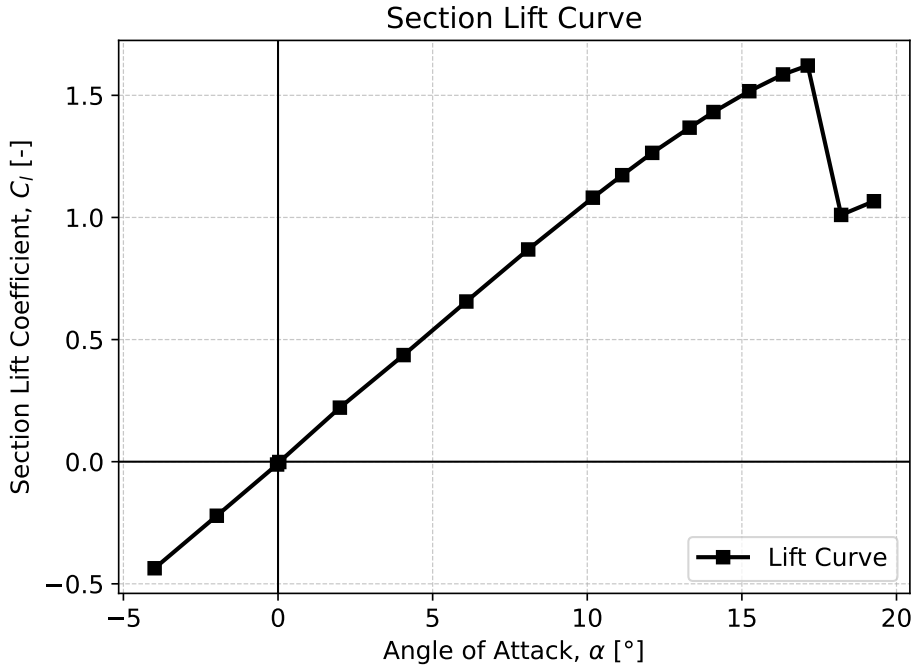


Figure 12.3: Section lift curve using experimental data obtained from [61]

Interpolating Figure 12.3, the angles of attack of interest for the to-be-conducted comparison can be obtained, which are presented in Table 12.2.

Table 12.2: Section lift coefficient for specific angles of attack for the NACA 0012 airfoil, at a flow condition of $Re = 6 \cdot 10^6$ and $M = 0.3$ [61]

Angle of Attack, α [°]	**Section Lift Coefficient, C_l [-]**
0.0169	-0.00466432
10.0254	1.06567681
15.0299	1.50574158

13

Physics-Informed Neural Network Model: Computational Fluid Dynamics

The goal of this part of the research is to investigate whether a PINN can be used for CFD simulations and compare this modelling method against a traditional numerical method. To do so, a PINN needs to be developed, which can perform CFD simulations. This chapter will describe the development of the PINN model, as well as a description of the network architecture and some example outputs the PINN can generate. In section 13.1, the network architecture of the PINN will be presented, as well as the implementation of the physics and boundary loss terms in Python code. Furthermore, a flow chart of the PINN will be presented, as well as the inputs of the PINN. Next, the setup of the computational domain and how this differs from a traditional numerical CFD simulation method will be explained in section 13.2. Lastly, an example of generated outputs by the PINN model will be presented in section 13.3.

13.1. Setup of the Model

After careful evaluation, using the results obtained from section 11.6, it was decided that an FNN was a suitable network architecture for CFD PINN models. This means that use can be made of the developed Python class as was explained in chapter 3. The developed PINN model will have 2 inputs, the x- and y-coordinates of the domain, meaning that the network will have 2 neurons in its input layer. Furthermore, the PINN model will have 3 outputs, namely the velocity component in the x direction (u-velocity), the velocity component in the y direction (the v-component) and the pressure, meaning that the PINN will have 3 neurons in the output layer. After some test runs were conducted, it was found that 2 hidden layers with 64 neurons per layer were required to obtain sufficient results. The hidden layers of the model make use of the Tanh activation function, as described in subsubsection 2.1.1.2, since the physical model used in the PINN, which is a simplified version of the Navier-Stokes equations as described in Equation 11.2, contains second-order derivatives. This means that an activation function is required, which is at least twice differentiable. When using activation functions which are only differentiable once, such as the ReLU activation function, the second-order derivatives will be zero. The initialisation method of the CFD PINN model makes use of normal Xavier initialisation for its weights after evaluating subsubsection 2.1.1.4. The biases are initialised to zero. The CFD PINN network can be visually seen in Figure 13.1.

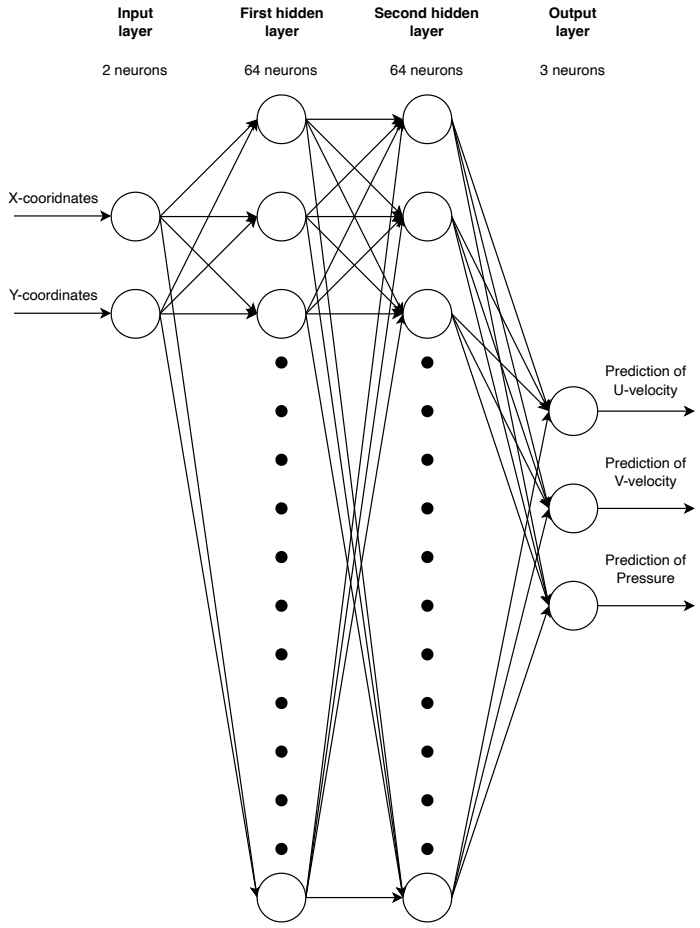


Figure 13.1: Neural network architecture for the Physics-Informed Neural Network developed for the Computational Fluid Dynamics simulation model

As with the cooling down of objects PINN, the developed class, as was presented in Figure 3.1, needs a physics loss and a set of boundary conditions to train the network with a physical model. This even becomes more important for the CFD PINN since one of the to-be-developed models will not make use of experimental data during the training process, meaning that the network will solely be trained on the physics loss and boundary conditions within the computational domain. In Code 13.1, the implementation of the physics loss function for the CFD PINN is presented. The physics loss makes use of reduced Navier-Stokes equations, as explained in Equation 11.2. Furthermore, 5000 domain points, being drawn at random, are being used to evaluate the physical model of Equation 11.2. In Code 13.2, the implementation of the various boundary losses is presented. For the input boundary conditions, a Dirichlet boundary condition is used, enforcing the u-component of the velocity to be the far field velocity, the v-component of the velocity to be zero and the inlet pressure to be equal to the total pressure. The outlet boundary conditions, on the other hand, are Neumann boundary conditions, enforcing the derivative of the u- and v-components of the velocity and pressure to be equal to zero. The upper and lower wall boundary conditions are set to Dirichlet boundary conditions, enforcing the u-component of the velocity to be equal to the free stream velocity, while the v-component is set to zero. Lastly, the object's boundary condition is being implemented, which is set to a non-slip boundary condition, meaning that the u- and v-components of the velocity are set to zero at the object's boundary. The input, outlet and wall boundary conditions make use of 200 collocation points, while the boundary points at the object are set to 600. The individual weights of the physics loss, as well as the various boundary losses, are set to 1, meaning that all losses are equally important during the training process.

Code 13.1: Implementation of the physics loss for the CFD simulation

```
def Navier_stokes_loss(NN_output: torch.nn.Module, used_inputs):
    # Obtain predictions from the network within the domain
    u, v, p = NN_output(points_domain).split(1, dim=1)

    # Obtain u_x, u_xx, u_y, u_yy, v_x, v_xx, v_y, v_yy, p_x, p_y
    du_dx = gradient(u, points_domain)[:, 0]
    du_dxx = gradient(du_dx, points_domain)[:, 0]
    du_dy = gradient(u, points_domain)[:, 1]
    du_dyy = gradient(du_dy, points_domain)[:, 1]
    dv_dx = gradient(v, points_domain)[:, 0]
    dv_dxx = gradient(dv_dx, points_domain)[:, 0]
    dv_dy = gradient(v, points_domain)[:, 1]
    dv_dyy = gradient(dv_dy, points_domain)[:, 1]
    dp_dx = gradient(p, points_domain)[:, 0]
    dp_dy = gradient(p, points_domain)[:, 1]

    # Continuity equation
    CE = du_dx + dv_dy

    # Momentum equations (in x and y directions)
    Mx = u * du_dx + v * du_dy + 1/rho * dp_dx - nu * (du_dxx + du_dyy)
    My = u * dv_dx + v * dv_dy + 1/rho * dp_dy - nu * (dv_dxx + dv_dyy)

    return ((CE**2).mean() + (Mx**2).mean() + (My**2).mean())
```

Code 13.2: Implementation of the boundary loss for the CFD simulation

```
# Define inlet boundary loss
def BC_inlet_loss(NN_output: torch.nn.Module, used_inputs):
    # Obtain predictions from the network within the domain
    u, v, p = NN_output(points_input).split(1, dim=1)

    return (((u-u_inf)**2).mean() + (v**2).mean() + ((p-p_tot)**2).mean())

# Define outlet boundary loss
def BC_outlet_loss(NN_output: torch.nn.Module, used_inputs):
    # Obtain predictions from the network within the domain
    u, v, p = NN_output(points_exit).split(1, dim=1)

    # Obtain u_x and v_x
    du_dx = gradient(u, points_exit)[:, 0]
    dv_dx = gradient(v, points_exit)[:, 0]
    dp_dx = gradient(p, points_exit)[:, 0]

    return ((du_dx**2).mean() + (dv_dx**2).mean() + (dp_dx**2).mean())

# Define upper wall boundary loss
def BC_upper_wall_loss(NN_output: torch.nn.Module, used_inputs):
    # Obtain predictions from the network within the domain
    u, v, p = NN_output(points_down).split(1, dim=1)

    return (((u-u_inf)**2).mean() + (v**2).mean())

# Define lower wall boundary loss
def BC_lower_wall_loss(NN_output: torch.nn.Module, used_inputs):
    # Obtain predictions from the network within the domain
    u, v, p = NN_output(points_up).split(1, dim=1)

    return (((u-u_inf)**2).mean() + (v**2).mean())


# Define object boundary loss
def BC_object_loss(NN_output: torch.nn.Module, used_inputs):
    # Obtain predictions from the network within the domain
    u, v, p = NN_output(points_object).split(1, dim=1)

    return ((u**2).mean() + (v**2).mean())
```

After the various implementations of the various loss terms have been presented, the overall structure of the implementation of the CFD PINN model is presented in Figure 13.2.

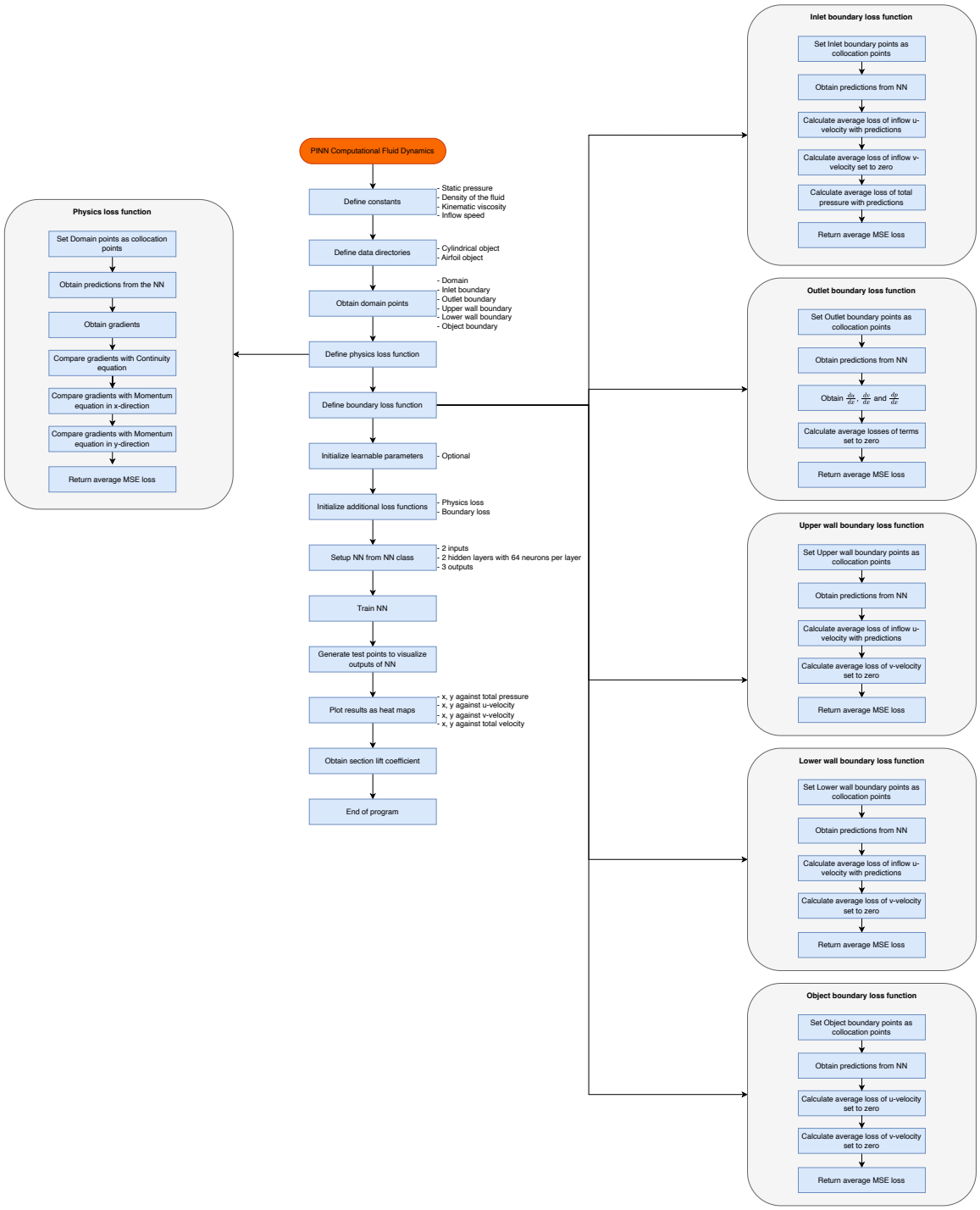


Figure 13.2: Flowchart presenting the PINN program for the Computational Fluid Dynamics simulation Python code

The equations used in the physical loss, as presented in Equation 11.2 and the various boundary conditions, make use of some constants. Although not being used as inputs for the PINN, these constants

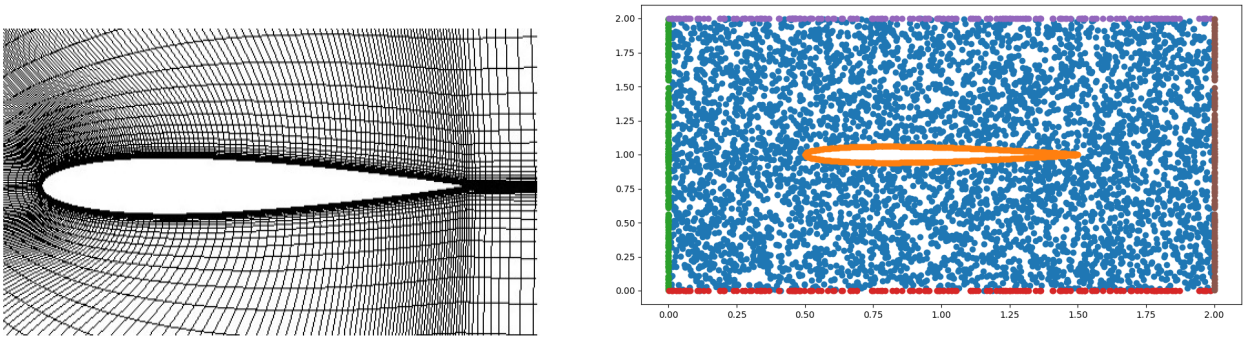
are required for the simulation. These constants and their units are summarised in Table 13.1.

Table 13.1: Summary of constants used for the PINN CFD simulation

Symbol	Meaning	Unit
P	Static pressure in the domain	[bar]
ρ	Density of the fluid inside the domain	[kg/m^3]
U_∞	Velocity of fluid upstream and downstream of the domain in x-direction	[m/s]
x_{domain}	Size of the length of the domain	[-]
y_{domain}	Size of the height of the domain	[-]

13.2. Setup of the Computational Domain

In this chapter, a PINN is being developed to compare traditional numerical methods with PINNs for CFD simulations. As explained in section 11.6, the greatest advantage of using a PINN for CFD simulations is the fact that PINNs make use of a meshless approach to compute derivatives in equations. By doing so, theoretically, the accuracy of this method is dependent on the accuracy of the computer, rather than the accuracy of the chosen discretisation scheme. This results in the fact that a computational mesh no longer needs to be generated, potentially enabling fine-tuning and adapting the computational domain of interest with more ease compared to traditional computational mesh generation methods. In Figure 13.3a, a typical domain for a traditional numerical CFD simulation is presented, making use of a computational mesh. In Figure 13.3b, on the other hand, a similar type of domain is presented, but instead of using a mesh, this domain makes use of collocation points, which are used by the PINN to perform the calculations within the computational domain.



(a) Traditional Structured Computational Fluid Dynamics Simulation mesh [49]

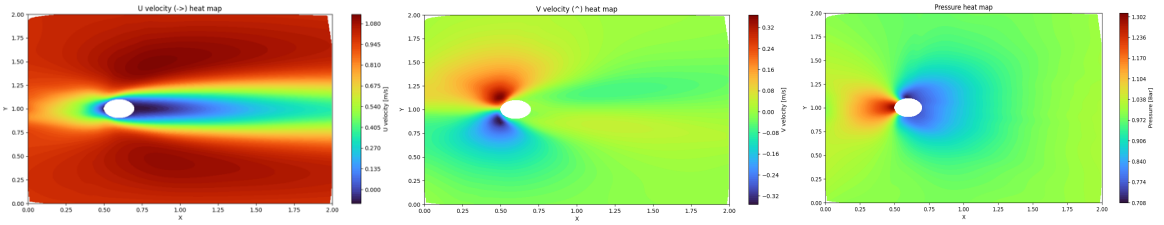
(b) Computational Fluid Dynamics Simulation domain setup as used for PINNs

Figure 13.3: Difference between traditional mesh for a Computational Fluid Dynamics Simulation compared to the meshless approach used in PINNs

From Figure 13.3, it can be seen that the structured mesh for the traditional numerical CFD simulation, which is presented in Figure 13.3a, has a lot of cells which are decreasing in size when nearing the boundary of the object. The domain for the PINN, on the other hand, which is presented in Figure 13.3b, has collocation points randomly drawn within the computational domain. It can be seen that these points are not clustered towards the object's boundary layer compared to the decreasing cell sizes for the structured mesh. Due to the randomly drawn samples in the domain, the object can be freely positioned within the domain, enabling easier adjustments of the position and rotation of the object, while for the structured mesh, a certain procedure takes more pre-processing steps.

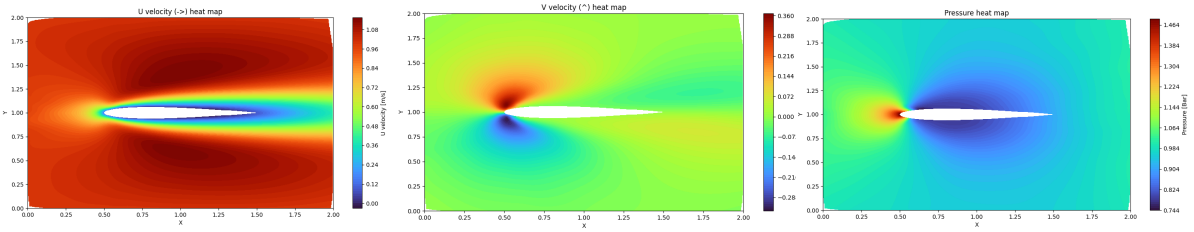
13.3. Results of the Model

The developed PINN model will provide predictions of the u-component of the velocity, the v-component of the velocity, as well as the pressure, as was explained in section 13.1. In Figure 13.4, an example of this prediction for these quantities for a 2D cylinder is presented. In Figure 13.5, the same predictions for these quantities are being presented, but now for a 2D airfoil section.



(a) Simulation of u-velocity in the domain **(b)** Simulation of v-velocity in the domain **(c)** Simulation of pressure in the domain

Figure 13.4: Example output of the PINN developed for Computational Fluid Dynamics simulations, for u-velocity, v-velocity and pressure in the provided domain using a 2D cylinder as an object



(a) Simulation of u-velocity in the domain **(b)** Simulation of v-velocity in the domain **(c)** Simulation of pressure in the domain

Figure 13.5: Example output of the PINN developed for Computational Fluid Dynamics simulations, for u-velocity, v-velocity and pressure in the provided domain using a NACA 0012 airfoil as an object

14

Model Comparison: Computational Fluid Dynamics

After the PINN model has been developed, tests can be performed to be able to answer the research questions as presented in chapter 10. In this chapter, various tests will be performed to assess the performance of the various modelling methods and compare them with the PINN. In section 14.1, the setup of the various tests will be explained. Next, a traditional numerical CFD method will be compared to a PINN in section 14.2, comparing the predictions in terms of velocity and pressure profiles. Lastly, a comparison will be made of the prediction capability of the section lift coefficient in section 14.3, using the experimental data as presented in chapter 12.

14.1. Comparison Model

To be able to answer the research questions, two tests will be conducted to investigate the performance of the to-be-tested models. The models which will be compared are a traditional numerical model, a PINN without experimental data, which only uses the Navier-Stokes equations as presented in Equation 11.2 and a PINN with experimental data, which uses the pressure coefficient distribution obtained from the experiments as presented in chapter 12. Two models will be used for the traditional numerical method. The first test will make use of the open-source CFD solver Open-source Field Operation And Manipulation, shortened OpenFOAM [92]. The second test will make use of Computational Fluids Laboratory 3-Dimensional, shortened CFL3D, which is a Reynolds-averaged Navier-Stokes (RANS) solver developed by NASA [28]. It should be noted that CFL3D is also able to perform 2D simulations [28]

14.2. Velocity and Pressure Profile Comparison

The first test to assess the performance of the PINN compared to a traditional Numerical method for CFD simulations will be to compare the velocity and pressure prediction profiles of the two methods. To do so, both methods will be provided with equal input conditions, whereafter the profiles will be compared in terms of maximum and minimum magnitudes for velocity and pressure, as well as their locations. The object used during this test is a 2D cylinder with a diameter of 0.1 [m]. The flow conditions are set to incompressible flow, making use of a steady-state simulation type. The flow conditions are summarised in Table 14.1.

Table 14.1: Flow condition for the first test to compare a numerical method to the PINN for a CFD simulation [12]

Quantity	Value	Unit
Inflow velocity (U_∞)	1	[m/s]
Kinematic viscosity of the flow (ν)	0.02	[m^2/s]
Freestream pressure (p_∞)	0.2	[pa]
Reynolds number (Re)	5	[-]

The traditional numerical method makes use of the OpenFOAM solver. The incompressible Navier-Stokes equations as presented in Equation 11.2 are being solved, making use of the Semi-Implicit Method for Pressure Linked Equations algorithm, or shortened simpleFoam, to update the velocity and pressure iteratively [12]. The spatial discretisation is performed using second-order upwind schemes and FVM [12]. The computational domain for the OpenFOAM solver, including the boundary conditions for each wall, is presented in Figure 14.1.

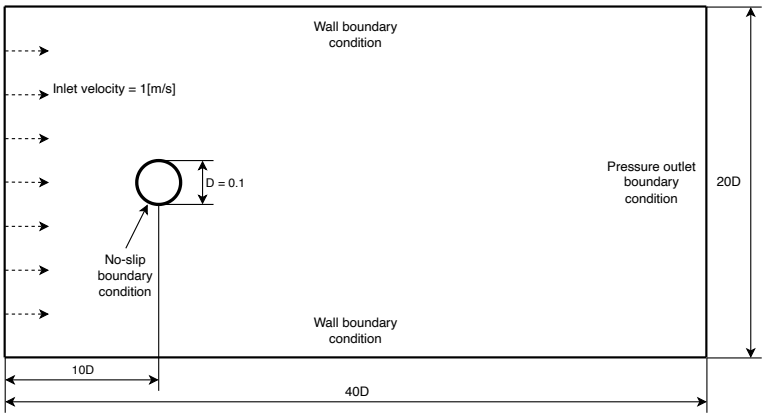


Figure 14.1: Computational domain for the OpenFOAM simulation [12]

From Figure 14.1, it can be seen that the inlet boundary condition is set to a Dirichlet boundary condition, with an inflow velocity of 1 [m/s] in the horizontal direction. The cylinder wall has a no-slip boundary condition, which means that the velocity in the u- and v-directions is set to zero. The top and bottom walls also have Dirichlet boundary conditions, enforcing the u-velocity component to be equal to the free stream velocity, while the v-velocity component is set to zero. The outlet boundary is set to a pressure outlet, enforcing the outgoing pressure to be equal to the free stream pressure.

To be able to compare the velocity and pressure contours and magnitudes, the PINN should have the same flow conditions as the ones presented in Table 14.1. The computational domain for the PINN is presented in Figure 14.2. The diameter of the cylinder is set to 0.1 [m], and the domain has a length of 4 [m] and a height of 2 [m]. To perform the simulation, the PINN uses 20000 epochs, with a learning rate of $1 \cdot 10^{-2}$.

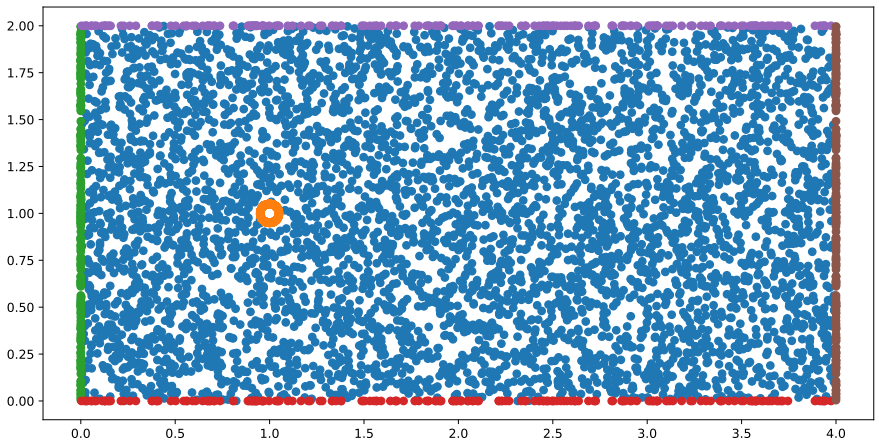
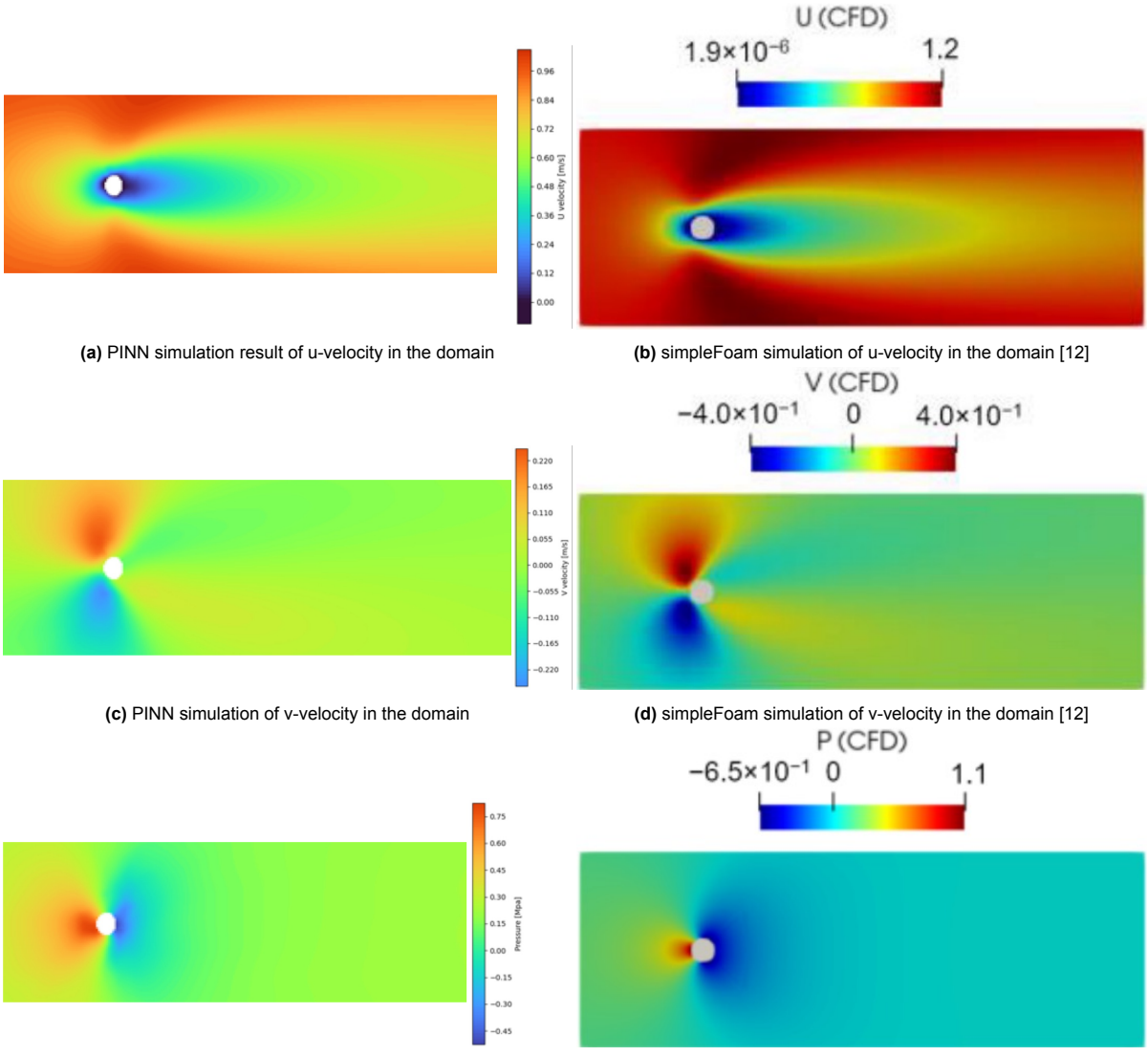


Figure 14.2: Computational domain for the PINN CFD simulation comparing the pressure and velocity contours and magnitudes of a 2D cylinder

After some simulations were performed using the PINN, it was found that the boundary conditions for the in and outlet needed to be changed to have a faster convergence time and more reliable results. The inlet conditions needed two additional Dirichlet boundary conditions, namely to enforce the v-component of the velocity to be zero, as well as to enforce the inlet pressure of the domain to be equal to the total pressure. The outlet boundary of the domain needed 3 Neumann conditions, enforcing the derivative of the u- and v-components of the velocity to be zero, as well as the static pressure. The results of the CFD simulations making use of the PINN and SimpleFOAM are presented in Figure 14.3, and the maximum and minimum of each method are presented in Table 14.2. It should be noted that the colour of the heatmaps showing the PINN's result was corrected to match the scale of the simpleFoam simulation, and that the size of the domain is zoomed in to a 2.25 [m] length and 0.75 [m] height.



(a) PINN simulation result of u-velocity in the domain

(b) simpleFoam simulation of u-velocity in the domain [12]

(c) PINN simulation of v-velocity in the domain

(d) simpleFoam simulation of v-velocity in the domain [12]

(e) PINN simulation of pressure in the domain

(f) SimpleFoam simulation of pressure in the domain [12]

Figure 14.3: Comparison of output predictions of the CFD PINN developed for CFD simulations and the traditional CFD solver using simpleFoam [12], for u-velocity, v-velocity and pressure in the provided computational domain using a 2D cylinder as an object

Making use of Figure 14.3, the contour plots for the predictions of the u- and v-velocity components, as well as the pressure, can be compared between the PINN and the simpleFoam simulation. Firstly, the contour plots will be compared in terms of the general shape of the provided prediction. Comparing the prediction for the u-velocity component for the PINN, presented in Figure 14.3a against the simpleFoam predictions presented in Figure 14.3b, it can be seen that there is a good agreement between the two.

On the top and the bottom of the cylinder in the simpleFoam simulation presented in Figure 14.3b, two small indents are present with a high u-velocity region. These two regions are also present in the PINN simulation as presented in Figure 14.3a; however, not as clearly as the simpleFoam predictions as presented in Figure 14.3b. Furthermore, the low-speed regions on the front and the back of the cylinder in Figure 14.3b, presented using the blue colour, are also present at the same location as presented in the prediction of the PINN, as presented in Figure 14.3a. However, these regions are not as sharp as compared to the simpleFoam simulation presented in Figure 14.3b. This also holds for the yellow boundary at the outer edge of the slow velocity regions of the simpleFoam simulation, presented in Figure 14.3b. The predictions of the PINN do not have this clear slower velocity boundary. Instead, the velocity change happens more gradually, as can be seen from Figure 14.3a.

Continuing to the v-velocity component, of which the predictions for the simpelFoam simulation are being presented in Figure 14.3d and the prediction for the PINN presented in Figure 14.3c, it can be seen that 4 regions are present around the cylinder. These regions can be presented as 2 upward velocity regions and 2 downward velocity regions. Using the predictions from Figure 14.3d, it can be seen that at the front of the cylinder, an upward velocity region is present at the top of the cylinder, and a downward velocity region at the bottom. On the back of the cylinder, these two regions are also present but in reverse order, meaning that a downward velocity region is present at the top of the cylinder, while an upward velocity region is present at the bottom of the cylinder. However, the two regions at the back of the cylinder are smaller in magnitude compared to the two velocity regions at the front of the cylinder. Now, comparing Figure 14.3d to the predictions of the PINN in Figure 14.3c, it can be seen that these 4 regions and their locations are also present. Furthermore, the two lower magnitudes of the velocity regions compared to the front of the cylinder, as presented in Figure 14.3d, are also correctly being predicted by the PINN, as can be seen from Figure 14.3c. Furthermore, the two higher in magnitude regions on the front of the cylinder, when getting closer towards the boundary of the cylinder, as presented in Figure 14.3d, are also correctly being predicted by the PINN, as can be seen in Figure 14.3c. However, as with the u-velocity component, the sharp boundary transitions between the various velocity magnitudes, as presented by the yellow and light blue colours in Figure 14.3d, are not correctly being predicted by the PINN. Instead, from Figure 14.3d, it can be seen that the PINN has a more gradual change in v-velocity regimes.

Lastly, the pressure shape predictions will be compared. From Figure 14.3f, it can be seen that a high-pressure component is present at the front of the cylinder and a low-pressure region at the back of the cylinder. The high-pressure region presented in Figure 14.3f has a sharp boundary transition at the front, while the low-pressure region on the back has a more gradual transition. Comparing these results to the predictions of the PINN, as presented in Figure 14.3e, it can be seen that both the high and low pressure regions are correctly being predicted, as well as their locations. However, likewise with the predictions for the u- and v-velocity components, the PINN is not able to predict the sharp boundary transition of the high-pressure region at the front of the cylinder as present in Figure 14.3f. Furthermore, the predictions of the PINN as presented in Figure 14.3e show a much larger value compared to the high-pressure region from the simpleFaoam simulation as presented in Figure 14.3f. On the other hand, the PINN can predict the gradual change in the low-pressure region as presented in Figure 14.3f, but the region is smaller in height compared to the simpleFoam simulation presented in Figure 14.3f. Furthermore, the locations of the minimum pressure at the boundary of the airfoil are not correctly positioned, as it can be seen from Figure 14.3f by the dark blue color that the minimum pressure is presented all across the cylinder, while the PINN predictions are only at a small section at the back of the cylinder, as presented in Figure 14.3e.

After the contour plots have been compared on their general shape, the magnitudes presented by the plots will be compared. To do so, the magnitudes of the maximum and minimum predictions for each quantity for the PINN and simpleFoam simulations are summarised in Table 14.2.

Table 14.2: Maximum and Minimum values for the traditional CFD solver using simpleFoam and the PINN, for the u-velocity, v-velocity and pressure

Quantity	simpleFoam	PINN
Maximum u-Velocity [m/s]	1.2	1.05
Minimum u-Velocity [m/s]	$1.9 \cdot 10^{-6}$	-0.09
Maximum v-Velocity [m/s]	-0.4	-0.255
Minimum v-Velocity [m/s]	0.4	0.245
Maximum pressure [Mpa]	1.1	0.825
Minimum pressure [Mpa]	-0.65	-0.525

Making use of Table 14.2, it can be seen that for the u-velocity component, the maximum prediction between the PINN and the simpleFoam simulation is relatively small. The PINN has an underprediction of the u-velocity component of about 12.5%. The minimum velocity, on the other hand, shows that there are some discrepancies between the simpleFoam and PINN predictions. The PINN, namely, predicts that there will be a negative u-velocity component, which means that there will be a location with flow reversal. However, the simpleFoam simulation does not show this behaviour. Comparing the magnitudes of the v-velocity components, it can be seen that the minimum and maximum magnitudes have a similar difference tween the PINN and the simpleFoam simulation, about 38.8% for the maximum and about 36.3% for the minimum magnitudes of the v-velocity components. The PINN does, however, under-predict the maximum v-velocity component and the minimum v-velocity component. However, the signs of the magnitudes are correct, meaning that the correct directionality of the flow is being captured by the PINN for the v-velocity component compared to the simpleFoam simulation. Lastly, comparing the magnitudes for the pressure, it can be seen that the signs of the magnitudes for the minimum and maximum pressure regions are correctly being predicted by the PINN compared to the simpleFoam simulation. The minimum pressure has a difference of about 19.2% compared to the simpleFoam simulation, and the maximum pressure prediction has a difference of about 25% compared to the simpleFoam simulation.

In terms of computational efficiency, the computational time of the PINN is much higher compared to the simpleFoam simulation. Where the OpenFOAM simulation takes 170 [s] to run [12], the same PINN simulation takes about 816 [s]. This means that the simpleFoam simulation is much faster in terms of computational time compared to the PINN. However, where the OpenFOAM solver required quite some time to set up the mesh and refine the mesh [12], the setup of the computational domain of the PINN was much faster and less time-consuming to change. Not only can the ”mesh” be refined more easily, but the domain and object can also be easily changed, repositioned or reoriented. Besides this, experimental data could be incorporated, which could lead to more insights when validated against experimental data.

Concluding this comparison, it can be seen that the PINN is not able to provide close predictions compared to a traditional numerical CFD simulation method when comparing contour shapes and magnitudes, discarding the use of experimental data. Although the velocity contour shapes are mostly correctly predicted by the PINN, fast transition layers are not being correctly predicted by the PINN. The contour prediction shape of the PINN for the pressure is the least accurate, showing a too large high-pressure region and a too small low-pressure region. The most likely reason for this behaviour of the pressure is because the pressure is not explicitly being implemented in the physics loss. Instead, the pressure is embedded in the momentum equations, enabling the fact that PINN is not able to capture the pressure trend. To solve this, an additional equation could be implemented in the physics loss, as was recommended by the literature [12]. In terms of magnitudes, on average, the PINN underpredicts the maximum values by about 25.4%, while for the minimum predictions, the PINN sometimes makes prediction mistakes in terms of signs. Furthermore, the computational time of a PINN is also higher compared to a traditional numerical CFD solver, 170 [s] compared to 816 [s]. This shows that the PINN does not provide an improvement in computational efficiency.

After performing some additional tests, it was found that changing the domain length to 2 [m] instead of the previously mentioned 4 [m] provided better results while leaving all the other parameters at the same values. The most likely explanation for this is that there are more collocation points present around the

object, leading to better predictions. The new computational domain is presented in Figure 14.4.

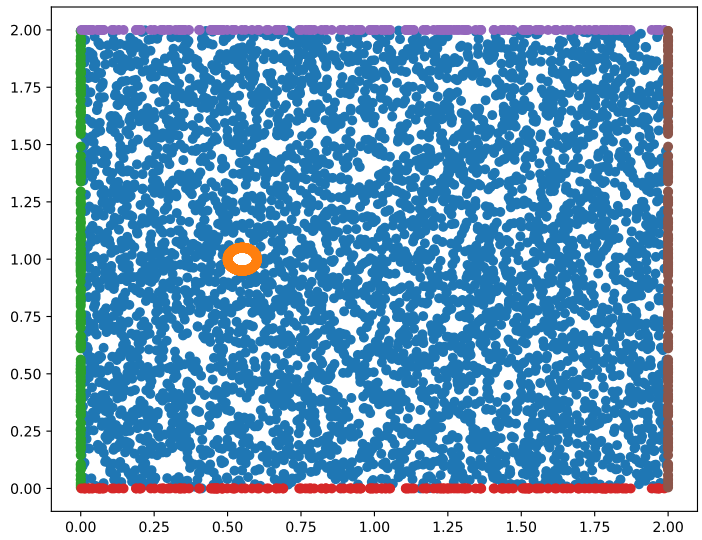
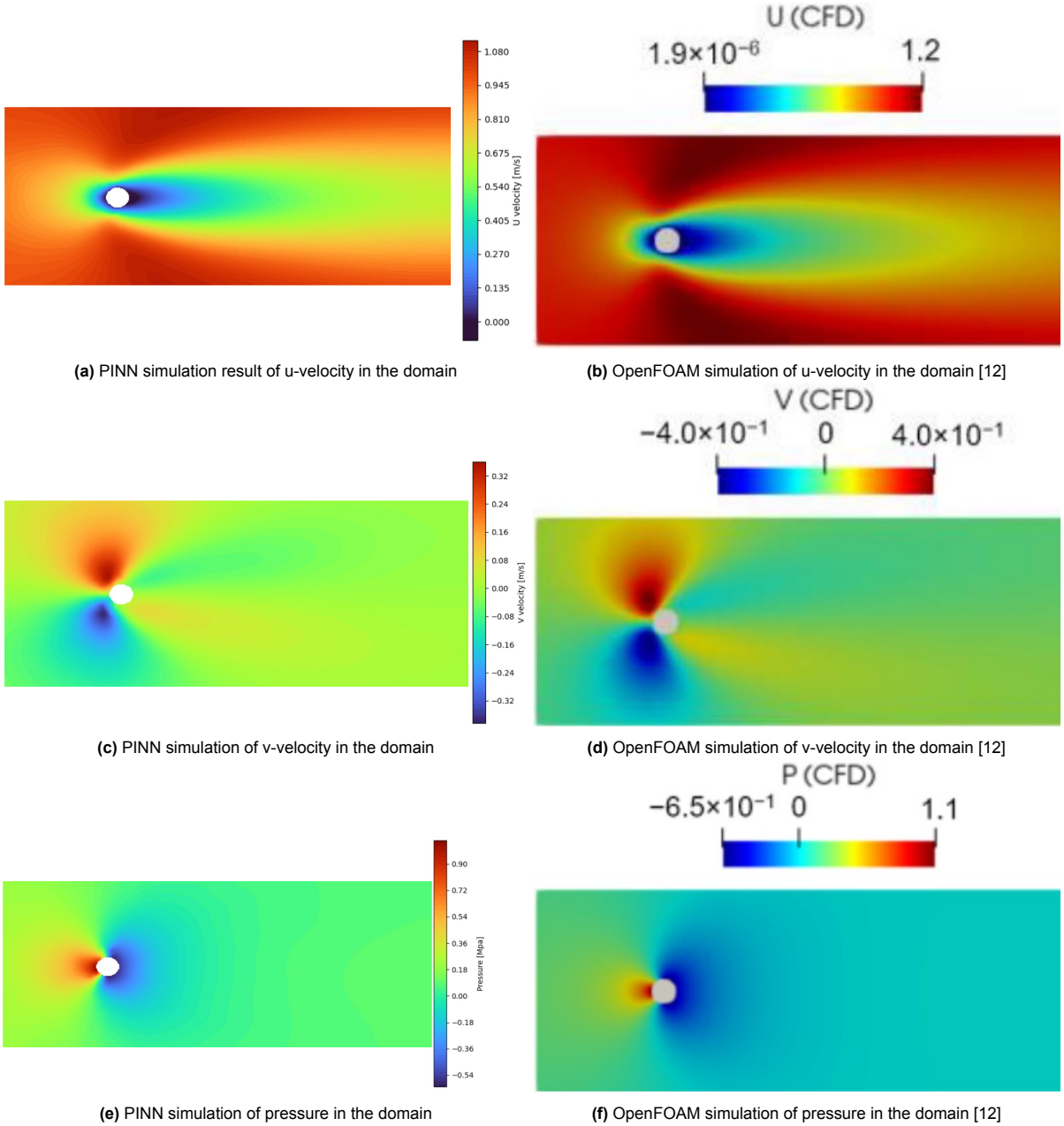


Figure 14.4: Smaller computational domain for the PINN CFD simulation comparing the pressure and velocity contours and magnitudes of a 2D cylinder

Making use of this new smaller computational domain for the PINN, as presented in Figure 14.4, the comparison with the smaller domain is presented in Figure 14.5. The corresponding maximum and minimum magnitudes of the predictions of the u- and v-velocity components and the pressure are presented in Table 14.3. As with the other comparison, the PINN contour plot colours are corrected compared to the simpleFoam simulation, and the presented domain is scaled to a length of 1.8 [m] and 0.75 [m] in height.



(a) PINN simulation result of u-velocity in the domain

(b) OpenFOAM simulation of u-velocity in the domain [12]

(c) PINN simulation of v-velocity in the domain

(d) OpenFOAM simulation of v-velocity in the domain [12]

(e) PINN simulation of pressure in the domain

(f) OpenFOAM simulation of pressure in the domain [12]

Figure 14.5: Comparison of prediction outputs of the CFD PINN making developed for CFD simulations making use of a smaller computational domain and the traditional CFD solver OpenFOAM [12], for u-velocity, v-velocity and pressure a 2D cylinder as an object

From Figure 14.5, it can be seen that for the predictions of the PINN, there is now a better agreement in terms of rapid boundary condition changes compared to the larger computational domain. Furthermore, the upwards and downwards velocity region predictions for the PINN, as presented in Figure 14.5c, do have a better agreement now with the simpleFoam prediction as presented in Figure 14.5d. Furthermore, the low-pressure regions on the back of the cylinder are now also better predicted by the PINN compared to the simpleFoam simulation, as can be seen from Figure 14.5f for the PINN and Figure 14.5e for the simpleFoam simulation. From these figures, it can also be seen that the high-pressure region at the front of the airfoil is now more refined compared to the previous prediction, as was presented in Figure 14.3f.

After the contour plot shapes of the smaller domain have been compared, the magnitudes of the minimum and maximum predictions for the u- and v-velocity components and the pressure predictions can

be compared. To do so, the predictions of the PINN for this smaller domain and the predictions of the simpleFoam simulation using the old domain are summarised in Table 14.3.

Table 14.3: Maximum and Minimum values for the traditional CFD solver using simpleFoam and the PINN, for the u-velocity, v-velocity and pressure

Quantity	simpleFoam	PINN
Maximum u-Velocity [m/s]	1.2	1.125
Minimum u-Velocity [m/s]	$1.9 \cdot 10^{-6}$	-0.075
Maximum v-Velocity [m/s]	-0.4	-0.384
Minimum v-Velocity [m/s]	0.4	0.36
Maximum pressure [Mpa]	1.1	1.06
Minimum pressure [Mpa]	-0.65	-0.62

From Table 14.3, it can be seen that the predicted values of the PINN are now closer to the predictions from the simpleFoam simulation. Comparing the u-component for the velocity, the difference of the PINN with the simpleFoam simulation is about 6.3% for the maximum value. The minimum value still shows a negative value, meaning flow reversal. The v-velocity component shows a similar improvement, having a predicted difference between the PINN and simpleFoam simulation of 4% for the prediction of the minimum value and 10% for the maximum value. The pressure value predictions between the PINN and simpleFoam simulation also shows an improvement, having a difference of about 3.6% for the prediction of the maximum value and 4.6% for the minimum value. The computational time for this simulation is about 840 [s], meaning that it is similar compared to the previous domain, which took about 816 [s]. However, compared to the simpleFoam solver, which uses about 170 [s], the PINN still takes a much larger time to be able to predict the velocity components and the pressure.

To conclude, using a smaller domain while keeping the other quantities, such as flow quantities, fixed, results in an improvement in the prediction capabilities of the PINN. Not only do the predictions for the contour shapes for the velocity components and pressure, but also the numerical values in terms of minimum and maximum prediction values improve. This shows that, compared to previously shown neural networks having more hidden layers and neurons per layer in literature [12], it is possible to obtain satisfactory results with a smaller network architecture using PINNs. However, the computational efficiency of the PINN remains lower compared to a traditional numerical CFD solver, having a computational time of about 840 [s] for the PINN, while the simpleFoam simulation has a computational time of about 170 [s]. However, it is doubtful if the provided boundary conditions are acceptable given the current domain size. A potential reason why the simulation seems to provide more accurate results and does not seem to have issues with the boundary conditions seems to rely in the fact how the boundary conditions are enforced. In an NN, the boundary conditions are not enforced as hard constraints, rather, they are tried to be enforced as a sort of guideline. Therefore, it could be that certain boundary conditions are not being fully adhered to, which positively influenced the outcome of the result as previously presented when decreasing the domain size.

After switching the objects geometry, changing the object for the NACA 0012 airfoil instead of the cylinder with a 0.1 [m] diameter, an interesting observation was made keeping the domain, boundary conditions and problem setup the same. It was found that the computational time for the airfoil was about 730 [s] for the PINN, while an OpenFOAM simulation took about 1687 [s] [12]. Changing the airfoil angle of attack only slightly increased the PINNs training time to about 805 [s] for 5 and 15 [$^\circ$] angles of attack, respectively. This shows that, although the computational time for PINNs is larger for simple problems, there is a possibility that the computational time will remain of a similar order of magnitude when increasing problem difficulty. This means that the PINN's computational time shows signs of being independent of the problem setup, while this is not the case for a traditional numerical CFD solver. However, the obtained accuracy has not been shown to be sufficient to compete with traditional numerical CFD solving methods. The fact that it was previously shown that a greater accuracy can be obtained by making use of a smaller domain is an interesting finding, but it is hard to justify if this is an acceptable method. However, if the training time required to train the PINN is reduced, a PINN could potentially become an interesting alternative or extension to the CFD simulation tool cases, decreasing pre-processing steps and simplifying of setup of the simulation.

14.3. Section Lift Coefficient Prediction Comparison

After the first test was conducted, as presented in section 14.2, it was concluded that, when using a similar problem setup in terms of computational domain and flow characteristics, a PINN without experimental data is not able to obtain similar order of accuracy compared to a traditional numerical CFD method. However, when changing the domain size, the results are satisfactorily accurate, only showing some small differences with the traditional numerical CFD method. Furthermore, it was shown that, while for simple problems such as the flow over a cylinder the computational time is much larger compared to a traditional numerical CFD method, the computational time for a PINN remains relatively similar when changing the problem to a more difficult setup such as the flow around a NACA 0012 airfoil compared to the major increase in computational time for the traditional numerical CFD method. Currently, there is no direct comparison between a PINN and a traditional numerical CFD solver when a PINN uses experimental data. This could be a potentially interesting tool which can refine CFD simulations by directly incorporating experimental measurements to improve the simulations.

In this section, the performance of a PINN, making use of experimental data, will be compared against a traditional numerical CFD method. For this comparison, the section lift coefficient will be predicted and compared to the section lift coefficient of the NACA 0012 airfoil as measured in section 12.3. To do so, the experimental data will be compared against a traditional numerical method, which makes use of a CFL3D simulation, a PINN without experimental data and a PINN with experimental data. Furthermore, the section lift coefficient will also be compared against the experimental data used in the PINN to identify any advantages. The comparison will be performed for 3 angles of attack, namely $0.0169[^\circ]$, $10.0254[^\circ]$ and $15.0299[^\circ]$, due to the availability of experimental data sets. The flow conditions used during this experiment are summarised in Table 14.4.

Table 14.4: Flow condition for the second test to compare the prediction of the section lift coefficient of a NACA 0012 airfoil making use of a traditional numerical CFD solver and a PINN with and with experimental data [61]

Method	Value	Unit
Inflow velocity (U_∞)	104.1	[m/s]
Flow density (ρ)	1.1325	[kg/m^3]
Kinematic viscosity of the flow (ν)	$1.735 \cdot 10^{-5}$	[m^2/s]
Freestream pressure (p_∞)	1	[Bar]
Reynolds number (Re)	$6 \cdot 10^6$	[-]
Mach number (M)	0.3	[-]

The computational domain of the PINN, making use of the NACA 0012 airfoil as object for various angles of attack is presented in Figure 14.6. The simulation of the PINN makes use of 20000 epochs and a learning rate of $1 \cdot 10^{-2}$.

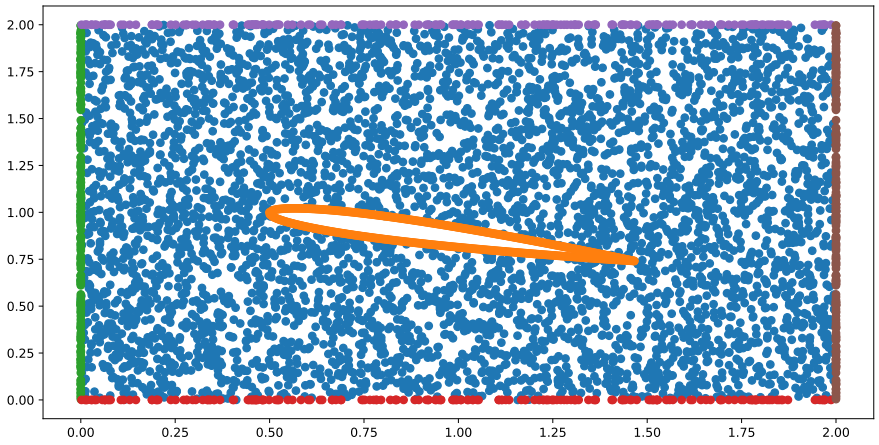


Figure 14.6: Computational domain of the PINN, using the NACA 0012 airfoil as object. The airfoil shown has an angle of attack of 15.0299 [°]

To be able to incorporate the experimental data into the PINN simulation, the experimental data obtained from chapter 12 needs to be correctly processed to convert it back into a pressure, whereafter the location of the pressure value needs to correspond to the pressure tab location present in the physical experimental setup. This means that the individual measured pressure points need to be positioned on the boundary of the airfoil, of which an example is presented in Figure 14.7.

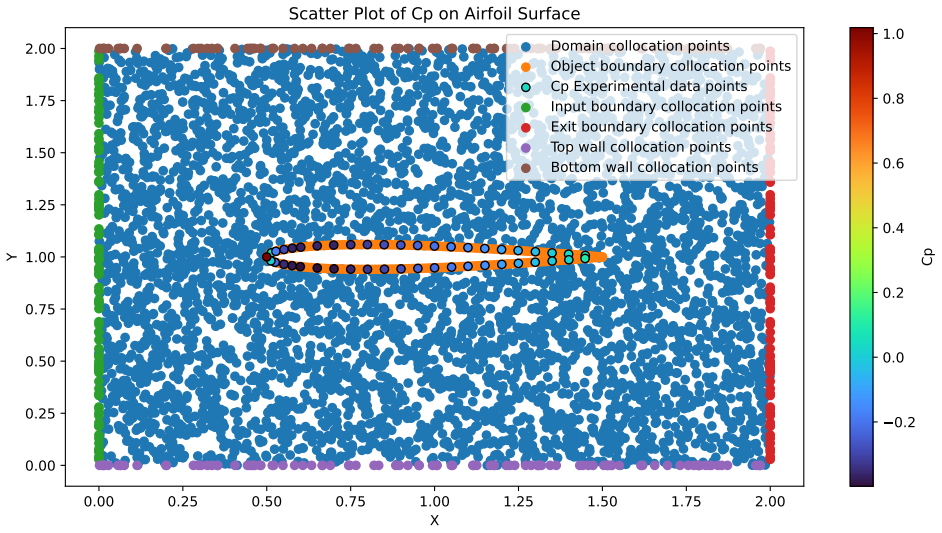


Figure 14.7: Computational domain of the PINN including experimental Cp data, using the NACA 0012 airfoil as object. The airfoil shown has an angle of attack of 0.0169 [°]

The comparison of the predictions of the section lift coefficient for the PINN, PINN including experimental data and traditional Numerical method is presented in Figure 14.8.

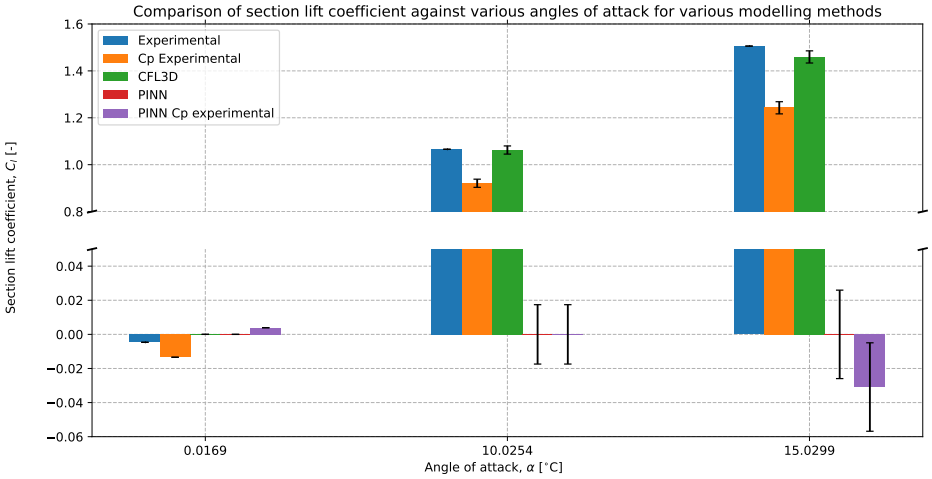


Figure 14.8: Bar chart presenting the predictions of the various modelling methods against experimental data of the section lift coefficient of a NACA 0012 airfoil for various angles of attack, with error margins introduced due to the simplification of the section lift coefficient calculation

From Figure 14.8, it can be seen that the traditional Numerical method has a good agreement with the section lift coefficient found when performing wind-tunnel experiments. When comparing the section lift coefficient, by solely making use of the pressure coefficient experimental data, it can be seen that this is less accurate compared to a traditional Numerical method. The most likely reason for this is that certain regions of the flow at the boundary of the airfoil are present in the Numerical simulation, but are not measured by the experimental data, since measurements have not been performed at these locations. Therefore, it can be concluded that an additional method, such as a PINN, would be required to obtain the missing information. When comparing the PINN and PINN including experimental data, it can be seen that both methods are not able to compete with the traditional Numerical method. The most likely reason for this is that the PINN has not fully converged yet, or the currently shallow NN is not sufficient to capture the trends for high Mach number and High Reynolds number flows. A detailed overview of the predicted section lift coefficient for each method is presented in Table 14.5.

Table 14.5: Comparison of section lift coefficient predictions from different methods at various angles of attack

Angle of Attack [°]	Experimental	C_p Exp.	CFL3D	PINN	PINN (C_p)
0.0169	-0.00466432	-0.013411649555689166	$8.186455625482092 \cdot 10^{-6}$	0	0.003904
10.0254	1.06567681	0.9207829401464712	1.0625313059560701	-0.000004	0
15.0299	1.50574158	1.242599369403683	1.4595262773949997	0	-0.030883

To conclude, currently, using a shallow NN, it has not been shown that a PINN can compete against a traditional Numerical model when validated against experimental data of the section lift coefficient. Furthermore, while introducing experimental data to the PINN opens up possibilities which are currently not possible with traditional Numerical methods, it has not been shown that this method is currently able to compete against a traditional Numerical method. However, it should be noted that only a shallow NN was used, making use of non-dedicated hardware for NN training.

Part III

Solid Rocket Motor Modelling

15

Introduction: Solid Rocket Motor

So far, this report has shown two different applications of PINNs. In Part I, the performance of a PINN was compared against a purely Data-Driven method and a Numerical method in terms of accuracy, data efficiency and computational efficiency for the prediction of a decaying temperature profile. Furthermore, it was investigated whether a PINN could be used as a surrogate model for the same purpose. In Part II, another aspect of PINNs was investigated, namely obtaining fluid flow predictions for velocity and pressure without experimental data, by training the PINNs network parameters solely on differential equations describing a physical model. After this, experimental data was added to compare the performance of the PINN with and without experimental data against unseen experimental data and a traditional numerical method. Part I and Part II showed that PINNs have a comparable performance compared to purely Data-Driven and Numerical methods, obtaining a higher data-efficiency and sometimes accuracy. However, the computational efficiency of PINNs compared to the other methods is worse.

The problems presented in the previous parts of the report can be considered to be relatively simple, straightforward problems, suitable to test the PINNs' performance. For example, Part I makes use of a single differential equation, for which an analytical solution exists. Part II makes use of well-known and studied examples, mostly used to optimise and compare traditional numerical models. However, although the previous parts made use of relatively simple problems, this does not mean that they are not useful. Using these simple models, the PINNs' performance could be assessed, obtaining the advantages of PINNs and their limitations. In this part, it was decided to take a more difficult modelling problem to assess the performance of PINNs against other modelling methods, namely the modelling of a solid rocket motor, or shortened SRM.

SRMs are being used in various aerospace applications, ranging from spaceflight to military applications [110]. Being able to make predictions about their performance, in terms of generated thrust and chamber pressure, aids in the design process of this type of propulsion system, as well as related subsystems. Furthermore, predictions for SRMs aid in trajectory design for launch vehicles, obtaining the most optimal path from the launch site to the designated orbit in space [137]. In this chapter, the performance of SRMs in terms of generated thrust and chamber pressure will be predicted using a PINN model. After this, the PINN model's performance will be assessed by comparing the predictions from the PINN with a purely Data-Driven method and a Numerical method.

To aid in answering the main research question as presented in chapter 1, this part of the report will focus on the following sub-research questions:

1. *To what extent can a hybrid model, using a Physics-Informed Neural Network, improve the accuracy and computational efficiency of chamber pressure and generated thrust predictions in Solid Rocket Motor modelling compared to a Numerical method?*
2. *To what extent can a hybrid model, using a Physics-Informed Neural Network, improve the accuracy and data efficiency of chamber pressure and generated thrust predictions in Solid Rocket Motor modelling compared to a purely Data-Driven method?*
3. *To what extent does treating solid propellant regression rate characteristics (a and n in $\dot{r} = a \cdot p^n$) as learnable parameters in a Physics-Informed Neural Network affect the accuracy, data efficiency, and computational efficiency of chamber pressure and generated thrust predictions in Solid Rocket Motor modelling, compared to a Numerical and purely Data-Driven method?*
4. *To what extent can a Physics-Informed Neural Network accurately predict chamber pressure and generated thrust in a Solid Rocket Motor when presented with new design parameters (for example, grain and nozzle geometries) not seen during training, compared to Numerical and purely Data-Driven methods, when validated against experimental data? What are potential limitations?*

This part of the report is structured as follows. First, a literature review to obtain missing information regarding SRMs and SRM modelling will be presented in chapter 16. Next, the experimental data used during this part of the report and how the data files were processed will be examined in chapter 17. After this, the numerical model used to model SRMs is explained in chapter 18. Next, the setup of the PINN model will be discussed in chapter 19. Lastly, the performance of the PINN model will be compared against a purely Data-Driven method and a Numerical method in chapter 20 to be able to answer the research questions for this part of the report.

16

Literature Review: Solid Rocket Motor

As mentioned in chapter 15, an SRM is a type of propulsion system. The word propulsion is derived from the Latin words pro, which means forward and pellere, which means to drive [137]. So the translation of propulsion is to drive something forward [137]. In general, propulsion is associated with the change of an object's momentum using a propulsive force, which is called thrust [137]. To generate thrust, a device is needed, which is mostly referred to as a propulsion system [137]. In rocketry, use is made of a pure reaction system, meaning that thrust is generated by the variation of the momentum of the system itself [137]. This reaction does not depend on external factors, which means that, to produce thrust, only internal mass will be expelled [137]. The internal mass used by a propulsion system to generate thrust is called a propellant [137]. A propellant is a combination of a fuel and an oxidiser, which are elements required for a combustion process [137].

In this chapter, background information regarding propulsion methods and systems used in space flight will be examined. Furthermore, details regarding SRMs and the types of physical models used to obtain predictions for chamber pressure and generated thrust by SRMs will be investigated. This chapter is structured as follows. First, a general overview of propulsion systems will be presented in section 16.1. Next, the general working principle of an SRM will be explained in section 16.2. After this, the combustion process of a propellant used in SRMs will be discussed in section 16.3. After this, a general physical model for predicting pressure and thrust of space propulsion systems will be examined in section 16.4. Next, a specific physical model for SRMs will be discussed in section 16.6. After this, the current status of using machine learning in SRMs will be examined in section 16.7. Lastly, an overview of the sources used for this part of the literature review will be presented in section 16.8.

16.1. Types of Space Propulsion Systems

In general, there are various types and classes of space propulsion systems. The most well-known class is the thermal rocket propulsion class [137]. In thermal rocket propulsion, the propellant is first heated up before passing through a nozzle to produce thrust [137]. The heating up of the propellant most often happens due to a combustion process, which is the chemical reaction between a fuel and an oxidiser, generating a high-pressure and temperature gas [137]. Generally, chemical rocket engines are used as propulsion systems in launch vehicles [137]. Chemical rocket engines can be divided into three subcategories, namely [137]:

- Liquid rocket engines
  This type of chemical propulsion system makes use of liquid propellant. The liquid propellant is injected into the combustion chamber for the combustion process [137]. During the injection, droplets of propellant will be generated to stimulate the evaporation process of the propellant from the liquid to the gas phase [137]
- Solid rocket motors
  This type of chemical propulsion system makes use of solid propellants [137]. This means that the fuel and oxidiser are combined and stored in their solid states [137]. The solid propellant is

stored inside the combustion chamber [137].

- Hybrid rocket engines
  This type of chemical propulsion system is a combination of a liquid propulsion system and a solid propulsion system [137]. In general, a solid fuel is used and a liquid oxidiser as propellants for the combustion process [137]. The solid fuel is stored inside the combustion chamber, while the liquid oxidiser is injected into the combustion chamber, making use of injectors [137].

When comparing the various subcategories of chemical propulsion systems, solid propulsion systems offer a wide variety of advantages compared to the other propulsion system types. The main advantages of SRMs are their simplicity, reliability and low cost compared to the other chemical propulsion systems [138]. Furthermore, SRMs can be relatively easily handled, have a long storage life and can be easily stored, while solid propulsion systems are also relatively easy in terms of maintenance [115]. Comparing performances of the various classes of space propulsion systems, making use of Figure 16.1, it can be seen that the thrust of SRMs compared to other systems is high. However, the efficiency of SRMs is lower, as is represented by the specific impulse, I_{sp} [23]. The specific impulse is a measure of the amount of thrust delivered by a certain amount of propellant [137]. The higher this number, the more efficient a space propulsion system is [137].

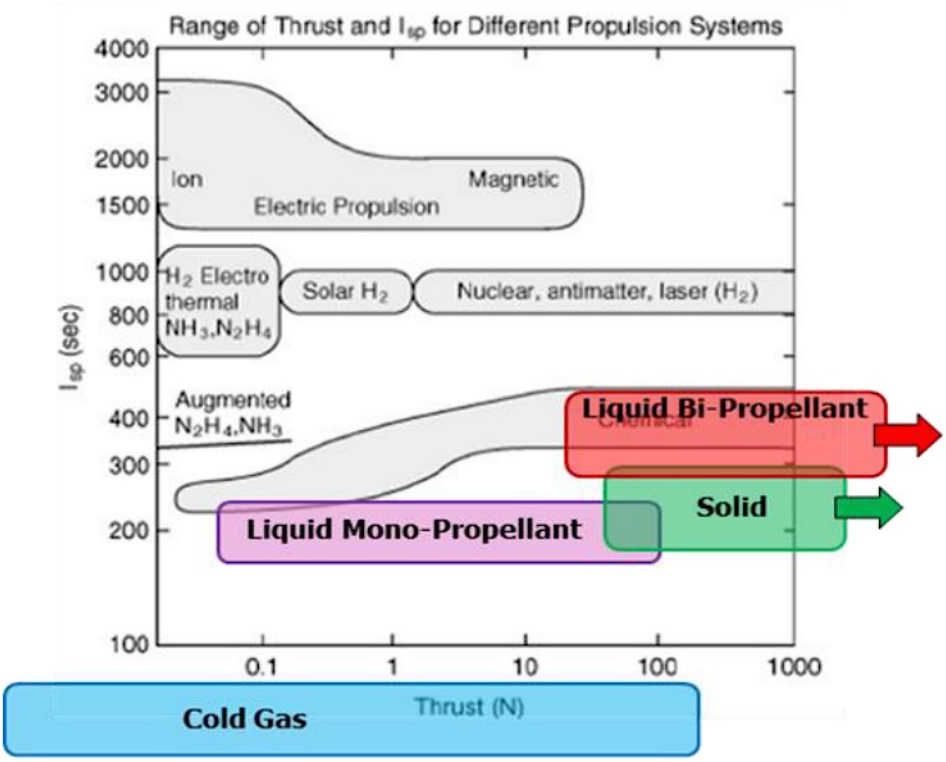


Figure 16.1: Classification of various rocket propulsion systems based on specific impulse and generated thrust, showcasing efficiency of a system compared to the produced thrust [23]

16.2. Working Principle of Solid Rocket Motors

As previously discussed in section 16.1, an SRM is a type of chemical rocket engine which stores propellant in solid form inside the combustion chamber. This means that both the fuel and oxidiser are stored inside one single block of propellant, which is also called a propellant grain [137]. The combustion process takes place on the surface of the propellant grain, where the temperature is high enough to sustain a chemical reaction, while the rest of the propellant acts as an insulator for the wall of the combustion chamber [137]. A typical layout of a solid rocket motor is presented in Figure 16.2.

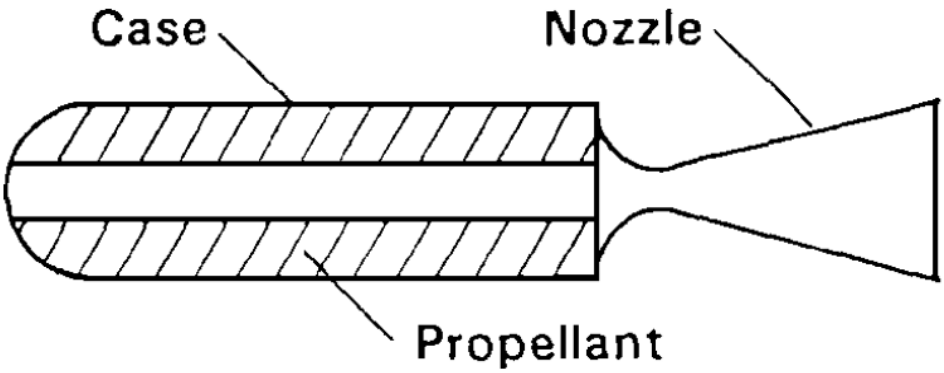


Figure 16.2: Cross section of a solid rocket motor [67]

From Figure 16.2, it can be seen how the propellant grain is typically stored inside a solid rocket motor. The inside, or core, of the propellant grain is hollow [137]. At the sides of the hollow core, the combustion process will take place, in case the other walls are inhibited [137]. The propellant grain size will decrease during the combustion process as the propellant is consumed and expelled as a high-pressure gas through the nozzle to generate thrust [137].

16.3. Combustion Process of a Solid Propellant Grain

In section 16.2, the combustion process of a solid propellant grain was briefly discussed. In this section, the combustion process of an SRM and related aspects will be examined in more detail.

The process of going from a solid propellant to a high-pressure and temperature gas is complex. Usually, the combustion reaction of solid propellants takes place at a very thin layer [137]. The propellant is heated up by the combustion zone, which melts and vaporises the propellant located underneath the combustion zone [137]. This results in new fuel and oxidiser gases for the combustion process. A visual overview of this process is presented in Figure 16.3.

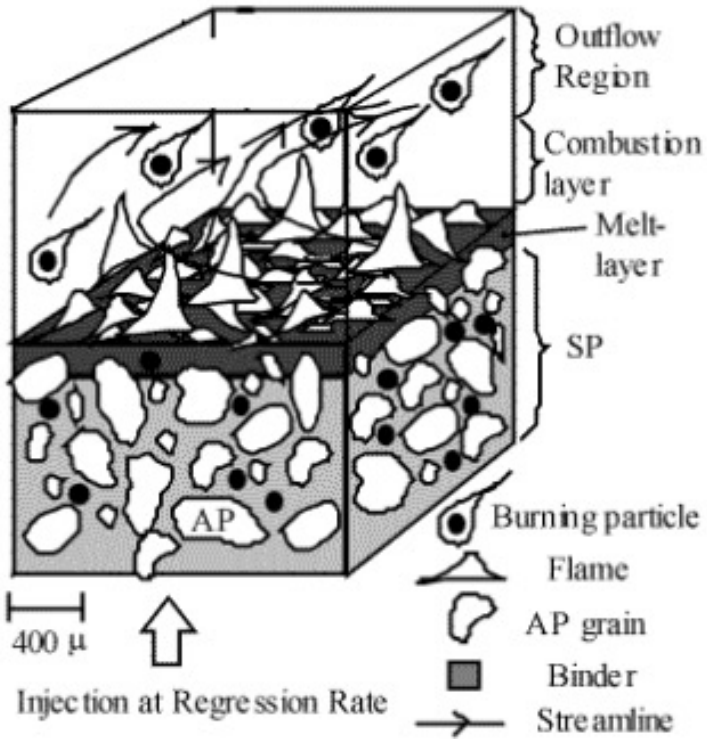


Figure 16.3: Illustration of the combustion process on the edge of the solid rocket motor propellant grain [137]

In Figure 16.3, the combustion process at the edge of the propellant grain is presented. The propellant grain not only consists of a fuel and an oxidiser, but also contains a binder. Besides these materials, commercial propellant grains can consist of additional components, such as multiple oxidisers, curing agents, phase stabilisers, and solvents, as well as other additives which speed up or slow down the burn rate, altering the burn profile [79]. The main purpose of the binder is to hold the fuel and oxidiser together [137]. On the surface of the propellant grain, a multiphase "melt layer" is located at the closest position to the location where the combustion process takes place [137]. Above this melt layer lies the surface flames, and beyond that lies a hydrodynamic outflow region [137]. The mass flow rate of the propellant being transported inside the combustion region can be modelled by making use of Equation 16.1 [137].

$$\dot{m} = \rho_p r S \tag{16.1}$$

In Equation 16.1, ρ_p represents the propellant density, r represents the regression rate of the propellant and S represents the burn surface of the propellant [137]. The regression rate can be modelled by making use of Equation 16.2. Equation 16.2, known as De Vielle's law, is an empirical approximation that describes the regression velocity of the burning surface of the propellant grain [137].

$$r = ap^n \tag{16.2}$$

In Equation 16.2, n represents the burning rate exponent and a represents the burning rate coefficient [137]. The regression rate is very sensitive to the parameter n, where high values of n change the regression rate rapidly with operating pressure, resulting in unsafe situations [137]. Therefore, a safe propellant has an exponent n which is smaller than one [137]. Solid propellants are very sensitive to the initial temperature of the grain, meaning that their performance is dependent on environmental conditions [137]. To model the differences in performance based on the environmental conditions, use can be made of Equation 16.3 [137].

$$a = a_0 e^{\sigma(T_i - T_{i,0})} \tag{16.3}$$

In Equation 16.3, a represents the burning rate coefficient, a_0 represents the burning rate coefficient for a standard temperature, σ represents the temperature coefficient, T_i represents the current temperature of the grain and $T_{i,0}$ represents the reference temperature [137]. It should be noted that various numerical values of σ can be obtained from literature [137].

16.3.1. Erosive Burning

To pack as much propellant as possible inside the combustion chamber, the core's diameter can be decreased, enabling a higher mass of propellant to be stored inside the same amount of volume. However, the limiting factor of this decrease is the nozzle throat diameter, which is the absolute minimum at which the diameter of the core can decrease while still being able to burn [137]. However, decreasing the propellant's core diameter has a major downside, namely that the Mach number inside the core increases [137]. As a result, erosive burning could occur, a process where high-temperature gases flow at high velocity over the burning surface of the core to speed up the burn rate of the propellant [137]. This happens when a certain threshold velocity of the gas is exceeded [137]. Failure of solid rocket motors due to erosive burning is a common occurrence, and should therefore be carefully considered [137].

16.3.2. Propellant Geometry and Impact on Burn Profile

The design of a solid rocket grain and the position of inhibitors affect the burn profile of a solid rocket motor. As a visual example, various propellant grain geometries and their effect on the burn profile are presented in Figure 16.4

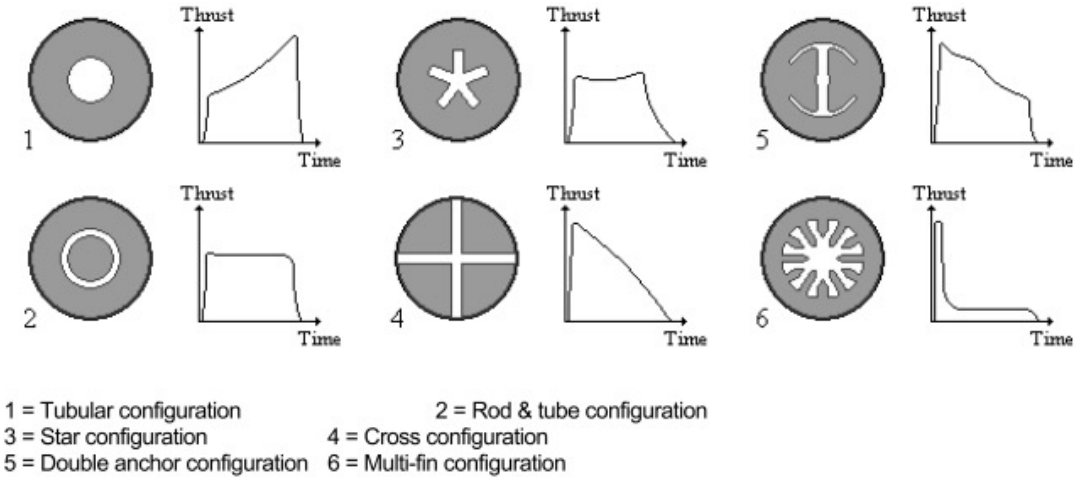


Figure 16.4: Example burn profiles of various grain geometries [137]

From Figure 16.4, it can be observed that a singular port geometry, which is located on the top left of the Figure 16.4, will have an increase in thrust level over time, while a cross configuration, located in the middle at the bottom row in Figure 16.4, has a decrease in thrust level over time [137]. An even burn profile can be obtained by making use of either a star configuration or a rod & tube configuration [137]. However, these two configurations are relatively more difficult to manufacture compared to a tubular configuration [137]. Another method to obtain an even burn profile in terms of thrust over time is to make use of a Bates grain geometry. A Bates grain geometry makes use of a cylindrical design and a circular port and consists of multiple grain segments [67]. Besides this, the individual propellant grains not only burn from the inside of the grain but also along the ends of the grain [67]. An example of a Bates grain geometry is presented in Figure 16.5.

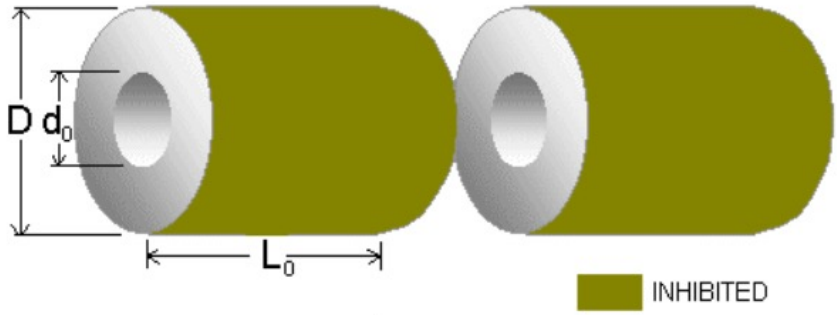


Figure 16.5: Bates grain geometry example [67]

The burning surfaces of the propellant grain can be controlled by making use of inhibitors. An inhibitor is a material that will prevent the combustion process of the underlying material during the entire duration of the burn [79]. Generally, the outsides of a propellant grain consist of inhibitors. Inhibitors can also be used on the insides of the grain to dictate the exposed burning surface areas of the grain during the burn, thereby changing the burn profile [79]. To simplify, inhibitors can be seen as a sort of wall, preventing a specific side of the propellant grain from reaching melting temperatures and, therefore, not being able to contribute to the combustion process. As an example, in Figure 16.6, two grains with a cylindrical hollow core are presented, one with an inhibited grain along the outsides and the other making use of a Bates grain geometry. Furthermore, the direction in which the regression of the propellant will take place is presented by the red arrows, marking the regression direction.

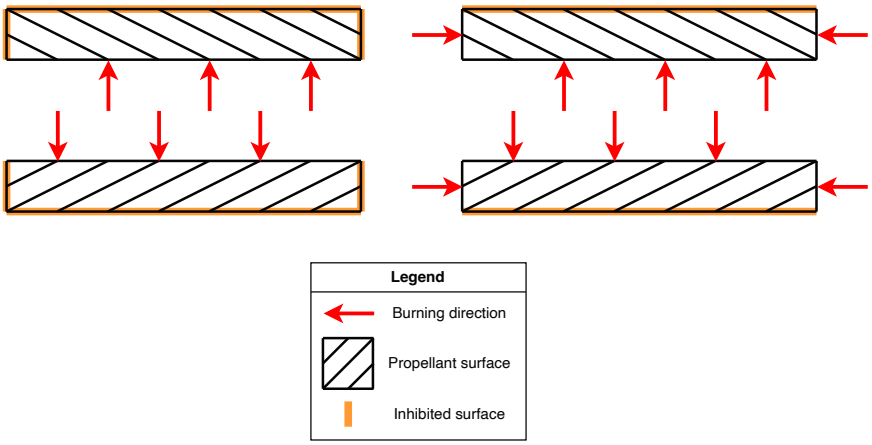


Figure 16.6: Example of burn directions of inhibited propellant grain (left) and bates grain (right) [67]

In Figure 16.6, it can be seen that the burning direction of the inhibited propellant grain is only in the radial direction, while the burning direction of the Bates grain is in both radial and axial directions. As a result of positioning inhibitors at the sides of the propellant grain, the burn profile of a grain changes. For the two grains presented in Figure 16.6, approximations of the corresponding burn profiles are presented in Figure 16.7.

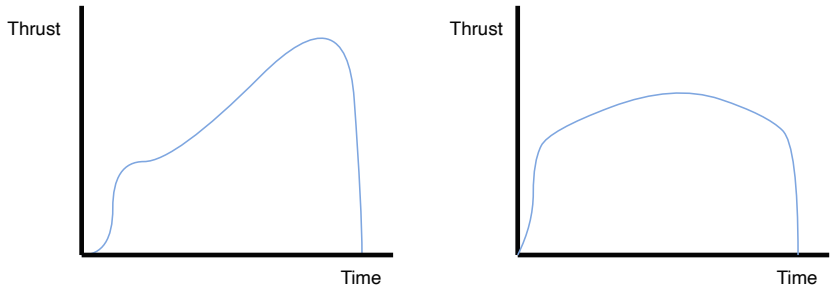


Figure 16.7: Example of a typical burn profile of an inhibited cylindrical hollow core grain (left) and a single Bates grain (right) [67]

From Figure 16.7, it can be seen that while the propellant grain with the inhibited sides has a regressive burn profile, as presented on the left of Figure 16.7, the propellant grain with only the outer surface inhibited has a more neutral burn profile, as presented on the right of Figure 16.7.

16.4. Propellant Grain Properties and Combustion Characteristics

In a rocket engine, a high-pressure gas is transformed into a high-velocity gas and expelled to generate thrust [137]. In thermal rocket propulsion, this high-pressure gas is created by the combustion process of a fuel and oxidiser [137]. In solid rocket motors, the fuel and oxidiser are stored in their solid state as propellant grain inside the combustion chamber [137], as was previously discussed in section 16.1. In this section, KNSB, the type of fuel and oxidiser used in the experimental data as presented in chapter 17 will be examined in more detail. In most amateur solid rocket motors, Potassium Nitrate (KN) is used as oxidiser and Sorbitol as fuel, which, as a combination used in SRM propellant, is better known as KNSB [81]. Sorbitol is an artificial sweetener and has the chemical formula $C_6H_{14}O_6$ [81]. Sorbitol serves as the fuel in the propellant grain as well as the binder [81]. Potassium Nitrate, on the other hand, has the chemical formula KNO_3 and serves as an oxidiser [81]. Potassium Nitrate is also known as saltpetre and is a widely used chemical [81]. The oxidiser-to-fuel ratio (O/F ratio) for KNSB is 65% to 35% and serves as a practical upper limit due to the manufacturing properties [81]. A higher

O/F ratio would lead to more difficulties during the casting process since the substance will have a higher viscosity [81]. The adiabatic combustion properties of KNSB are presented in Table 16.1.

Table 16.1: KNSB combustion properties for an O/F ratio of 65%/35% [80, 81]

Quantity	Value	Unit
γ	1.137	[-]
W_g	39.9	[g/mol]
T_0	1600	[K]
a	5.13	[mm/s - Pa^n]
n	0.222	[-]

The adiabatic flame temperature, which is defined as the temperature the flame would have in case no heat is lost to the surroundings, will differ for variations in O/F ratios [118]. The change in adiabatic flame temperature of KNSB for different O/F ratios is presented in Figure 16.8.

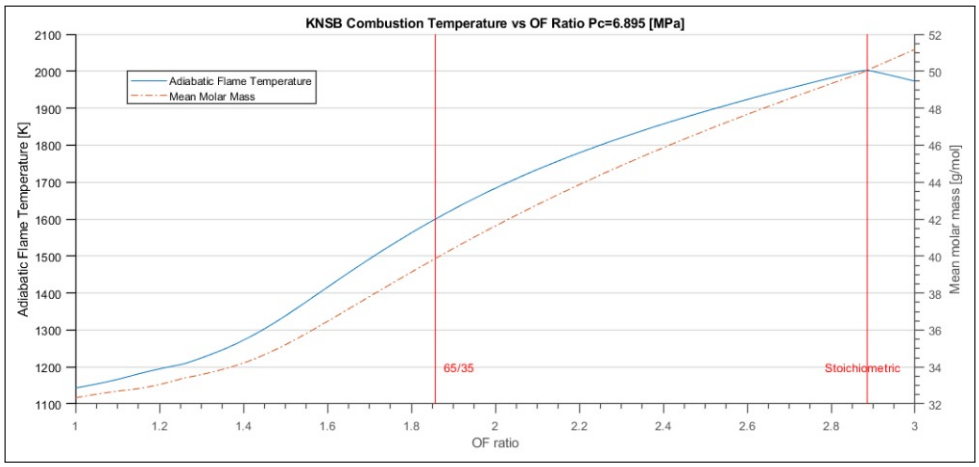


Figure 16.8: Adiabatic flame temperature and mean molar mass of the reaction products for KNSB as a function of O/F ratio [91]

From Figure 16.8, it can be seen that by increasing the O/F ratio, the adiabatic flame temperature will increase up until stoichiometric conditions are reached, whereafter the temperature of the flame decreases again. Stoichiometric conditions are defined as the process where all the reactants are converted into products [114]. The stoichiometric formula for KNSB is presented in Equation 16.4 [78].

$$\mathrm{C_6H_{14}O_6} + 3.345\,\mathrm{KNO_3} \longrightarrow 1.870\,\mathrm{CO_2} + 2.490\,\mathrm{CO} + 4.828\,\mathrm{H_2O} + 2.145\,\mathrm{H_2} + 1.672\,\mathrm{N_2} + 1.644\,\mathrm{K_2CO_3} + 0.057\,\mathrm{KOH} \tag{16.4}$$

The burn rate of a propellant is obtained by making use of experiments. However, since multiple institutions have performed those measurements, various coefficients and burn rates have been measured for KNSB. During this thesis, the burn rate and corresponding coefficients which will be used have been recoded by Magnus, providing a burn rate coefficient and pressure coefficient as presented in Figure 16.9 [81].

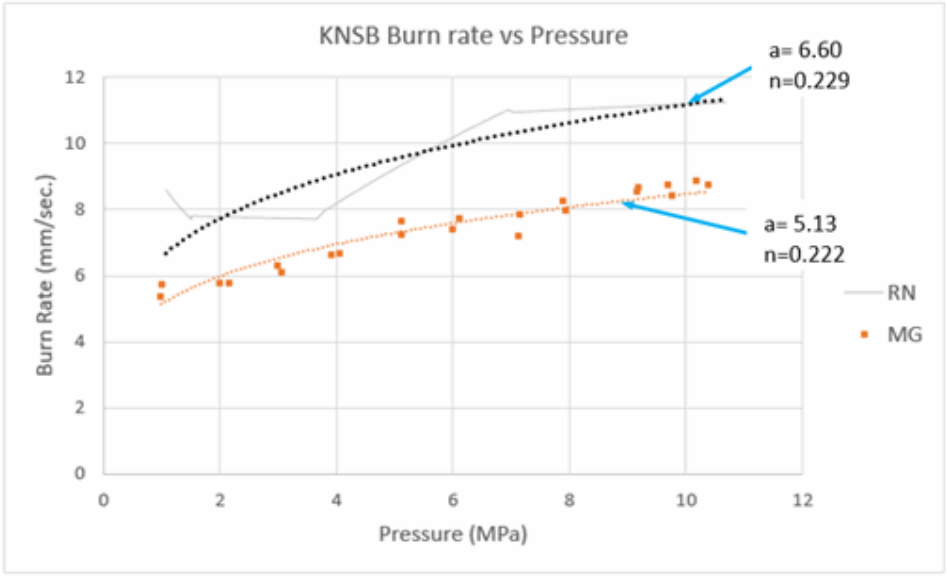


Figure 16.9: Experimental results of Magnus of the burn rate against the pressure [81]

16.5. Equations for the Ideal Rocket Motor

A rocket engine provides a thrust force by using high-pressure combustion gases. This is a complex process, but can be described by making use of ideal rocket theory [137]. In this section, firstly, an equation for thrust will be obtained and examined, whereafter various other important parameters will be presented.

16.5.1. Thrust

In general, thrust is generated by expelling high-velocity gases through the nozzle [137]. The pressure inside the combustion chamber will increase during the combustion process due to nozzle of the rocket nozzle, which limits the outflow of the combustion gases [137]. The amount of thrust produced during this process can be modelled by making use of Equation 16.5 [137].

$$F = \dot{m} \cdot U_e + (p_e - p_a) \cdot A_e \tag{16.5}$$

In Equation 16.5, $\dot{m}$ represents the mass flow rate of the propellant leaving the engine, U_e represents the exit exhaust velocity of the gas, p_e represents the pressure of the gas at the exit of the nozzle, p_a represents the atmospheric pressure of the surroundings at the exit of the nozzle and A_e represents the area of the nozzle exit [137]. The term $\dot{m} \cdot U_e$ is also referred to as momentum thrust and the term $(p_e - p_a) \cdot A_e$ is referred to as pressure thrust [137]. Effectively, this means that Equation 16.5 is dependent on the linear momentum through the nozzle exit and the pressure thrust, which indicates if there is over-expansion (pressure thrust is negative), optimal (pressure thrust is zero) or under-expanded (pressure thrust is positive) [137]. The expansion of the flow is dependent on the rocket motor nozzle [137]. In general, a rocket nozzle can either be over-, ideal or under-expanded. These types of flow expansion are visually presented in Figure 16.10. Over- and under-expanded nozzles are decreasing the efficiency of the motor [137]. Therefore, a slightly over-expanded nozzle is used for launch vehicles intended for space flight launching from Earth, since the largest part of the time, the launch vehicle is present in the higher portions of Earth's atmosphere, where the pressure is lower compared to sea level values [137].

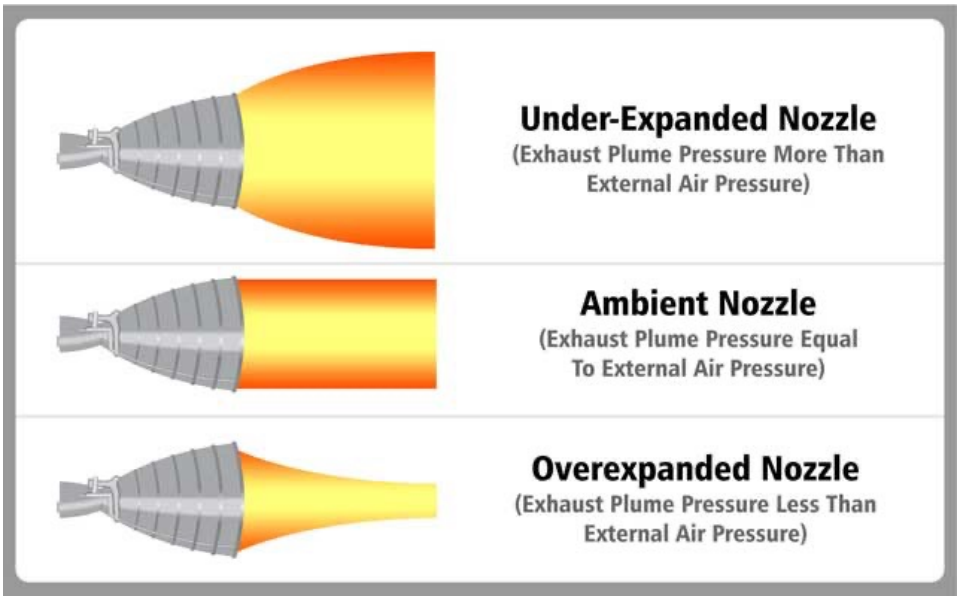


Figure 16.10: Rocket motor nozzle expansion types

16.5.2. Ideal Rocket Equations

From Equation 16.5, it becomes clear that obtaining a high exhaust velocity and mass flow rate is advantageous for obtaining high thrust force. To generate high exhaust velocity, high-pressure gases are required. In rocketry, these high-pressure gases are generated by making use of the combustion process of the propellants, as was explained in section 16.2. The exhaust velocity can be estimated by making use of assumptions which lead to the ideal rocket motor. This equation can relate the change in pressure, temperature and thermal energy over the nozzle to exhaust velocity [137]. Using the same assumptions, equations can be obtained relating exhaust velocity and thrust with the size of the rocket nozzle, which is referred to as ideal nozzle theory [137]. The most important assumptions for the ideal rocket motor and ideal nozzle theory are summarised in the list below [137]:

- The exhaust gases are homogeneous and have a constant composition
- The gas or gas mixture obeys the ideal gas law, $PV = nR_aT$
- The heat capacity of the gas or mixture of gases expelled is constant
- The flow through the nozzle is one-dimensional, steady and isentropic

When using these assumptions, the equation for the exhaust velocity can be obtained, which is presented in Equation 16.6.

$$U_e = \sqrt{\frac{2\gamma}{(\gamma-1)}\frac{R_a}{M_w}T_c\left(1-\left(\frac{p_e}{p_c}\right)^{\frac{\gamma-1}{\gamma}}\right)} \tag{16.6}$$

In this equation, γ represents the specific heat ratio, R_a represents the gas constant, M_w represents the mass of the gas, T_c represents the chamber temperature, p_e represents the exit pressure and p_c represents the pressure inside the combustion chamber. Previously, the type of expansion and the role of a solid rocket motor nozzle were briefly discussed. A typical shape of a rocket engine nozzle is presented in Figure 16.11. It can be seen that first, the flow from the combustion chamber will flow through a convergent part, whereafter it reaches the nozzle throat, flowing into the divergent part.

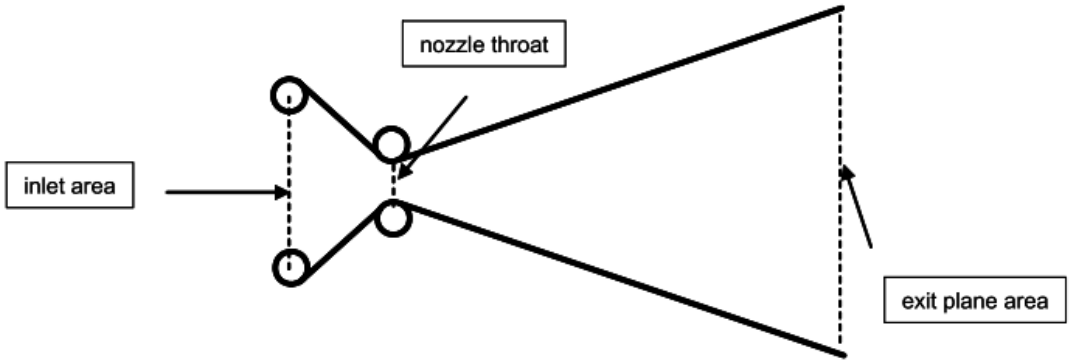


Figure 16.11: Typical convergent-divergent nozzle used in rocket engines [137]

During the passage over the rocket motor nozzle, the quantities of the flow will change. First, the flow will accelerate in the convergent part, whereafter it will reach the speed of sound, Mach 1, at the nozzle throat [137]. At this point, the flow is called sonic, and the divergent part will increase the velocity of the flow even more, resulting in supersonic flow regimes at the end of the nozzle exhaust [137]. During the passage of the flow through the rocket motor nozzle, not only will the velocity of the flow change, but other quantities such as the pressure and the temperature will change as well. In Figure 16.12, a typical change in velocity, pressure and temperature of the flow through a rocket engine nozzle is presented at various locations.

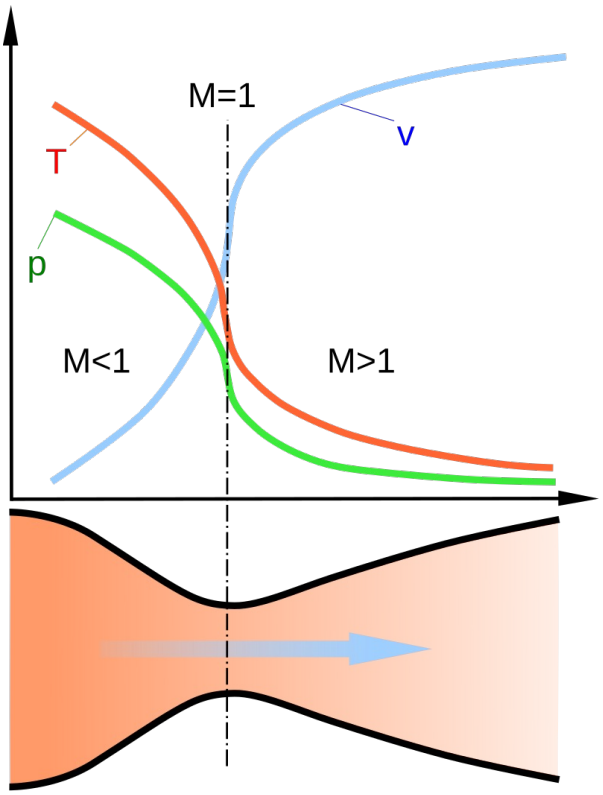


Figure 16.12: Variation of temperature, velocity and pressure of the combustion gases during the passage through a convergent-divergent nozzle [30]

It can be seen from Figure 16.12 that while the velocity of the flow increases, the pressure and temperature decrease during the flow passing through the nozzle. However, this is only true if the flow at the nozzle becomes sonic [137]. The velocity at the throat can only become sonic in case the nozzle becomes choked, meaning that no more mass flow can pass through the nozzle [137]. To reach this

condition, a certain pressure ratio between the chamber pressure and exit pressure should be reached. The relationship between the pressure ratio of the required pressure to reach sonic conditions at the throat and the total pressure is presented in Figure 16.13.

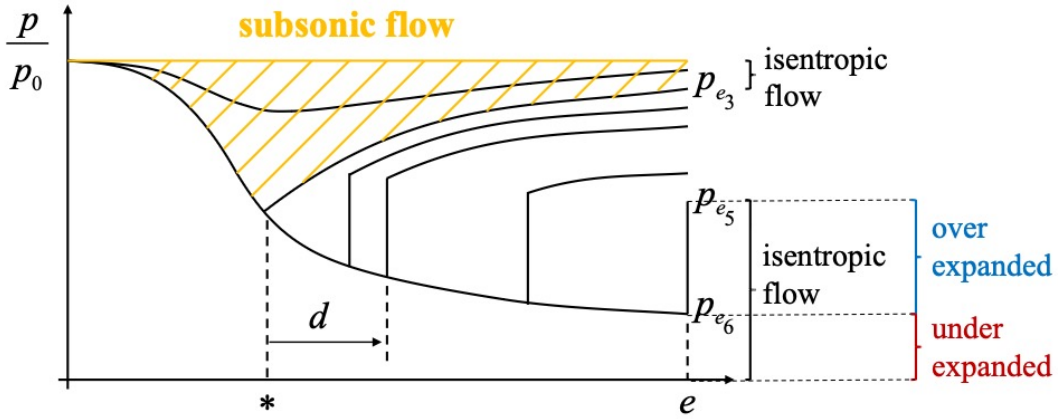


Figure 16.13: Relationship between pressure difference and distance along the nozzle [122]

In Figure 16.13, the pressure relation between overexpansion and underexpansion of the nozzle flow at the exit is also presented, represented by P_{e_5} and P_{e_6}, respectively. In case the flow is choked, the so-called critical mass flow rate, or the mass flow rate the flow would have in case the flow is choked at the throat, could be calculated, making use of Equation 16.7 [137]. In this equation, use is being made of the Vandenkerckhove function, which can be calculated making use of Equation 16.8 [137].

$$\dot{m} = \frac{\Gamma p_c A_t}{\sqrt{RT_c}} \tag{16.7}$$

$$\Gamma = \sqrt{\gamma}\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{2(\gamma-1)}} \tag{16.8}$$

In Equation 16.7 and Equation 16.8, p_c represents the chamber pressure, A_t represents the throat area, R represents the specific gas constant $R = \frac{R_a}{M_w}$, T_c represents the combustion chamber temperature and γ represents the specific heat ratio of the flow [137]. To obtain a choked mass flow, a certain pressure ratio should be obtained [137]. To obtain this pressure ratio, there exists a relationship between the pressure ratio of the chamber pressure and the exit pressure against the nozzle throat and nozzle exit area [137]. This relationship is presented in Equation 16.9 [137].

$$\frac{A_e}{A_t} = \frac{\Gamma}{\sqrt{\frac{2\gamma}{\gamma-1}\left(\frac{p_e}{p_c}\right)^{\frac{2}{\gamma}}\left(1-\left(\frac{p_e}{p_c}\right)^{\frac{\gamma-1}{\gamma}}\right)}} \tag{16.9}$$

An important performance parameter for rocket engines is the thrust coefficient [137]. The thrust coefficient determines the amplification of the thrust due to the gas expansion in the nozzle as compared to the thrust that would be exerted if the chamber pressure acted over the throat area only and if there were no chamber flow [137]. This parameter can be calculated by making use of Equation 16.10 [137].

$$C_f = \Gamma\sqrt{\frac{2\gamma}{\gamma-1}\left(1-\left(\frac{p_e}{p_c}\right)^{\frac{\gamma-1}{\gamma}}\right)} + \left(\frac{p_e}{p_c}-\frac{p_a}{p_c}\right)\frac{A_e}{A_t} \tag{16.10}$$

Besides the thrust coefficient, another performance parameter is the characteristic velocity. The characteristic velocity is a measure of the energy level of the propellants available for propulsion purposes [137]. The characteristic velocity can be calculated by making use of Equation 16.11 [137].

$$c^* = \frac{1}{\Gamma}\sqrt{RT_c} \tag{16.11}$$

In Equation 16.5, a relationship was presented to obtain the thrust force of a chemical rocket engine. However, another way of determining the thrust force is to make use of the mass flow rate, the thrust coefficient and the characteristic velocity, as is presented in Equation 16.12 [137].

$$F = \dot{m} C_f c^* \tag{16.12}$$

16.5.3. Quality Factors for Ideal Rocket Theory

Unfortunately, most of the time, the ideal rocket theory does not provide sufficient accuracy to match the previously presented physical models with real-world experiments [137]. Therefore, a set of correction factors has been introduced to be able to bridge the gap between the simplified ideal rocket theory and the complex behaviour of rocket propulsion systems [137]. In Equation 16.13, the most common quality factors or correction factors are provided, namely the thrust quality, discharge coefficient, nozzle flow quality, combustion quality and propellant consumption quality [137].

$$\begin{aligned}
\xi_F &= \frac{F_{\text{real}}}{F_{\text{ideal}}} \\
C_d &= \frac{m_{\text{real}}}{m_{\text{ideal}}} \\
\xi_n &= \frac{C_{f,\text{real}}}{C_{f,\text{ideal}}} \\
\xi_c &= \frac{c^*_{\text{real}}}{c^*_{\text{ideal}}} \\
\xi_s &= \frac{I_{sp,\text{real}}}{I_{sp,\text{ideal}}}
\end{aligned} \tag{16.13}$$

In Equation 16.13, I_{sp} represents the specific impulse [137]. As previously explained in section 16.1, the specific impulse is a measure of the efficiency of the engine, comparing the amount of produced impulse for a given amount of used propellant mass. The specific impulse can be obtained by making use of Equation 16.14 [137].

$$I_{sp} = \frac{1}{g_0}\frac{\int_0^t F(t)\,dt}{\int_0^t \dot{m}(t)\,dt} \tag{16.14}$$

In Equation 16.14, g_0 presents the earths gravitational constant at sea level, $F(t)$ presents the thrust delivered over time and $\dot{m}(t)$ presents the mass flow rate delivered over time. The integral on the top of Equation 16.14 calculates the total impulse delivered by the system, and the bottom integral calculates the total amount of propellant used, which delivered the total impulse.

16.6. Numerical Model for Solid Rocket Motors

In section 16.5, it was explained that a lot of factors determine the performance of an SRM. A simplified physical model in the form of state equations to model SRM's performance will be presented in this section. In Equation 16.15, the state equations for an SRM are presented using a cylindrical grain with a hollow circular core, and in Equation 16.16, the state equations are presented, but adapted for erosive burning [67].

$$\begin{bmatrix} \dot{P}_0 \\ \dot{r} \\ A_{\text{burn}} \\ V_c \end{bmatrix} = \begin{bmatrix} \left(\frac{A_{\text{burn}} \cdot \dot{r}}{V_c}\right) \cdot \left(\rho_p \cdot R_g \cdot T_0 - P_0\right) - \left(\frac{A^*}{V_c}\right) \cdot P_0 \cdot \sqrt{\gamma \cdot R_g \cdot T_0 \cdot \left(\frac{2}{\gamma+1}\right)^{\left(\frac{\gamma+1}{\gamma-1}\right)}} \\ a \cdot P_0^n \\ 2 \cdot \pi \cdot r \cdot L_{\text{port}} \\ \pi \cdot r^2 \cdot L_{\text{port}} \end{bmatrix} \tag{16.15}$$

In Equation 16.15, r represents the radius of the core and L_{port} represents the length of the core or grain [67].

$$\begin{bmatrix} \dot{P}_0 \\ \dot{r} \\ A_{\text{burn}} \\ V_c \end{bmatrix} = \begin{bmatrix} \left(\frac{A_{\text{burn}} \cdot \dot{r}}{V_c}\right) \cdot \left(\rho_p \cdot R_g \cdot T_0 - P_0\right) - \left(\frac{A^*}{V_c}\right) \cdot P_0 \cdot \sqrt{\gamma \cdot R_g \cdot T_0 \cdot \left(\frac{2}{\gamma+1}\right)^{\left(\frac{\gamma+1}{\gamma-1}\right)}} \\ a \cdot P_0^n \cdot \left(\frac{1+k \cdot \frac{M_{\text{port}}}{M_{\text{crit}}}}{1+k}\right) \\ 2 \cdot \pi \cdot r \cdot L_{\text{port}} \\ \pi \cdot r^2 \cdot L_{\text{port}} \end{bmatrix} \tag{16.16}$$

In Equation 16.16, K represents an empirical scale factor, M_{port} represents the critical Mach number of the port and M_{crit} represents the critical or threshold Mach number [67].

The state equations presented in Equation 16.15 and Equation 16.16 can be used to create a simplified physical model for the prediction of the pressure inside the combustion chamber over time. To obtain the generated thrust, use can be made of the equations from the ideal rocket theory, making use of the predicted chamber pressure from the state space model. However, the state equation model oversimplifies reality and is derived by making use of assumptions, which are summarised below [67]:

- The throat of the rocket nozzle chokes immediately at start-up
- A constant flame temperature is assumed inside the chamber
- The model is a 1-D simulation (simulation does not take into account position inside the combustion chamber)
- Model assumes a 1-segment, cylindrical grain with a circular hollow core.

For Bates grains, these state equations should be changed. The equations for the burn area, A_{burn} and the chamber volume, V_c, need to be modified, which results in an updated state space model as presented in Equation 16.17 [67].

$$\begin{bmatrix} A_{\text{burn}} \\ V_c \end{bmatrix} = \begin{bmatrix} N \cdot \pi \cdot \left\{ \left[\frac{D_0^2 - (d_0 + 2 \cdot s)^2}{2} \right] + (L_0 - 2 \cdot s) \cdot (d_0 + 2 \cdot s) \right\} \\ \frac{N \cdot \pi}{4} \cdot \left\{ (d_0 + 2 \cdot s)^2 \cdot (L_0 - 2 \cdot s) + D_0^2 \cdot 2s \right\} \end{bmatrix} \tag{16.17}$$

In Equation 16.17, s can be calculated by making use of Equation 16.18 [67].

$$s = r(t) - \frac{d_0}{2} \tag{16.18}$$

16.7. Current Status of Machine Learning and Solid Rocket Motor Modelling

Machine learning opens up a lot of possibilities for launch vehicle design and monitoring, as well as for their related subsystems [45]. Mostly, ML can be used to accelerate certain processes while reducing errors and enhancing human-in-the-loop decision-making [45]. The main advantage of ML is its capability of detecting trends and patterns in large datasets [45]. This could be beneficial for all sorts of activities, but for propulsion systems in launch vehicles, the area of anomaly detection would benefit the most. Currently, anomaly detection methods are based on redline limits and family data [45]. These redline limits have been determined using historical data, qualification campaigns, engineering judgment and analysis [45]. However, these limits do not take into account unpredictable failure modes or provide anomaly diagnosis [45]. An illustration of an example of a redline limit is presented in Figure 16.14.

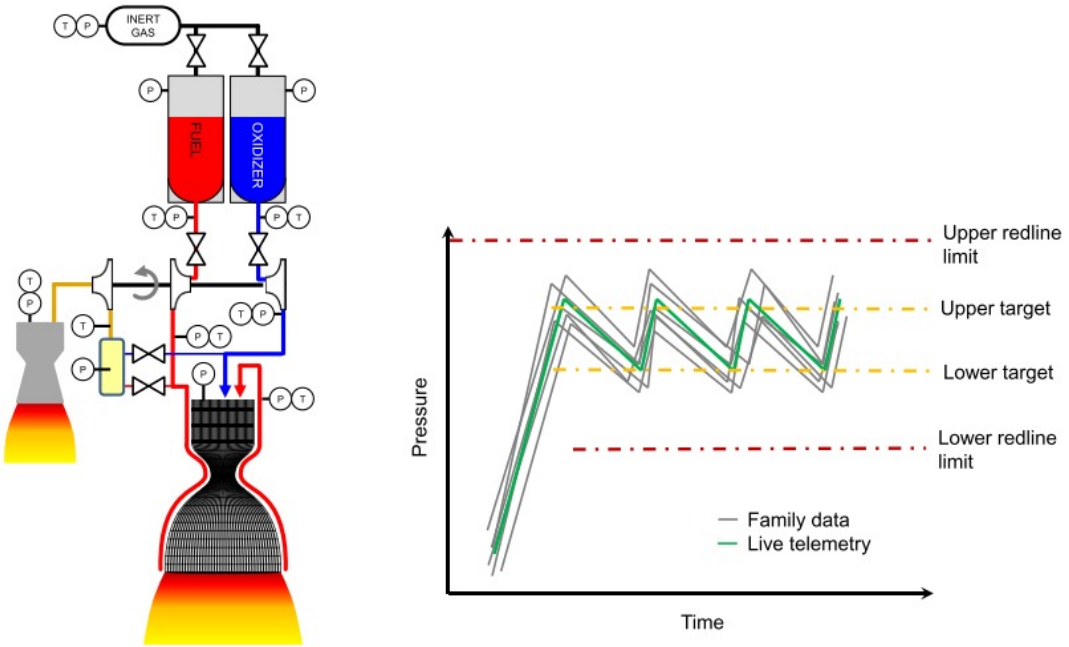


Figure 16.14: Example of redline limits during operation of a rocket engine [45]

However, ML alone can not directly be applied to new systems and is barely capable of detecting untrained anomalies [45]. PINNs could be an interesting option to enable ML methods to identify untrained anomalies [45]. Another application of PINNs for launch vehicles could be non-destructive testing algorithms and during launch operations [45]. Anomaly detection making use of ML has been successfully tested on the space shuttle main engine and showed promising results for future research [103].

16.7.1. Data-Driven Methods

Currently, SRM performance is being determined using ground test analysis [138]. These ground-based tests can obtain the thrust while operating the motor, but this method is irreversible, costly, complex and difficult to perform multiple times [138]. Therefore, different methods exist to determine SRM performance without testing [138]. In general, two main paths exist to model the performance of SRM. One is a model-based method, and the other is a fully data-driven method [138]. Model-based methods require a good understanding of the underlying physical processes occurring inside the SRM, as well as extensive data processing of test data [138]. Purely data-driven methods do not need this prior knowledge, but they do need a large amount of experimental data to be able to learn the underlying physical processes and to model them correctly [138]. Various data-driven methods have been used to successfully model SRMs, making use of neural networks, support vector machines and extreme learning machines [138]. Besides performance prediction, other data-driven methods have been developed

and tested to determine SRM defects [138]. It has been shown that convolutional neural networks can be used to diagnose defects in solid rocket motors [138], as well as to use convolutional neural networks to investigate X-ray data for the detection of solid propellant grains [138]. X-ray analysis is crucial for determining if a propellant grain is still able to operate according to the specifications or if refurbishment is required [42]. Currently, human experts will assess X-rays of solid propellant grains, which is a time-consuming and labour-intensive process [42]. Convolutional neural networks have proven to be able to automatically detect flaws with the propellant grain from X-ray data [42]. Besides this, SRM's performance could also be determined accurately, making use of a different type of convolutional neural network [138]. However, this method could only be used with relatively high accuracy on small time windows, meaning that the accuracy of the prediction decreases with an increasing time frame [138]. Ultimately, data-driven methods could reduce the reliance on ground testing, resulting in more efficient evaluation and reduced cost of solid rocket motor performance estimation [138].

16.7.2. Physics-Informed Neural Network Models

Currently, Physics-Informed Neural Networks have not been directly used for SRM performance modelling. However, a PINN has been developed to model the regression transient of the burning surface of the solid propellant [115]. Mostly, use has been made of analytical and mesh methods for these types of simulations [115]. The burning rate is both spatially and temporally dependent, meaning that this relationship can only be solved by making use of partial differential equations [115]. As discussed in subsection 2.1.2, PINNs could solve partial differential equations, making use of a meshless approach, potentially obtaining a higher accuracy due to the absence of a truncation error. The developed model to predict the regression transient of the burning surface was able to predict surface burning in a general way with little training data available, showcasing that PINNs are a very data-efficient method while maintaining accuracy [115].

16.7.3. Digital Twins

Data-driven methods and PINNs can be used as models to make predictions about the performance of a system. However, these modelling methods do not take into account live measurements or the life cycles of a system. An approach to integrate live measurements and life cycles into the modelling methods, use can be made of a digital twin. The concept of a twin originates from NASA's Apollo program [138, 36]. The concept of a twin was to build two spacecraft, where one was launched to perform a certain mission while the other was used to simulate all the processes the mission spacecraft encountered, such that reference data for the maintenance of the mission spacecraft would be available on the ground [138]. This process is called physical twining since there is an actual physical twin present. In 2003, the concept of a digital twin was introduced [138]. A digital twin is a virtual representation of a physical system [138]. A digital twin infrastructure exists out of three parts: the physical system itself, the digital representation of the physical system and a bidirectional data stream between the two systems [138]. The big advantage of a digital twin is to have a virtual model upon which you could perform tests and obtain predictions at a future state, rather than needing a physical model to predict these quantities [138]. A digital twin should have access to real-time data and should be ahead of the physical system, meaning that the digital twin will be able to predict the future of the physical system [138]. Furthermore, a digital twin should be capable of simulating the performance of a physical system in real time [36]. A digital twin could be developed for SRMs and their systems as well. A digital twin could exist out of multiple virtual models, which together model the behaviour of the SRM [138]. For example, a virtual model for the nozzle stress and strain can exist, as well as a virtual model for the combustion process. Combined, this would form a digital twin for life cycle assessment.

The difference between digital twins and other modelling methods, such as PINNs and Data-Driven methods, is that a digital twin has a real-time, continuous connection to its physical system. If a problem is detected, the digital twin will calculate a corresponding approach to counteract this problem and will send this to the physical system or systems acting on the physical system [138]. Ideally, a digital twin is created at the beginning of the design and production phase [138]. For SRMs, a digital twin can be created for the predictive maintenance of the propellant [138]. However, for this application, sensors should be located inside the grain, which brings additional complexities such as making use of sensors which could sustain the high temperature during the casting process [138]. A digital twin can also be used during the design phase of an SRM. During the design phase, the data involved in the digital twin

model originates from simulations [36]. Normally, a digital twin is created in steps. First, a digital model is created, which is then transformed into a digital shadow [130]. Lastly, the digital shadow is being transformed into a digital twin [130]. The difference between a digital model, digital shadow and digital twin is visually presented in Figure 16.15.

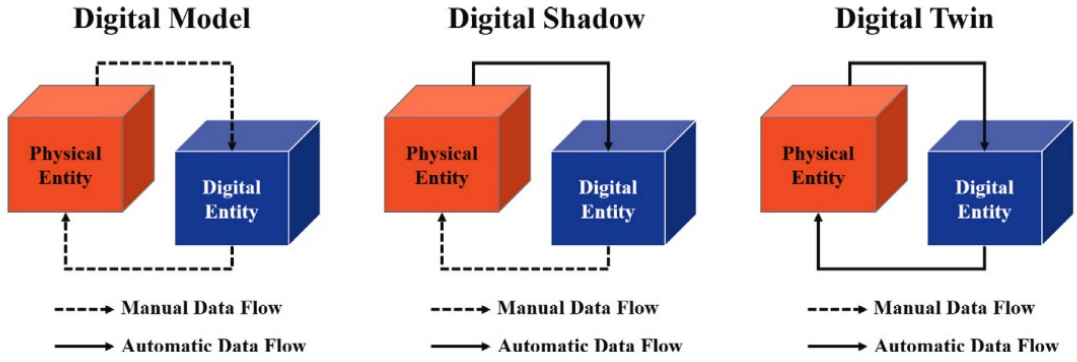


Figure 16.15: Illustration of the differences between a digital model, a digital shadow and a digital twin [130]

16.8. Overview of Sources

Throughout the literature review for the SRM part of the report, use have been made of various sources. To provide an overview of the sources used per section, the sources have been summarised in Table 16.2.

Table 16.2: Overview of Sources and Their Usage in the Literature Review

Section	References
Solid Rocket Motor Modelling	
Introduction	[137], [138], [23]
Working Principle of Solid Rocket Motors	[137], [67]
Combustion Process of a Solid Propellant Grain	[137], [79]
Erosive Burning	[137]
Propellant Geometry and Impact on Burn Profile	[137], [67], [79]
Propellant Grain Properties and Combustion Characteristics	[137], [81], [78], [80], [114], [118], [91], [55]
Equations for the Ideal Rocket Motor	[137]
Thrust	[137]
Ideal Rocket Equations	[137], [30], [122]
Numerical Model for Solid Rocket Motors	[67]
Current Status of Machine Learning and Solid Rocket Modelling	[45], [103]
Data-Driven Methods	[138], [42],
Physics-Informed Neural Network Models	[115]
Digital Twins	[138], [36], [130]

17

Experimental Setup and Data Processing: Solid Rocket Motor

After obtaining the required background information about SRMs from chapter 16 and processing and removing noise from experimental data, as presented in section 2.2, the obtained experimental SRM data will be processed in this chapter. First, an overview of the DARE BEM SRM, which has been used to collect the first part of the experimental data, will be examined in section 17.1. Next, an overview of the SRM2020 (EE6), which has been used to collect the other parts of the experimental data, will be examined in section 17.2. After this, an overview of the experimental data, in terms of how they were recorded and stored, how they were calibrated and filtered, as well as all the experimental data and an explanation of each file, is presented in section 17.3. Lastly, a partitioning strategy to partition the experimental data for the surrogate modelling tests will be presented in section 17.4

17.1. Overview of DARE BEM

The first data package contains data from the DARE BEM solid rocket motor tests. The DARE BEM was designed to be able to rapidly test a wide variety of propellant grains and their characteristics, such that the steady-state regression rate could be calculated [91]. Later, this engine was used to perform practicals to determine the performance for different grain compositions and nozzle geometries [113]. In the provided data sets, experimental data for differences in nozzle geometry and propellant grain properties are present. In Table 17.1, the overall characteristics in terms of the geometry of the DARE BEM are presented. In Appendix E, the technical drawings for the various parts are presented.

Table 17.1: Geometry specifications of the DARE BEM solid rocket motor [113]

Quantity	Value	Unit
Engine casing geometry		
Outside casing length	170	[mm]
Outside casing diameter	90	[mm]
Wall thickness	5	[mm]
Chamber diameter	80	[mm]
Chamber length	118.5	[mm]
Nozzle geometry		
Reference nozzle		
Throat diameter	8.37 ± 0.0005	[mm]
Exit diameter	16.73 ± 0.0005	[mm]
Nozzle length	42.73 ± 0.1	[mm]
Contraction half angle	45	[°]
Exit cone half angle	12	[°]
Smaller throat nozzle		
Throat diameter	7 ± 0.1	[mm]
Exit diameter	14 ± 0.1	[mm]
Nozzle length	40.2 ± 0.1	[mm]
Contraction half angle	45	[°]
Exit cone half angle	12	[°]
Larger throat nozzle		
Throat diameter	9.57 ± 0.1	[mm]
Exit diameter	19.14 ± 0.1	[mm]
Nozzle length	41.73 ± 0.1	[mm]
Contraction half angle	45	[°]
Exit cone half angle	12	[°]
Propellant grain properties		
Standard geometry		
Length of the grain	115 ± 0.1	[mm]
Port length	105 ± 0.5	[mm]
Outside diameter	80 ± 0.05	[mm]
Liner wall thickness	1.6 ± 0.05	[mm]
Port diameter	25 ± 0.05	[mm]
Propellant mass	758 ± 10	[g]
Bates geometry		
Length of the grain	115 ± 0.1	[mm]
Port length	50 ± 1	[mm]
Empty Spacing	5 ± 0.1	[mm]
Outside diameter	80 ± 0.5	[mm]
Liner wall thickness	1.6 ± 0.5	[mm]
Port diameter	25 ± 0.5	[mm]
Propellant mass	360 ± 10	[g]
Number of grains	2	[-]
Composition		
Propellant mass	97	[g]
Fuel (+ binder)	Sorbitol	[-]
Oxidizer	Potassium Nitrate	[-]
O/F	65/35	[%]
Igniter		
Standard		
Composition	Black powder & nitrocellulose primer	[-]
Mass	2.0	[g]

17.2. Overview of SRM2020 (EE6)

The second data package contains data from the DARE SRM 2020 and the DARE SRM 2020 EE6. These engines are solid rocket motors making use of Bates grain configurations. The SRM 2020 and the SRM 2020 EE6 have been used for experiments regarding propellant grain configurations and igniter types. The difference between the SRM 2020 and the SRM 2020 EE6 is that the EE6 has a longer chamber length and can, therefore, store more individual propellant grains. The provided data set provides experimental data for changes in grain geometry, coating strategies, as well as changing igniter types. In Table 17.2, the overall characteristics of the DARE SRM 2020 and SRM 2020 EE6 are

presented. In Appendix G, the technical drawings are presented.

Table 17.2: Geometry specifications of the DARE SRM 2020 solid rocket motor [48, 56, 69]

Quantity	Value	Unit
Engine casing geometry		
Standard configuration		
Outside casing length	317	[mm]
Outside casing diameter	48	[mm]
Wall thickness	2	[mm]
Chamber diameter	44	[mm]
Chamber length	275	[mm]
Number of grains	4	[-]
Spacer ring length	7	[mm]
Number of spacer rings	5	[-]
EE6 configuration		
Outside casing length	451	[mm]
Outside casing diameter	48	[mm]
Wall thickness	2	[mm]
Chamber diameter	44	[mm]
Chamber length	409	[mm]
Number of grains	6	[-]
Spacer ring length	7	[mm]
Number of spacer rings	5	[-]
Nozzle geometry		
Throat diameter	11	[mm]
Exit diameter	22	[mm]
Nozzle length	43	[mm]
Contraction half angle	50	[°]
Exit cone half angle	10.5	[°]
Propellant grain properties		
Geometry		
Length of the grain	60	[mm]
Port length	60	[mm]
Outside diameter	43.5	[mm]
Liner wall thickness	1.75	[mm]
Port diameter	20	[mm]
Composition		
Propellant mass	97	[g]
Fuel (+ binder)	Sorbitol	[-]
Oxidizer	Potassium Nitrate	[-]
O/F	65/35	[%]
Coating	Black Powder & Nitrocellulose	[-]
Coating mass composition	66/33	[%]
Igniter		
Standard		
Composition	Black powder	[-]
Mass	1.5	[g]
Black powder and Magnesium		
Composition	Black powder & Magnesium	[-]
Mass composition	50/50	[%]
Mass	3.0	[g]
Thermite		
Composition	Copper oxide & Magnesium	[-]
Mass composition	1.155/0.345	[g]
Mass	1.5	[g]

17.3. Overview of Experimental Data Solid Rocket Motors

As previously explained, 2 datasets were provided by DARE. The first dataset contains 8 experiments using the DARE BEM solid rocket motor, and the second contains 12 experiments, of which 10 were performed on the SRM 2020, and 2 on the SRM 2020 EE6. The data itself has been stored in .tdsm files and can not be opened or used directly.

17.3.1. Experimental Data File Structure

As previously mentioned, the data obtained during the test can not be directly accessed or used. The data is stored inside .tdsm files, which need special processing to be used. Tdsm stands for Technical Data Management Streaming and is a file which stores data in a hierarchical structure [89]. It is mostly used by the computer program LabVIEW to store multiple sensor outputs in a single file. A general example of the structure of a .tdsm file is presented in Figure 17.1.

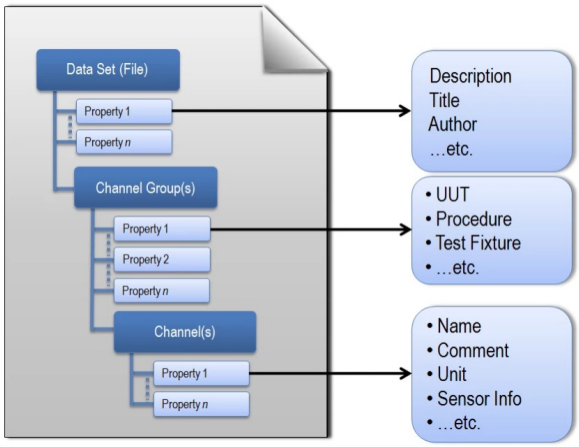


Figure 17.1: Typical structure of a .tdms file [89]

To open a .tdsm file, use has been made of the Python library npTDMS, which opens .tdms files as NumPy arrays. Inside the data files, voltages or currents, measured by the sensors during the experiment, are stored. Before these quantities can be converted, a calibration should first be performed to convert the voltages or currents to the correct measured quantity. During the two tests, the thrust force generated by the SRM and the chamber pressure are being measured, of which the sensors store the current and voltage of these measurements, rather than the converted pressure and thrust. The calibration for the load cell, which is being used to measure the thrust force, has been performed by attaching known masses to the load cell and storing the measured voltage reading inside a data file. For the pressure sensor, used to measure the chamber pressure, a similar process has been followed, but instead of making use of masses, use has been made of known pressures. These calibration measurements have been stored in the data file Calibration_211217_1050571.tdms. This data file is valid for all experiments conducted during the year 2021. For the experiments conducted in the years 2023 and 2024, the conversion formulas have been provided by DARE.

17.3.2. Determination of Conversion Formulas

The calibration file Calibration_211217_1050571.tdms contains 3 different types of data, where for each data type, an FFT has also been provided. The measurements contain information from a load cell, a pressure sensor and a sound sensor for calibration of the FFT domain. This results in a total of 6 different data packages, which are summarised in the list below:

- Load cell measurements in Volts
- Load cell measurements FFT
- Pressure sensor measurements in Amperes
- Pressure sensor measurements FFT

- Sound sensor measurements in Amperes
- Sound sensor measurements FFT

Unfortunately, no data for time or sampling frequency is provided for these frequencies. To obtain the sampling frequency, use can be made of the 15 [kHz] sound measurement. Looking at the FFT of the sound measurement, the frequency can be tuned until the peak at 15 [kHz] overlaps with the 15 [kHz] of the FFT frequency spectrum. This can be done by dividing the total length of the FFT measurements in such a way that the 15 [kHz] tone overlaps with the 15 [kHz] frequency. This number should also be used as the distance between each measurement. Making use of an algorithm which finds the ideal match, this number is found to be 105.35690099998257. It is assumed that this correction can and has to be applied to the load cell and weight frequencies as well. After the 15 [kHz] signal has been shifted to the correct location, the sampling frequency of each measurement can now be obtained by making use of the frequencies present in the FFT plot. The sampling frequency equals twice the maximum frequency in the FFT plot, as was explained in section 2.2. Making use of the sampling frequency, the time can now be obtained, using the maximum length of the data samples in the time domain, divided by the sampling frequency, where the distance between each step is the inverse of the sampling frequency. This results in the raw data of the calibration file as is presented in Figure 17.2.

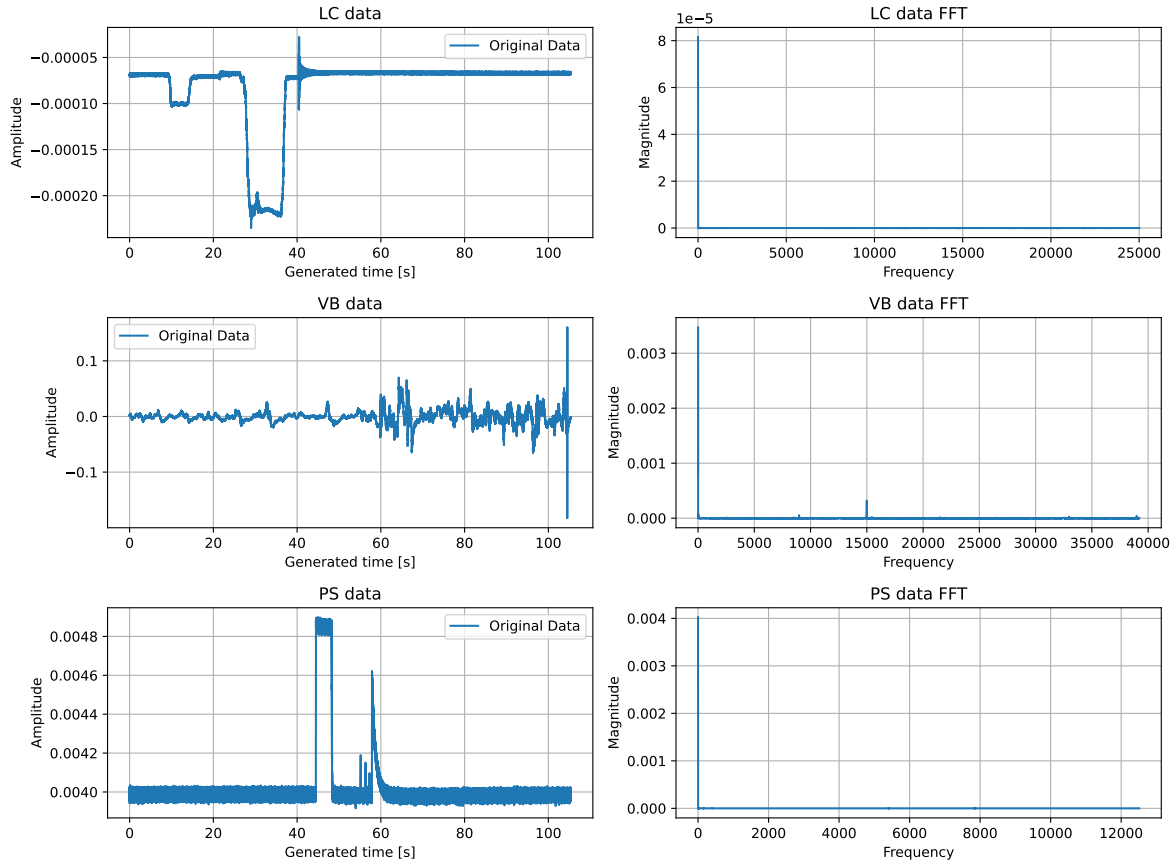


Figure 17.2: Raw data visualized of the Calibration_211217_1050571.tdsm file

The raw data can be used to determine the conversion formulas for the load cell and the pressure sensor. The raw data is used rather than the filtered data since the raw thrust and pressure data will be converted directly to thrust and pressure data, whereafter they will be filtered. By filtering the calibration data, the conversion will be performed, making use of a filtered signal rather than the raw signal, which could lead to inconsistencies. To obtain the required calibration formula, the procedure will be as follows. From Figure 17.2, the plateaus of the calibration measurements will be averaged between certain time steps, a process which is presented in Figure 17.3. Then, the average value of the plateaus will be

Table 17.3: Averaged values for Voltage and Amperes and corresponding quantities

Voltage	Amperes	Value
-6.880890167593203e-05	-	0 [g]
-0.00010038211684431318	-	3939 [g]
-0.00021719115041873271	-	17593 [g]
-7.197854276579282e-05	-	0 [g]
-	0.003988776134829559	0 [bar]
-	0.0048518767568728905	7 [bar]

considered as a single measurement point and used as x-value, while the corresponding values for the load and pressure will be used as y-value, as is presented in Table 17.3. Next, once all plateaus have been averaged, set as x-values and are given their corresponding y-values, a line will be obtained using least squares regression. This results in a plot showing the data points, as well as the regression line, which is presented in Figure 17.4.

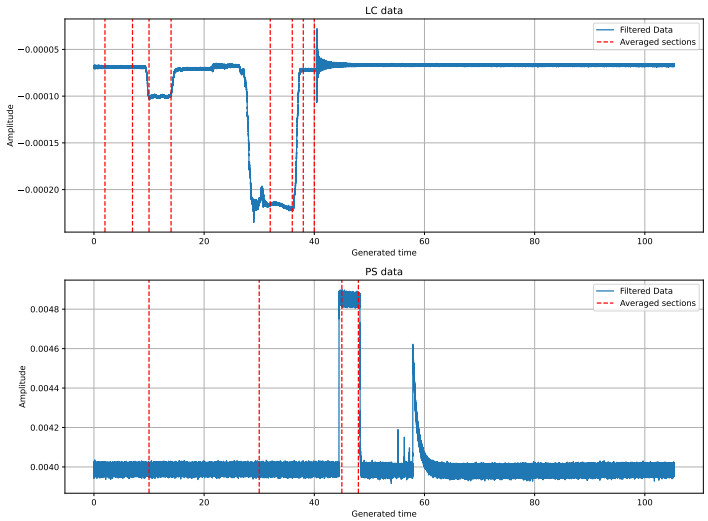


Figure 17.3: Sections of which data will be averaged from Calibration_211217_1050571.tdsm file to generate the transformation functions

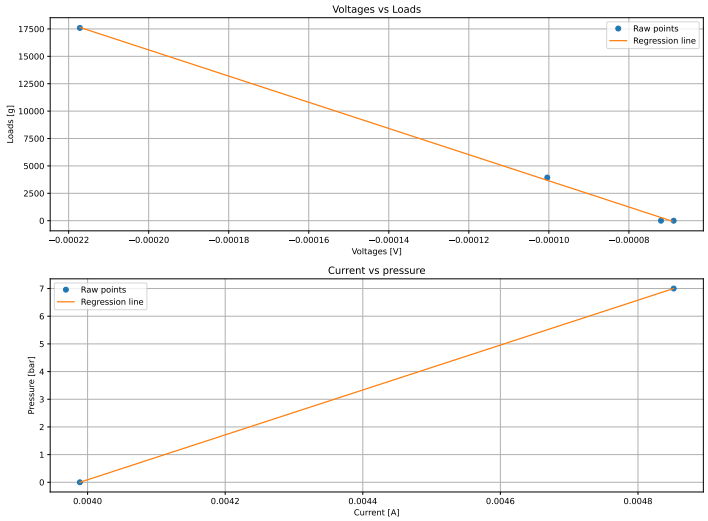


Figure 17.4: Regression line for load cell and pressure sensor calibration

This procedure results in the transformation equations from the calibration file Calibration_211217_1050571.tdms, valid for the experiments for the 2021 data package, for the weight and pressure as presented in Equation 17.1.

$$\begin{aligned} \text{Load [g]} &= -119469466.73812306 \cdot V_{\text{measured}} - 8307.02745026888 \\ \text{Pressure [bar]} &= 8110.294235946666 \cdot I_{\text{measured}} - 32.350148094789816 \end{aligned} \tag{17.1}$$

It should be noted that the equation for load in Equation 17.1 is provided in grams rather than in Newtons. Therefore, the result should be converted to Newtons by converting the grams to kilograms and dividing the final result by the gravitational constant at the Earth's surface, g_0. The conversion formulas for the data packages for 2023 and 2024 are provided in Equation 17.2 and Equation 17.3. It should be noted that these formulas for the load cell have been provided in Newtons.

$$\begin{aligned} \text{Load [N]} &= -1087254 \cdot V_{\text{measured}} - 63.2972 \\ \text{Pressure [bar]} &= 6250 \cdot I_{\text{measured}} - 24 \end{aligned} \tag{17.2}$$

$$\begin{aligned} \text{Load [N]} &= -1.1937 \cdot 10^6 \cdot V_{\text{measured}} - 1040.8 \\ \text{Pressure [bar]} &= 6250 \cdot I_{\text{measured}} - 24 \end{aligned} \tag{17.3}$$

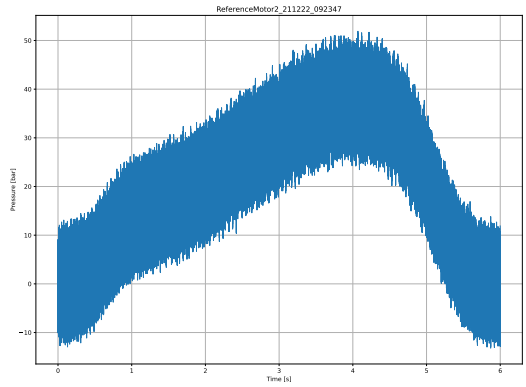
The sampling frequencies for the experimental data packages are presented in Table 17.4.

Table 17.4: Sampling frequencies of the data packages

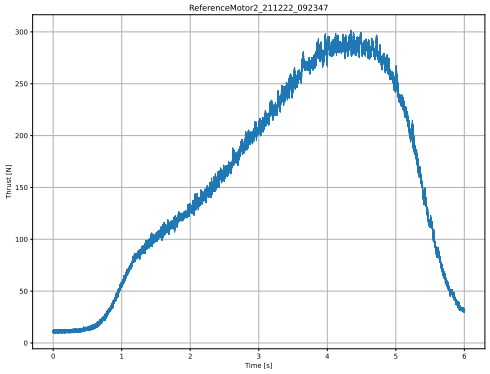
Type of Measurement	Sampling Frequency [Hz]
2021	
Load cell	50010
Pressure sensor	25024
2023	
Load cell	4000
Pressure sensor	4000
2024	
Load cell	2000
Pressure sensor	2000

17.3.3. Noise Removal Strategy

All of the experimental data measurements in the various packages contain noise. To enhance the training process of the PINN and to make an assessment of the PINN compared to experimental data and the numerical model, it is useful to reduce the noise as much as possible. To do so, use can be made of one of the filters explained in section 2.2. However, not all of the filters are suitable for a certain task. To determine the optimal filter for the data processing of the experimental data of the solid rocket motor tests, the file ReferenceMotor2_211222_092347.tdms will be used to determine the most suitable filter. Before the filtering process, the raw data has been converted to thrust and pressure by making use of the specific calibration formulas, as presented in subsection 17.3.2. After this, the data will be filtered, making use of one of the filters of section 2.2. The data will then be processed such that the start of the burn profile will be set at zero seconds, and the end of the burn will be the last data point. This removes unnecessary signals from the start of the signal as well as the end of the signal. Since the pressure and thrust data are not exactly on the same timescale, interpolation is used such that the pressure and thrust data have the same number of data points. This aids the training process of the PINN later on. The raw data for the signal of ReferenceMotor2_211222_092347.tdms is presented in Figure 17.5.



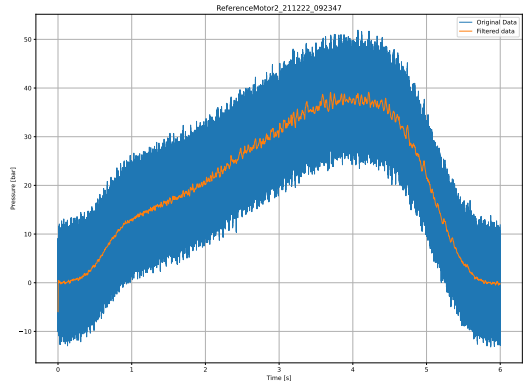
(a) Raw experimental pressure data of the ReferenceMotor2_211222_092347.tdms file



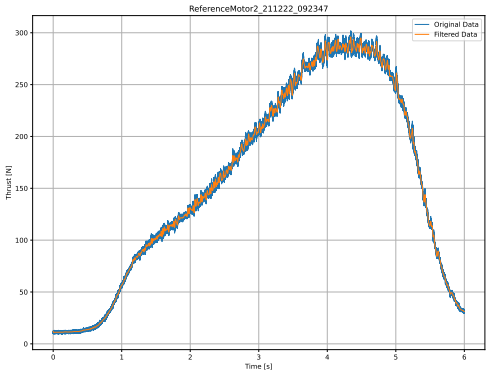
(b) Raw experimental thrust data of the ReferenceMotor2_211222_092347.tdms file

Figure 17.5: Raw experimental data of the ReferenceMotor2_211222_092347.tdms file

To decide which filtering method is the most suitable, use will be made of the mean and standard deviations of the time-corrected signals and the filtered signals, as well as their shapes. The closer the mean, standard deviation and shape are to the unfiltered signal, the more the filtered data resembles the raw data. First, the data is filtered by making use of a Butterworth low-pass filter. It should be noted that the cut-off frequency for the Butterworth filter is set in such a way that the mean and standard deviation of the filtered signal are close to the raw data. The order is set to 3 since it was found in section 2.2 that a typical order for propulsion test data is between 3 and 6. A 3rd order filter has been chosen because this will still add some higher frequency components to the signal. Although this procedure provides reasonable results, the position of the cut-off frequency seems too generic. It can not be determined if higher frequency components above the cut-off frequency are important components of the signal or if they are noise, using this method, meaning that these will be removed from the filtered signal. This could lead to the removal of important information, too. Furthermore, there are some leftover filter residuals present in the signal which do not correspond to the original signal. Therefore, it was decided that the Butterworth filter was not the most suitable method to filter the experimental data. An example of the filtered data signal using this method is presented in Figure 17.6.



(a) Filtered experimental pressure data of the ReferenceMotor2_211222_092347.tdms file, making use of a low-pass Butterworth filter

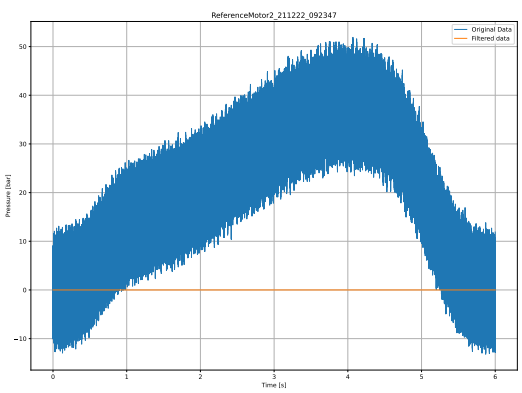


(b) Filtered experimental thrust data of the ReferenceMotor2_211222_092347.tdms file, making use of a low-pass Butterworth filter

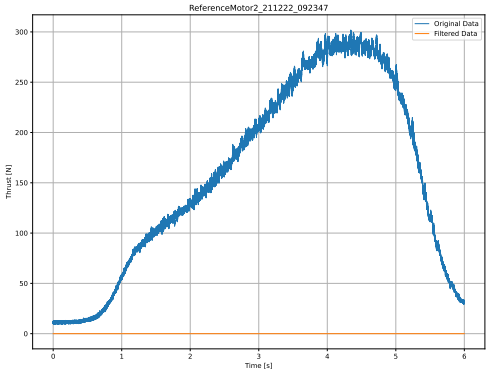
Figure 17.6: Filtered experimental data of the ReferenceMotor2_211222_092347.tdms file, making use of a low-pass Butterworth filter

Making use of the filtering method, which uses Spearman's/Pearson correlation to determine a certain threshold, seems to overcome the issue of important frequency removal, being able to recognise the

higher frequency components that could still be part of the signal. Unfortunately, although this method seems very suitable for making use of experimental data as explained in section 2.2, the filter has been found to provide insufficient results. The filter results in a horizontal line as output, rather than filtered data. Besides this, it was found that this method is hard to generalise, meaning that the filter parameters can not be made generic to be able to work for all signals. Therefore, this filtering method does not seem to be the most suitable option to filter the experimental data. An example of the filtered data using Spearman's/Pearson correlation is presented in Figure 17.7.



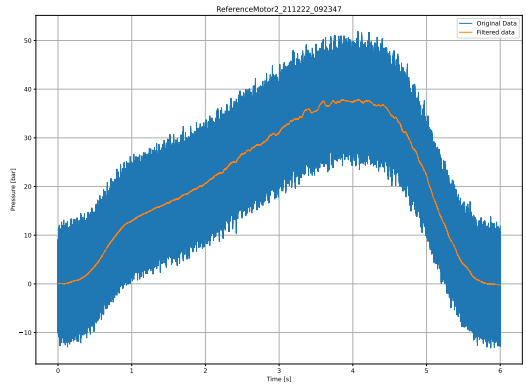
(a) Filtered experimental pressure data of the ReferenceMotor2_211222_092347.tdms file, making use of a Spearman's/Parson cross-correlation filter



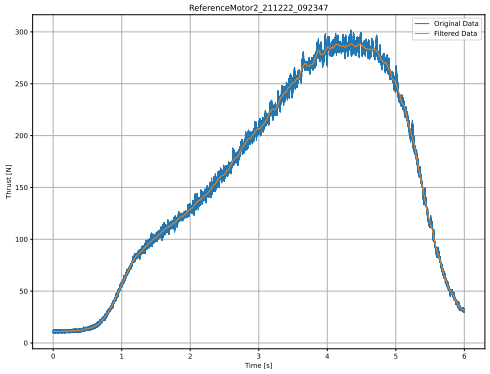
(b) Filtered experimental thrust data of the ReferenceMotor2_211222_092347.tdms file, making use of a Spearman's/Parson cross-correlation filter

Figure 17.7: Filtered experimental data of the ReferenceMotor2_211222_092347.tdms file, making use of a Spearman's/Parson cross-correlation filter

The last option to be considered, the Savitzky-Golay filter, averages out the signal, making use of a filtering window and a polynomial, as was explained in section 2.2. The order of the polynomial is set at 3 to be able to capture peaks in the signal. The window length is set in such a way that it will be closest to the standard deviation and mean. It turns out that the window length needs to be set at approximately 7550 data points. Unfortunately, the window length also turns out to be quite generic. There is no general rule as to why a certain window length or polynomial is the most suitable for a certain signal. However, since the filter only smoothens out the signal but does not necessarily remove high-frequency components of the signal, it was decided to make use of the Savitzky-Golay filtering method to filter the experimental data. It should be noted that for each calibration batch (in other words, the data that belongs to a certain calibration formula), the window length will be determined by making use of the mean and standard deviation. An example of the filtered signals is presented in Figure 17.8.



(a) Filtered experimental pressure data of the ReferenceMotor2_211222_092347.tdms file, making use of a Savitzky-Golay filter



(b) Filtered experimental thrust data of the ReferenceMotor2_211222_092347.tdms file, making use of a Savitzky-Golay filter

Figure 17.8: Filtered experimental data of the ReferenceMotor2_211222_092347.tdms file, making use of a Savitzky-Golay filter

The mean and standard deviation values for the various filtering methods are presented in Table 17.5.

Table 17.5: Mean and standard deviation of various filters and the raw experimental data of the ReferenceMotor2_211222_092347.tdms file for Thrust

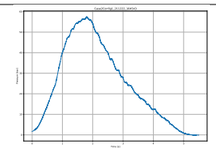
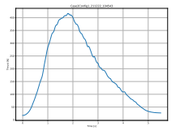
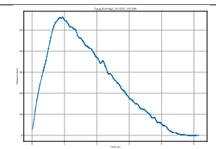
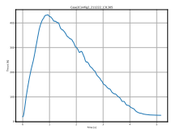
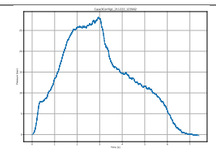
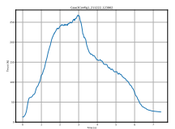
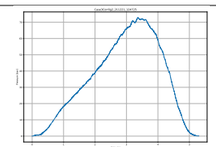
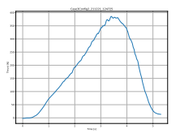
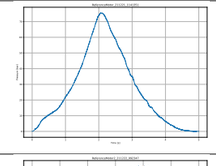
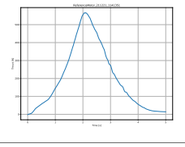
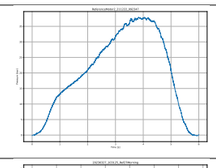
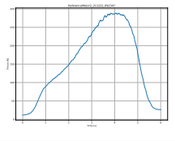
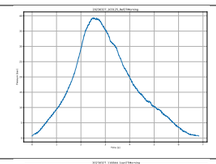
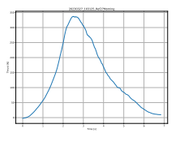
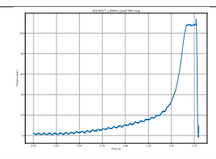
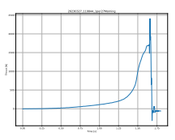
Mean	Standard Deviation
Raw experimental data	
156.87251398164906	94.58070435824884
Butterworth Filter	
156.8725666743039	94.56021192711172
Spearman's/Pearson cross-correlation filter	
0.0	0.0
Savitzky-Golay filter	
156.87250070109698	94.52019097181055

17.3.4. DARE BEM Experimental Data

In Table 17.6, the details of the dataset obtained from the DARE BEM experiments are presented. In total, there are 8 experimental datasets, of which 1 dataset recorded a failure. The calibration formulas are provided in Equation 17.1 and Equation 17.2. Enlarged versions of the experimental data figures are presented in Appendix F, which can be accessed by pressing on the figure in the table[1].

[1] This is only possible in case this thesis is being accessed in a PDF viewer

Table 17.6: AE4S01P data overview

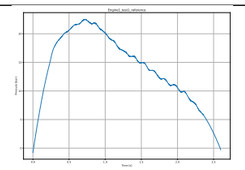
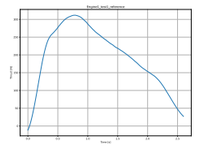
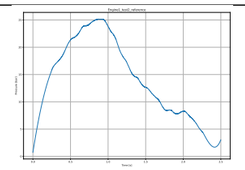
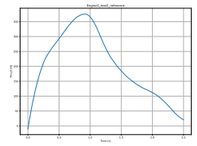
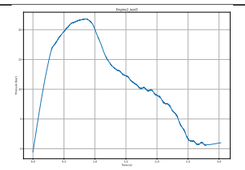
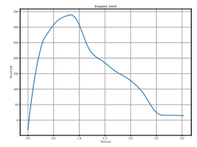
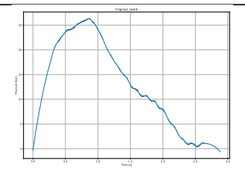
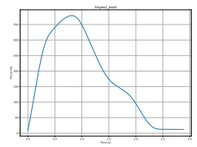
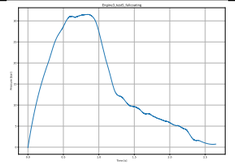
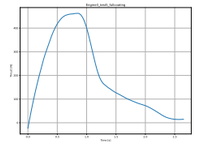
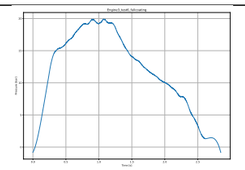
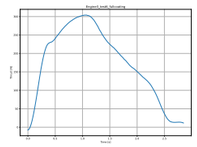
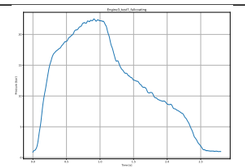
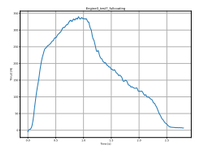
File name	Description	Calibration file		Window length	Amount of Data Points	Pressure figure	Thrust figure
Case2Config1_211222_104543.tdms	Config V grain uninhibited, Standard igniter (Figure E.3), reference nozzle (Table 17.1)	Calibration_211217_1050571.tdms Equation 17.1		7550	275058		
Case2Config2_211222_131305.tdms	Config VI grain Bates, Standard igniter (Figure E.3), reference nozzle (Table 17.1)	Calibration_211217_1050571.tdms Equation 17.1		7550	257554		
Case3Config1_211222_123902.tdms	Reference grain inhibited, Standard igniter (Figure E.3), larger throat nozzle (Table 17.1)	Calibration_211217_1050571.tdms Equation 17.1		7550	370078		
Case3Config2_211221_124725.tdms	Reference grain inhibited, Standard igniter (Figure E.3), smaller throat nozzle (Table 17.1)	Calibration_211217_1050571.tdms Equation 17.1		7550	265057		
ReferenceMotor_211221_1141351.tdms	Config V grain uninhibited, Standard igniter (Figure E.3), reference nozzle (Table 17.1)	Calibration_211217_1050571.tdms Equation 17.1		7550	250052		
ReferenceMotor2_211222_092347.tdms	Reference grain inhibited, Standard igniter (Figure E.3), reference nozzle (Table 17.1)	Calibration_211217_1050571.tdms Equation 17.1		7550	300064		
20230327_103125_Ref27Morning.tdms	Config V grain uninhibited, Standard igniter (Figure E.3), reference nozzle (Table 17.1)	Equation 17.2		1000	27319		
20230327_110844_1per27Morning.tdms	(Failure) Reference grain inhibited with 1% red iron oxide additive, Standard igniter (Figure E.3), reference nozzle (Table 17.1)	Equation 17.2		100	7199		

17.3.5. SRM2020 Experimental Data

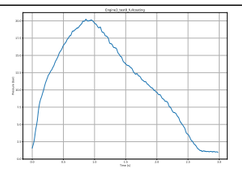
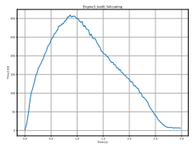
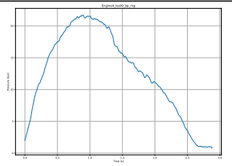
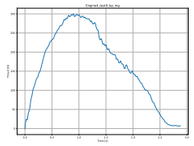
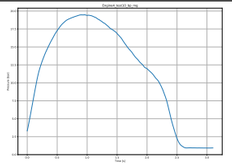
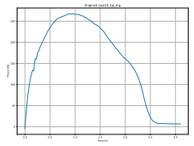
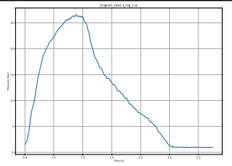
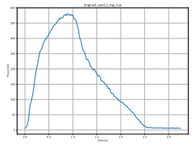
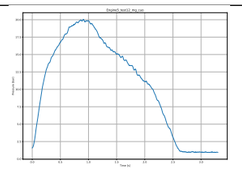
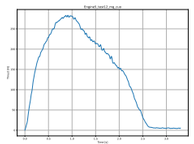
In Table 17.7, the details of the measured data of the SRM 2020 and SRM 2020 EE6 are presented. In total, there are 12 datasets. The calibration formula is provided in Equation 17.3. Enlarged versions of the experimental data figures are presented in Appendix H, which can be accessed by pressing on the figure in the table[2].

[2]This is only possible in case this thesis is being accessed in a PDF viewer

Table 17.7: AE4897 data overview

File name	Description	Calibration file		Window Length	Amount of Data Points	Pressure figure	Thrust figure
Engine1_test1_reference.tdms	Standard configuration, Standard igniter (Table 17.2), 2 grains fully coated, 2 grains only faces coated	Equation 17.3		950	5199		
Engine1_test2_reference.tdms	Standard configuration, Standard igniter (Table 17.2), 2 grains fully coated, 2 grains only faces coated	Equation 17.3		950	4999		
Engine2_test3.tdms	EE6 configuration, Standard igniter (Table 17.2), 4 grains only cores coated	Equation 17.3		950	6058		
Engine2_test4.tdms	EE6 configuration, Standard igniter (Table 17.2), 4 grains only cores coated	Equation 17.3		950	5768		
Engine3_test5_fullcoating.tdms	Standard configuration, Standard igniter (Table 17.2), 4 grains fully coated	Equation 17.3		950	5299		
Engine3_test6_fullcoating.tdms	Standard configuration, Standard igniter (Table 17.2), 4 grains fully coated	Equation 17.3		950	5699		
Engine3_test7_fullcoating.tdms	Standard configuration, Standard igniter (Table 17.2), 4 grains fully coated	Equation 17.3		100	5599		

Continued on next page

File name	Description	Calibration file		Window Length	Amount of Data Points	Pressure figure	Thrust figure
Engine3_test8_fullcoating.tdms	Standard configuration, Standard igniter (Table 17.2), 4 grains fully coated	Equation 17.3		100	5959		
Engine4_test9_bp_mg.tdms	Standard configuration, Black powder & Magnesium igniter (Table 17.2), 4 grains fully coated	Equation 17.3		100	5759		
Engine4_test10_bp_mg.tdms	Standard configuration, Black powder & Magnesium igniter (Table 17.2), 4 grains fully coated	Equation 17.3		600	6199		
Engine5_test11_mg_cuo.tdms	Standard configuration, Thermite igniter (Table 17.2), 4 grains fully coated	Equation 17.3		100	6497		
Engine5_test12_mg_cuo.tdms	Standard configuration, Thermite igniter (Table 17.2), 4 grains fully coated	Equation 17.3		100	6599		

17.4. Data Partitioning Strategy Solid Rocket Motors

From subsection 17.3.4 and subsection 17.3.5, it can be seen that a total of 20 tests have been recorded. From this data, 1 test existed out of a failure event. It was concluded that this test could not be used for training, validation, or testing purposes in this thesis. However, it could be used for future testing to see if a trained model can predict a failure event, an idea which was explored in section 16.7. In general, the following design variables are changing within the experimental datasets:

- 4 different nozzle sizes (8.37/16.73 [mm], 7/14 [mm], 9.57/19.14 [mm], 11/22 [mm])
- 3 different grain designs (sides inhibited, nothing inhibited, bates)
- 3 different chamber lengths (118.5 [mm], 275 [mm], 409 [mm])
- 2 different chamber diameters (80 [mm], 44 [mm])
- 3 different igniter types (Black powder, Blackpowder + Magnesium, Thermite)
- 4 different coating strategies (2 full + 2 faces, 4 full, 4 cores only, none)
- 3 different number of grains (1, 2, 4)

Given the 3 different types of SRMs and all their varying configurations, there can be various ways of separating and training the data to be able to assess the performance of the PINN. If it is required to model a specific motor, it might be beneficial to only make use of the data specifically for that type of motor. However, this thesis investigates PINNs to see if they can be used for surrogate modelling. To do so, all the data of the various engines will be used, with varying input parameters, to be able to train the PINN for the various SRM configurations. To determine the final accuracy of the model, the model should be trained and validated using different datasets or types. For each type, the overall accuracy shall be determined, such that, after all the data has been used as either train or test data, the overall accuracy can be determined by averaging out the individual obtained accuracies, as is explained in subsubsection 2.1.1.12.

To divide the data packages to investigate the performance of the PINN, a single engine test data file will be used as either training or testing data. This means that the data will not be mixed, such that parts of a certain data file will not unintentionally end up in other groups than it was supposed to. Firstly, the 19 data sets will be used to identify if the to-be-developed PINN can capture the general trend by making use of the LOO-CV process. During this process, 18 data sets will be used for training, while 1 will be used to test the performance. Then, the process will be repeated but with another test file out of the total of 19 data sets. This process will be performed by making use of a certain number of data points from each test file to investigate the data efficiency of the various modelling methods. After this process has been completed, the ability of the PINN to operate as a surrogate model will be investigated. To do so, the 19 data files will be separated according to various groups, as was previously explained. With the first test, the data will be partitioned according to nozzle sizes, for which the partitioning of the data files is presented in Table 17.8.

Table 17.8: Data partitioning strategy for the various nozzle sizes

8.37/16.73 [mm]	7/14 [mm]	9.57/19.14 [mm]	11/22 [mm]
Case2Config1_211222_104543.tdms	Case3Config2_211221_124725.tdms	Case3Config1_211222_123902.tdms	Engine1_test1_reference.tdms
Case2Config2_211222_131305.tdms			Engine1_test2_reference.tdms
ReferenceMotor_211221_1141351.tdms			Engine2_test3.tdms
ReferenceMotor2_211222_092347.tdms			Engine2_test4.tdms
20230327_103125_Ref27Morning.tdms			Engine3_test5_fullcoating.tdms
			Engine3_test6_fullcoating.tdms
			Engine3_test7_fullcoating.tdms
			Engine3_test8_fullcoating.tdms
			Engine4_test9_bp_mg.tdms
			Engine4_test10_bp_mg.tdms
			Engine5_test11_mg_cuo.tdms
			Engine5_test12_mg_cuo.tdms

Next, the data will be partitioned according to grain design, as is presented in Table 17.9.

Table 17.9: Data partitioning strategy for the various grain designs

Sides inhibited	Nothing inhibited	Bates
Case3Config1_211222_123902.tdms	Case2Config1_211222_104543.tdms	Case2Config2_211222_131305.tdms
Case3Config2_211221_124725.tdms	ReferenceMotor_211221_1141351.tdms	Engine1_test1_reference.tdms
ReferenceMotor2_211222_092347.tdms	20230327_103125_Ref27Morning.tdms	Engine1_test2_reference.tdms
		Engine2_test3.tdms
		Engine2_test4.tdms
		Engine3_test5_fullcoating.tdms
		Engine3_test6_fullcoating.tdms
		Engine3_test7_fullcoating.tdms
		Engine3_test8_fullcoating.tdms
		Engine4_test9_bp_mg.tdms
		Engine4_test10_bp_mg.tdms
		Engine5_test11_mg_cuo.tdms
		Engine5_test12_mg_cuo.tdms

After this, the data will be partitioned according to chamber length, which is presented in Table 17.10.

Table 17.10: Data partitioning strategy for the various chamber lengths

118.5 [mm]	275 [mm]	409 [mm]
Case2Config1_211222_104543.tdms	Engine1_test1_reference.tdms	Engine2_test3.tdms
Case2Config2_211222_131305.tdms	Engine1_test2_reference.tdms	Engine2_test4.tdms
Case3Config1_211222_123902.tdms	Engine3_test5_fullcoating.tdms	
Case3Config2_211221_124725.tdms	Engine3_test6_fullcoating.tdms	
ReferenceMotor_211221_1141351.tdms	Engine3_test7_fullcoating.tdms	
ReferenceMotor2_211222_092347.tdms	Engine3_test8_fullcoating.tdms	
20230327_103125_Ref27Morning.tdms	Engine4_test9_bp_mg.tdms	
	Engine4_test10_bp_mg.tdms	
	Engine5_test11_mg_cuo.tdms	
	Engine5_test12_mg_cuo.tdms	

Next, the data will be partitioned according to chamber diameter, which is presented in Table 17.11.

Table 17.11: Data partitioning strategy for the various chamber diameters

80 [mm]	44 [mm]
Case2Config1_211222_104543.tdms	Engine1_test1_reference.tdms
Case2Config2_211222_131305.tdms	Engine1_test2_reference.tdms
Case3Config1_211222_123902.tdms	Engine2_test3.tdms
Case3Config2_211221_124725.tdms	Engine2_test4.tdms
ReferenceMotor_211221_1141351.tdms	Engine3_test5_fullcoating.tdms
ReferenceMotor2_211222_092347.tdms	Engine3_test6_fullcoating.tdms
20230327_103125_Ref27Morning.tdms	Engine3_test7_fullcoating.tdms
	Engine3_test8_fullcoating.tdms
	Engine4_test9_bp_mg.tdms
	Engine4_test10_bp_mg.tdms
	Engine5_test11_mg_cuo.tdms
	Engine5_test12_mg_cuo.tdms

After this, the data will be partitioned according to igniter type, as is presented in Table 17.12.

Table 17.12: Data partitioning strategy for the various igniter types

Black powder	Blackpowder + Magnesium	Thermite
Case2Config1_211222_104543.tdms	Engine4_test9_bp_mg.tdms	Engine5_test11_mg_cuo.tdms
Case2Config2_211222_131305.tdms	Engine4_test10_bp_mg.tdms	Engine5_test12_mg_cuo.tdms
Case3Config1_211222_123902.tdms		
Case3Config2_211221_124725.tdms		
ReferenceMotor_211221_1141351.tdms		
ReferenceMotor2_211222_092347.tdms		
20230327_103125_Ref27Morning.tdms		
Engine1_test1_reference.tdms		
Engine1_test2_reference.tdms		
Engine2_test3.tdms		
Engine2_test4.tdms		
Engine3_test5_fullcoating.tdms		
Engine3_test6_fullcoating.tdms		
Engine3_test7_fullcoating.tdms		
Engine3_test8_fullcoating.tdms		

Next, the data will be partitioned according to coating strategy, which is presented in Table 17.13.

Table 17.13: Data partitioning strategy for the various coating strategies

2 Full + 2 Faces	4 Full	4 Cores Only	None
Engine1_test1_reference.tdms	Engine3_test5_fullcoating.tdms	Engine2_test3.tdms	Case2Config1_211222_104543.tdms
Engine1_test2_reference.tdms	Engine3_test6_fullcoating.tdms	Engine2_test4.tdms	Case2Config2_211222_131305.tdms
	Engine3_test7_fullcoating.tdms		Case3Config1_211222_123902.tdms
	Engine3_test8_fullcoating.tdms		Case3Config2_211221_124725.tdms
	Engine4_test9_bp_mg.tdms		ReferenceMotor_211221_1141351.tdms
	Engine4_test10_bp_mg.tdms		ReferenceMotor2_211222_092347.tdms
	Engine5_test11_mg_cuo.tdms		20230327_103125_Ref27Morning.tdms
	Engine5_test12_mg_cuo.tdms		

Lastly, the data will be partitioned according to the number of grains, which is presented in Table 17.14.

Table 17.14: Data partitioning strategy for the various numbers of grains

1	2	4	6
Case2Config1_211222_104543.tdms	Case2Config2_211222_131305.tdms	Engine1_test1_reference.tdms	Engine2_test3.tdms
Case3Config1_211222_123902.tdms		Engine1_test2_reference.tdms	Engine2_test4.tdms
Case3Config2_211221_124725.tdms		Engine3_test5_fullcoating.tdms	
ReferenceMotor_211221_1141351.tdms		Engine3_test6_fullcoating.tdms	
ReferenceMotor2_211222_092347.tdms		Engine3_test7_fullcoating.tdms	
20230327_103125_Ref27Morning.tdms		Engine3_test8_fullcoating.tdms	
		Engine4_test9_bp_mg.tdms	
		Engine4_test10_bp_mg.tdms	
		Engine5_test11_mg_cuo.tdms	
		Engine5_test12_mg_cuo.tdms	

18

Numerical Model: Solid Rocket Motor

To be able to compare the PINN's accuracy with a traditional numerical modelling method, a numerical simulation model needs to be developed. In this chapter, the numerical model and its development process will be explained. Important equations, as well as assumptions and limitations, will be discussed. First, the underlying physical model will be examined, as well as relevant derivations, limitations and assumptions of the model in section 18.1. Next, the setup of the model and the logic behind the simulation will be explained in section 18.2. After this, the inputs for the model will be discussed in Table 18.1. Next, an example of the output predictions provided by the model will be examined in section 18.4. Lastly, the verification process will be discussed in section 18.5.

18.1. Equations, Assumptions and Limitations of the Numerical Model

The combustion process of a solid rocket motor generates high-temperature combustion gases. Making use of a convergent-divergent nozzle, these high-temperature gases are released with a limited flow rate through the nozzle, resulting in a high-pressure gas which will be expelled at high velocities, resulting in the generation of a thrust force, as was explained in section 16.2. To model this behaviour, use can be made of a set of differential equations, which was previously explained in section 16.5. These differential equations make use of simplifications and assumptions, meaning that they are not valid for all use cases.

18.1.1. Equations Used in the Numerical model

In section 16.6, a physical model in the form of a set of state equations was presented to model the behaviour of a solid rocket motor. The state equations are repeated below in Equation 18.1.

$$\begin{bmatrix} \dot{P}_0 \\ \dot{r} \\ A_{\text{burn}} \\ V_c \end{bmatrix} = \begin{bmatrix} \left(\frac{A_{\text{burn}} \cdot \dot{r}}{V_c}\right) \cdot \left(\rho_p \cdot R_g \cdot T_0 - P_0\right) - \left(\frac{A^*}{V_c}\right) \cdot P_0 \cdot \sqrt{\gamma \cdot R_g \cdot T_0 \cdot \left(\frac{2}{\gamma+1}\right)^{\left(\frac{\gamma+1}{\gamma-1}\right)}} \\ a \cdot P_0^n \\ 2 \cdot \pi \cdot r \cdot L_{\text{port}} \\ \pi \cdot r^2 \cdot L_{\text{port}} \end{bmatrix} \tag{18.1}$$

The first state equation, $\dot{P}_0$, can be derived from the conservation of mass. To do so, use will be made of Figure 18.1, which provides an overview of the combustion chamber of an SRM.

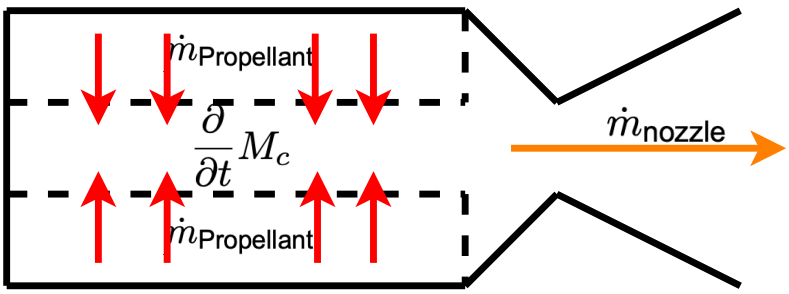


Figure 18.1: Overview of mass flow rate towards and out of the combustion chamber

Making use of the conservation of mass [91], the change in total mass inside the combustion chamber can be obtained, which can be modelled by making use of Equation 18.2

$$\frac{\partial}{\partial t} M_c = \left[\dot{m}_{\text{Propellant}}\right] - \dot{m}_{\text{Nozzle}} \tag{18.2}$$

The total mass inside the combustion chamber can be rewritten as the product of the density of the chamber and the volume of the chamber, as is presented in Equation 18.3

$$\frac{\partial}{\partial t} M_c = \frac{\partial}{\partial t} \left[\rho_c V_c\right] = \frac{\partial}{\partial t} \left[\rho_c\right] V_c + \rho_c \frac{\partial}{\partial t} \left[V_c\right] \tag{18.3}$$

If we assume that the flow through the nozzle chokes immediately at startup, as explained in subsection 16.5.2, the mass flow rate leaving the nozzle can be obtained by making use of Equation 18.4. Equation 18.4 makes use of the same assumptions as provided in subsection 16.5.2.

$$\dot{m}_{\text{Nozzle}} = A^* \sqrt{\frac{\gamma}{R_g} \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}} \frac{P_0}{\sqrt{T_0}} \tag{18.4}$$

Substituting Equation 18.4 and Equation 18.3 into Equation 18.2, results into Equation 18.5

$$\frac{\partial}{\partial t} \left[\rho_c V_c\right] + \rho_c \frac{\partial}{\partial t} \left[V_c\right] = \left[\dot{m}_{\text{Propellant}}\right] - A^* \sqrt{\frac{\gamma}{R_g} \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}} \frac{P_0}{\sqrt{T_0}} \tag{18.5}$$

If the assumption is made that the combustion gases behave like an ideal gas and that the combustion temperature remains constant, Equation 18.6 can be used to rewrite the density of the gas located inside the combustion chamber.

$$\rho_c = \frac{P_0}{R_g T_0} \quad \longrightarrow \quad \frac{\partial}{\partial t} \left[\rho_c\right] = \frac{1}{R_g T_0} \frac{\partial}{\partial t} \left[P_0\right] \tag{18.6}$$

If Equation 18.6 is now substituted into Equation 18.5, will result in Equation 18.7.

$$\frac{\partial P_0}{\partial t} \frac{V_c}{R_g T_0} + \frac{P_0}{R_g T_0} \frac{\partial V_c}{\partial t} = \left[\dot{m}_{\text{Propellant}}\right] - A^* \sqrt{\frac{\gamma}{R_g} \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}} \frac{P_0}{\sqrt{T_0}} \tag{18.7}$$

Rearranging Equation 18.7 and multiply both sides with $\frac{R_g T_0}{V_c}$, results in Equation 18.8.

$$\frac{\partial P_0}{\partial t} + P_0 \frac{1}{V_c}\frac{\partial V_c}{\partial t} + \frac{R_g T_0}{V_c} A^* \sqrt{\frac{\gamma}{R_g}\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}} \frac{P_0}{\sqrt{T_0}} = \frac{R_g T_0}{V_c}\left[\dot{m}_{\text{Propellant}}\right] \tag{18.8}$$

Rearranging Equation 18.8, results in Equation 18.9.

$$\frac{\partial P_0}{\partial t} + P_0 \left[\frac{1}{V_c}\frac{\partial V_c}{\partial t} + \frac{A^*}{V_c}\sqrt{\gamma R_g T\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}}\right] = \frac{R_g T_0}{V_c}\left[\dot{m}_{\text{Propellant}}\right] \tag{18.9}$$

The change in volume over time can be calculated by making use of Equation 18.10.

$$\frac{\partial V_c}{\partial t} = A_{\text{Burn}}\dot{r} \tag{18.10}$$

The propellant mass flow rate going inside the chamber can be obtained by making use of Equation 18.11.

$$\dot{m}_{\text{Propellant}} = \rho_p A_{\text{Burn}}\dot{r} \tag{18.11}$$

Substituting Equation 18.10 and Equation 18.11 in Equation 18.9, results in Equation 18.12, which is the first equation of Equation 18.1.

$$\frac{\partial P_0}{\partial t} = \left(\frac{A_{\text{burn}}\cdot\dot{r}}{V_c}\right)\cdot\left(\rho_p\cdot R_g\cdot T_0 - P_0\right) - \left(\frac{A^*}{V_c}\right)\cdot P_0\cdot\sqrt{\gamma\cdot R_g\cdot T_0\cdot\left(\frac{2}{\gamma+1}\right)^{\left(\frac{\gamma+1}{\gamma-1}\right)}} \tag{18.12}$$

The quantity $\dot{r}$, which represents the regression rate over time, can be modelled by making use of De Vielle's law, as explained in section 16.2. A_{Burn} and V_c, on the other hand, require more attention since these quantities vary for different types of grain geometries and inhibitors. In its simplest form, the grain geometry exists out of a single hollow port, as was presented in Figure 16.4. In case the outside of the grain and the sides are inhibited, the surface burning areas can only be located on the inside of the hollow port, in the radial direction. From geometry, the surface burn area can be calculated by making use of Equation 18.13. The chamber volume, on the other hand, can be calculated assuming that the grain does fill the combustion chamber, from which the volume of the port is subtracted. This is equal to the volume of a cylinder, which can be obtained by making use of Equation 18.14.

$$A_{\text{Burn}} = 2\pi r L_{\text{Port}} \tag{18.13}$$

$$V_c = \pi r^2 L_{\text{Port}} \tag{18.14}$$

Unfortunately, Equation 18.13 and Equation 18.14 do not take into account the propellant grains with a geometry smaller than the chamber geometry. Besides this, grains containing uninhibited sides and Bates grain designs can not be modelled making use of these equations. Therefore, Equation 18.13 and Equation 18.14 need to be adjusted. This is important because some of the experimental data presented in section 17.3 contains these grain designs. First, the equation to determine the burn area will be adjusted. Starting from Equation 18.13, first, if the core will be inhibited, the burn area should be zero. To do so, a variable, uninhibited_core, will be introduced, such that if the core is inhibited, certain parts of the equation will cancel out. Next, if the grain has uninhibited sides, the length of the grain will decrease on both sides with the regression rate. To implement this behaviour, another variable, uninhibited_sides, can be introduced, together with the subtraction of the total regressed rate from the length of the grain. It should be noted that the total regressed rate can be obtained by integration of

$\dot{r}$, from which the initial port radius is subtracted. However, since it is assumed that the grain ends will burn from both sides, this regressed rate should be multiplied by two. This results in a new equation for the burn area, which is presented in Equation 18.15.

$$A_b = \left(\text{uninhibited\_core} \left(2\pi r \left(L_{\text{grain}} - \text{uninhibited\_sides} 2 \left(r - r_{\text{grain\_init}} \right) \right) \right) \right) \tag{18.15}$$

Next, the possibility of modelling the surface burn area on the sides of the grain will be introduced as well, in case the sides of the grain are not inhibited. Lastly, to include the possibility of modelling Bates' grain designs, the burn area for each grain should be multiplied by the total number of grains. It is assumed that all of the grains will burn instantaneously. This results in a final expression to be able to calculate the surface burn area, which is presented in Equation 18.16.

$$\begin{aligned} A_b = &\left(\text{uninhibited\_core} \left(2\pi r \left(L_{\text{grain}} - 2 \text{ uninhibited\_sides} \left(r - r_{\text{grain\_init}} \right) \right) \right) \right) N_{\text{grains}} \\ &+ \left(\text{uninhibited\_sides} \left(2\pi \frac{\left(d_{\text{grain\_out}}^2 - (2r)^2 \right)}{4} \right) \right) N_{\text{grains}} \end{aligned} \tag{18.16}$$

Now that an expression for the surface burn area has been developed, an expression for the chamber volume should be developed as well. Starting from Equation 18.14, the empty volume inside the combustion chamber is included in case the propellant grain has a smaller geometry than the combustion chamber, resulting in additional empty volume. To do so, the total volume inside the chamber is calculated, making use of the chamber diameter and length. Next, the volume of the grain is subtracted from this volume (including the liner thickness). It should be noted that the subtraction of the volume of the propellant grain also includes the empty volume of the grain's port, which is later added to the total chamber volume. This results in an expression as is presented in Equation 18.17.

$$\begin{aligned} V_c = &\left(\pi r^2 L_{\text{grain}} \right) + \left(\pi \frac{d_{\text{chamber}}^2}{4} L_{\text{chamber}} \right) \\ &- \left(\pi \frac{(d_{\text{grain\_out}} + 2t_{\text{liner}})^2}{4} L_{\text{grain}} \right) \end{aligned} \tag{18.17}$$

To include the possibilities for uninhibited sides and Bates grain geometries, Equation 18.17 is adjusted. First, if the sides are uninhibited, additional chamber volume is introduced during the combustion process, but also subtracted from the port length. This can be modelled, making use of the regression rate variable, of which the initial port radius is subtracted. Lastly, to be able to model Bates grain geometries, the empty chamber volumes caused by the regressed propellant grain sides are multiplied by the number of grains. This results in a final expression for the chamber volume as is presented in Equation 18.18.

$$\begin{aligned} V_c = &\left(\text{uninhibited\_core} \left(\pi r^2 \left(L_{\text{grain}} - 2 \text{ uninhibited\_sides} \left(r - r_{\text{grain\_init}} \right) \right) \right) \right. \\ &\left. + \text{uninhibited\_sides} \left(\pi \frac{d_{\text{grain\_out}}^2}{4} 2 \left(r - r_{\text{grain\_init}} \right) \right) \right) N_{\text{grains}} \\ &+ \left(\pi \frac{d_{\text{chamber}}^2}{4} L_{\text{chamber}} \right) - N_{\text{grains}} \left(\pi \frac{(d_{\text{grain\_out}} + 2t_{\text{liner}})^2}{4} L_{\text{grain}} \right) \end{aligned} \tag{18.18}$$

After the expressions for the surface burn area and empty chamber volume have been obtained, the state equations, as presented in Equation 18.1, can be updated. This results in the final state equations, as presented in Equation 18.19.

$$\begin{bmatrix} \dot{P}_0 \\ \dot{r} \\ A_{\text{burn}} \\ V_c \end{bmatrix} = \begin{bmatrix} \left(\frac{A_{\text{burn}} \cdot \dot{r}}{V_c}\right) \cdot (\rho_p \cdot R_g \cdot T_0 - P_0) - \left(\frac{A^*}{V_c}\right) \cdot P_0 \cdot \sqrt{\gamma \cdot R_g \cdot T_0 \cdot \left(\frac{2}{\gamma+1}\right)^{\left(\frac{\gamma+1}{\gamma-1}\right)}} \\ a \cdot P_0^n \\ Equation\ 18.16 \\ Equation\ 18.18 \end{bmatrix} \tag{18.19}$$

To obtain a prediction of the generated thrust, use can be made of Equation 16.12. When making use of Equation 16.7 and Equation 16.11, Equation 16.12 can be rewritten into Equation 18.20.

$$F_t = C_f P_c A_t \tag{18.20}$$

In Equation 18.20, C_f represents the thrust coefficient, P_c represents the predicted chamber pressure obtained from Equation 18.19 and A_t represents the nozzle throat area. Making use of Equation 16.10, it can be seen that, to calculate the thrust coefficient, the ratio between the chamber pressure and nozzle exit pressure needs to be obtained. Since the areas of the nozzle exit and the nozzle throat are known, use can be made of Equation 16.9 to obtain the required pressure ratio. To do so, the area ratio will be subtracted from both sides of Equation 16.9, whereafter a root solving algorithm will be used, where the pressure ratio is used as the only changing variable.

18.1.2. Assumptions of the Numerical Model

The numerical simulation model makes use of several assumptions and simplifications to model the complex physical behaviours which occur during the operation of an SRM. These assumptions are summarised in the list below [67, 91, 137, 80]:

- The model is specifically tailored for SRMs
- The model is a 1-D simulation
- In the model, instantaneous combustion of the propellant grains is assumed
- The model assumes a fixed number of cylindrical grains with a circular hollow core
- The cylindrical grains can only burn in the radial or axial direction
- The cylindrical grains can only have a free or inhibited surface at the core or the sides
- Both of the propellant grain sides will be either free or inhibited
- The throat of the nozzle chokes immediately at start-up
- A constant flame temperature is assumed inside the combustion chamber
- The exhaust gases are homogeneous and have a constant composition
- The gas or gas mixture obeys the ideal gas law
- The heat capacity of the gas or mixture of gases expelled is constant
- The flow through the nozzle is one-dimensional, steady and isentropic

18.1.3. Limitations of the Numerical Model

From subsection 18.1.2, it can be seen that for the development of the numerical simulation, a lot of assumptions have been made. Most importantly, only cylindrical grains can be used. Furthermore, the cylindrical grains can only burn from inside the core and/or at the sides, meaning that the outside of the grain is always inhibited. This means that for this simulation, no other grain geometries can be modelled. In case other grain geometries are used as inputs for this model, unreliable results could be obtained. Next to this, instantaneous combustion is assumed. Considering that not all surfaces

will burn at and have been fully regressed at the same time, this assumption limits the accuracy of the predictability of start-up and trail-off transients. Furthermore, this means that no reliable predictions can be made about variations in burning strategies (for example, some grains burn at an earlier stage compared to others) or ignition delay times. Another limitation is that this numerical model is tailored towards SRM, meaning that this model can not be used for the prediction of liquid or hybrid rocket engines' performances. Lastly, this model needs combustion properties as input. These can only be obtained by making use of dedicated external software such as CEA [84] or literature. However, using software packages to obtain the values will introduce additional assumptions and limitations, meaning that a user should be aware of the limitations of the other software models or values obtained from literature to identify if the obtained inputs are suitable for the developed numerical simulation model.

18.2. Setup of the Numerical Model

Now that the state equations are known, the numerical simulation model can be developed. To present how the pressure changes over time inside the combustion chamber is obtained, implementing the state equations as presented in Equation 18.19 into Python code, a flowchart has been created, which is presented in Figure 18.2.

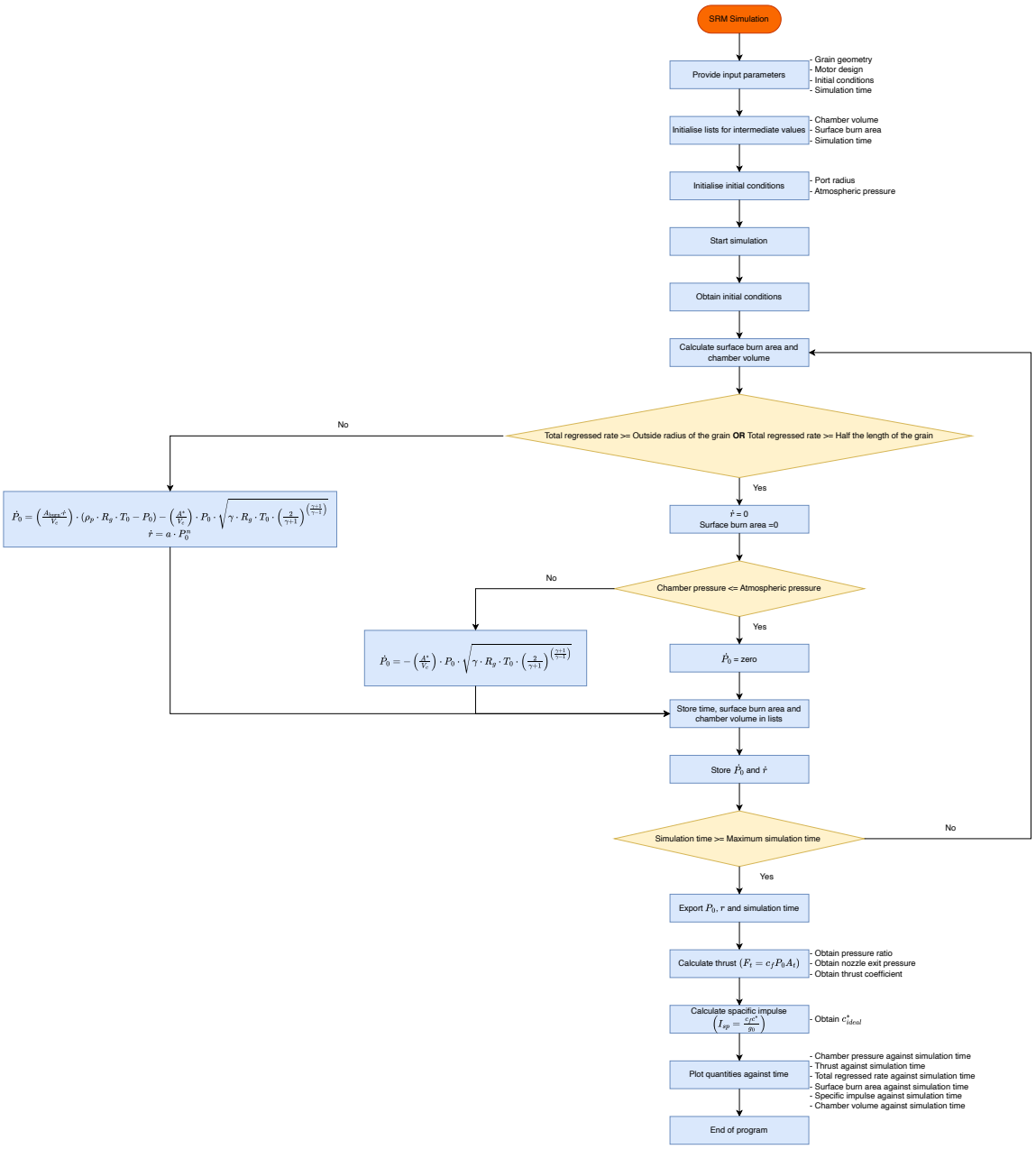


Figure 18.2: Flowchart presenting the structure of the Python code developed for the numerical simulation model for the SRM

The flow diagram as presented in Figure 18.2 has later been implemented in code. The model makes use of a Runge-Kutta 5th-order numerical integration scheme [104]. It should be noted that all the formulas are adjusted such that the provided pressure predictions are in bar and the generated thrust force in Newtons.

18.3. Inputs of the Numerical Model

To be able to obtain predictions of the chamber pressure and generated thrust, making use of the developed numerical model, inputs are required to specify the modelling problem. For the inputs, it is important to understand which quantities need to be provided as inputs and their respective units. The inputs and their units of the developed numerical simulation model for the prediction of chamber pressure and generated thrust force for SRMs are summarised in Table 18.1.

Table 18.1: Summary of inputs for the numerical simulation

Symbol	Meaning	Unit
Constants		
g_0	Gravitational acceleration at sea level	[m/s^2]
R_a	Universal Gas constant	[J/K · mol]
P_a	Standard pressure at sea level	[bar]
Grain Specifications		
L_{grain}	Length of the grain	[m]
$d_{grain_{in}}$	Port diameter of the grain	[m]
t_{liner}	Wall thickness of the liner	[m]
$d_{grain_{out}}$	Outside diameter of grain	[m]
$mass_{grain_{init}}$	Initial mass of the grain	[kg]
N_{grains}	Number of grain segments	[-]
a	Burning rate coefficient	[mm/s - Pa^n]
n	Burning rate exponent	[-]
$uninhibited_{core}$	Exposed core	[-]
$uninhibited_{sides}$	Exposed sides	[-]
Combustion Gas Predictions		
$gamma$	Specific heat of combustion mixture	[-]
W_g	Molecular mass of mixture	[kg/mol]
T_0	Temperature of combustion mixture	[K]
Motor geometry		
$d_{chamber}$	Outside diameter of the chamber	[m]
$L_{chamber}$	Length of the chamber	[m]
d_t	Nozzle throat radius	[m]
d_e	Nozzle exit radius	[m]
t_b	Simulation time	[s]

18.4. Results of the Numerical Model

After all inputs have been specified, the numerical simulation model can provide predictions for chamber pressure and generated thrust. An example prediction output of the numerical simulation model is presented in Figure 18.3.

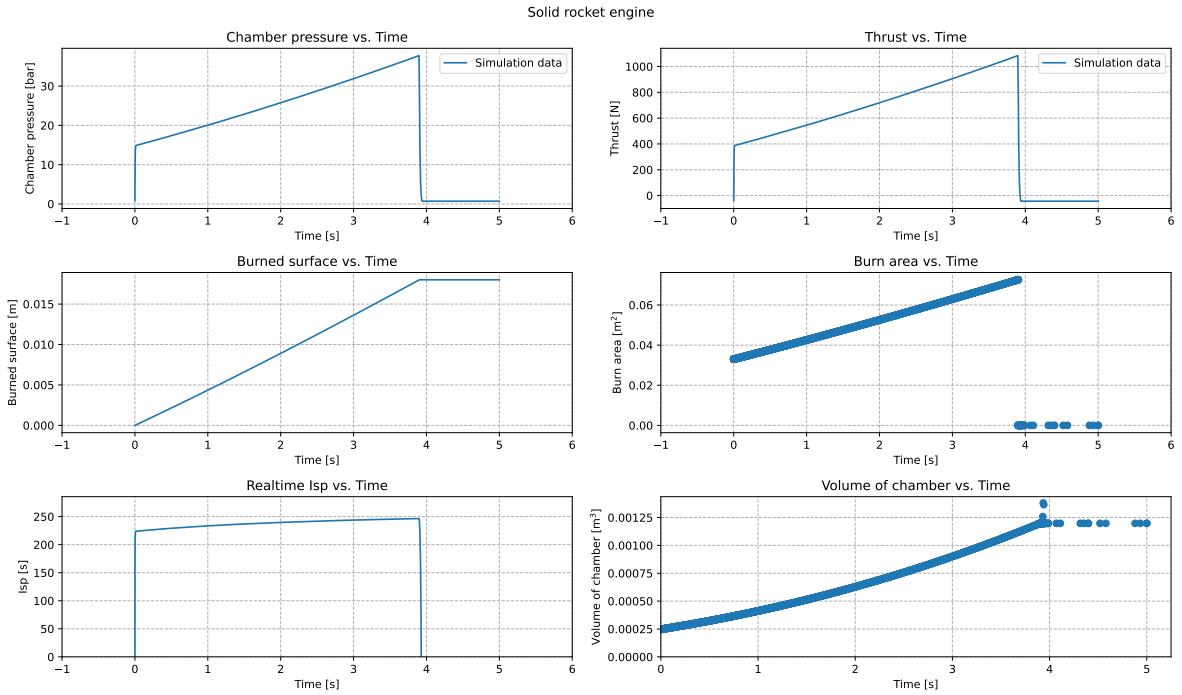


Figure 18.3: Example predictions of chamber pressure and generated thrust, obtained from the developed numerical simulation model in Python code

From Figure 18.3, it can be seen that various quantities are being predicted as an outcome of the simulation model. It can also be seen that some unexpected values appear at the end of the burn for the surface burn area and the chamber volume. It was found that these values do not influence the outcome of the simulation for the predictions of chamber pressure and generated thrust.

18.5. Verification of the Numerical Model

In this chapter, a numerical simulation model has been developed for an SRM. However, it is unknown how well this simulation performs, as well as whether it has been implemented correctly. As explained in subsubsection 2.1.1.15, verification is the process of inspecting if the correct equations have been used, as well as if they have been implemented correctly and how a simulation compares to a similar one which has already been used before [82]. During the literature review, a physical model has been found that can provide predictions of chamber pressure and generated thrust for an SRM. After the implementation of the physical model and its represented state space model in the Python code, the output predictions for chamber pressure and generated thrust were obtained and inspected for flaws. After it was concluded that predicted results were as expected, the code was checked for correct implementation of the state space equations used, as well as relevant stopping conditions and units used for the input quantities. It should be noted that each time the inputs are updated, a verification process has to be performed to make sure that all the inputs have the correct dimensions. After the process of inspecting the code for bugs was completed, it was concluded that there was a small mistake in the implementation of the surface burn area formula, which was corrected afterwards (The results presented in Figure 18.3 were obtained after the code was corrected). After the other parts of the code had been checked, this part of the verification process was completed, and there are currently no indications of implementation errors.

To compare the performance of the developed numerical simulation model against existing SRM simulations, use was made of the output of the SRM simulation software developed by Utah State University, or shortened USU [73]. To do so, two different SRM configurations were provided to the software developed by USU and the developed numerical model, whereafter the outputs predicted by both methods would be compared. The results of this comparison are visually presented in Figure 18.4.

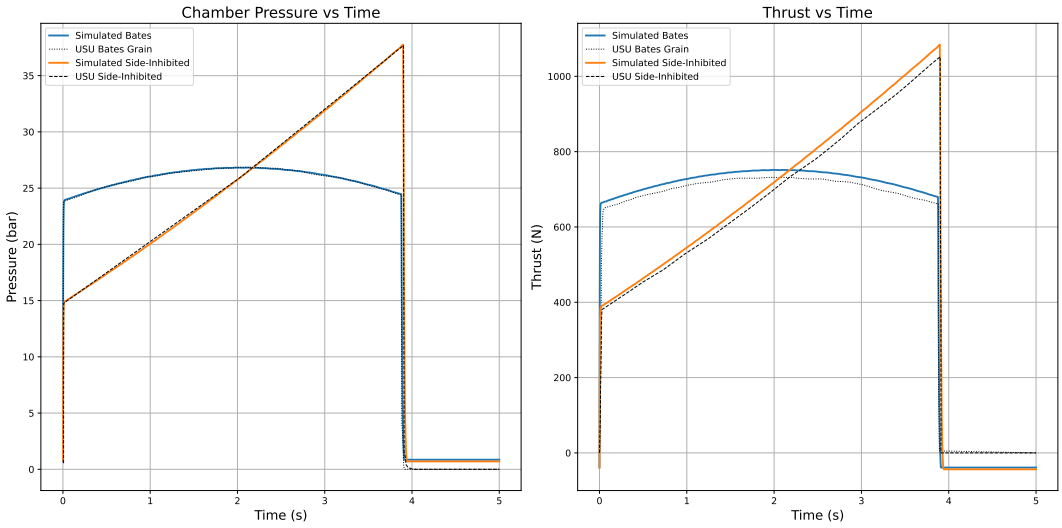


Figure 18.4: Verification run of the developed numerical simulation compared to the outputs of a similar simulation model developed by USU for chamber pressure and generated thrust predictions

From Figure 18.4, it can be seen that there is a good agreement between the chamber pressure predictions of the numerical simulation compared to the simulation model of USU. The predictions of the generated thrust, on the other hand, show some slight deviations between the numerical model and the simulation model of USU, where the numerical model over-predicts the pressure by about 30 [N] at its peak. It was decided, however, that this deviation is within acceptable limits, and therefore, the numerical model is being verified.

19

Physics-Informed Neural Network Model: Solid Rocket Motor

In this part of the report, the performance of a PINN in terms of accuracy, computational and data-efficiency is being investigated against a numerical and data-driven method. Furthermore, the ability of a PINN to become a surrogate model will be examined against these methods. In this chapter, the development of a PINN model for SRM predictions will be discussed. To develop the PINN model, use will be made of the experimental data as presented in chapter 17. Furthermore, the physical model, which was used during the development of the numerical simulation model as presented in chapter 18, will be used. This chapter is structured as follows. First, in section 19.1, the NN architecture of the PINN will be discussed, as well as the used inputs and outputs. Lastly, an example of the predictions of generated thrust and chamber pressure obtained from the PINN model will be presented in section 19.2.

19.1. Setup of the Model

To be able to generate a surrogate PINN, a conditioned PINN should be developed, making use of inputs which generalise the modelling problem, as was explained in subsection 2.1.2. To do so, it was decided to make use of the inputs for the numerical model developed in chapter 18. The inputs of the PINN are summarised in Table 19.1.

Table 19.1: Summary of inputs for the SRM PINN

Symbol	Meaning	Unit
Simulation Settings		
t	Simulation time	[s]
Grain Specifications		
L_{grain}	Length of the grain	[m]
D_{grain_i}	Inner diameter of the grain	[m]
D_{grain_o}	Outer diameter of the grain	[m]
t_{liner}	Thickness of the liner	[m]
m_{grain}	Mass of the grain	[kg]
N_{grains}	Number of grains	[-]
$Uninhibited_{core}$	Exposed grain core (1 = True, 0 = False)	[-]
$Uninhibited_{sides}$	Exposed grain sides (1 = True, 0 = False)	[-]
$-$	Number of coated cores	[-]
a	Burning rate coefficient	[mm/s - Pa^n]
n	burning rate pressure exponent	[-]
Combustion Gas Properties		
γ	Specific heat ratio of combustion gasses	[-]
W_g	Molecular weight of combustion gasses	[kg/mol]
T_0	Adiabatic flame temperature	[$^\circ$ K]
Motor Geometry		
D_{cc}	Diameter of the combustion chamber	[m]
L_{cc}	Length of the combustion chamber	[m]
D_{nozzle_t}	Diameter of the nozzle throat	[m]
D_{nozzle_e}	Diameter of the nozzle exit	[m]
Igniter Specifications		
$-$	Type of igniter (Black powder = 0, Magnesium = 1, Thermite = 2)	[-]

Comparing the inputs of the PINN model to the inputs of the numerical model as presented in section 18.3, it can be seen that almost all of the inputs of the numerical model will be used as inputs of the PINN model. However, some constants, which are an input of the numerical model, will not be used as inputs for the PINN model, since they are assumed to be constant throughout the various modelling problems. Therefore, the PINN will make use of the constants during the training process, but they will not be used as specific inputs of the model, as they do not have to be varied. Furthermore, an additional input has been added, namely the type of igniter. In conclusion, the SRM PINN, which will be developed, will have a total of 20 inputs. Since the PINN will be developed such that it will be able to make predictions of chamber pressure and generated thrust, the PINN will have a total of 2 outputs. This means that, for the architecture of the PINN, the PINN needs to have 20 neurons in its input layer and 2 neurons in its output layer. After the initial testing phase, it was found that 12 hidden layers, each containing 64 neurons, making use of the ELU activation function, would be able to provide satisfactory outputs. An illustration of the PINNs architecture for the prediction of SRM performance in terms of chamber pressure and generated thrust is presented in Figure 19.1.

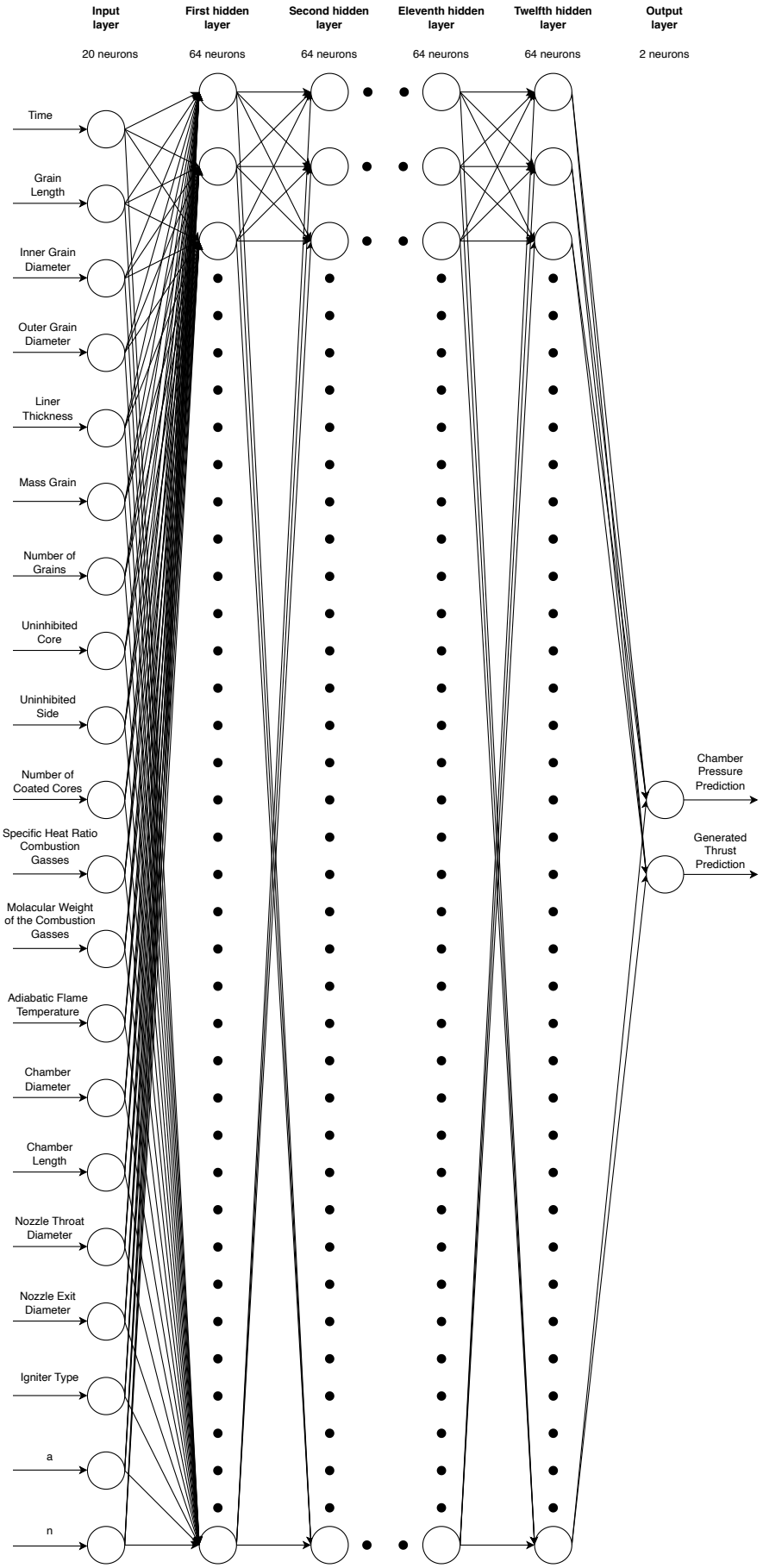


Figure 19.1: Neural Network architecture for the Physics-Informed Neural Network related to the modelling of SRM performance

As can be seen from Figure 13.1, the PINN has a total of 20 inputs. However, after performing some first tests, it was found that the PINN was not able to minimise the cost function satisfactorily, with a low accuracy and high computational time. From subsubsection 2.1.1.11, it was found that when a lot of inputs are being used, having a wide variety of magnitudes, scaling could be used, which aids data-driven methods to more easily capture the trends in the data and be able to converge to a satisfactory solution. After testing various scaling methods, it was found that the 20 inputs need to be scaled by making use of the $log1p$ scaling function. The reason why the $log1p$ has been chosen as a scaling method has to do with the fact that other scaling methods, such as min-max scaling, may cause certain inputs to be rounded to zero. This means that for certain inputs, there would be no differences between the different data sets and their inputs, leading to convergence issues in the NN. To solve this, the $log1p$ was selected to have the inputs larger than zero.

Before the PINN can be implemented, it should be noted that when the PINN is being trained and the differential equations, as presented in Equation 16.15, are being solved, one does not have $\frac{dP_0}{dt}$ directly available from the network, but rather $\frac{dP_0}{dt_{scaled}}$. To be able to compare the scaled version with $\frac{dP_0}{dt}$ calculated by the numerical model from Equation 18.19, one can make use of the chain rule from differential calculus. To do so, $\frac{dP_0}{dt}$ can be rewritten into another form, as is presented in Equation 19.1.

$$\frac{dP_0}{dt} = \frac{dP_0}{dt_{scaled}} \frac{dt_{scaled}}{dt} \tag{19.1}$$

Making use of the fact that $t_{scaled} = \ln(1+t)$, the derivative can be taken with regards to time, which is presented in Equation 19.2.

$$\frac{dt_{scaled}}{dt} = \frac{d\left(\ln(1+t)\right)}{dt} = \frac{1}{(1+t)} \tag{19.2}$$

Substituting Equation 19.2 into Equation 19.1 results in Equation 19.3.

$$\frac{dP_0}{dt} = \frac{dP_0}{dt_{scaled}} \frac{1}{(1+t)} \tag{19.3}$$

Since $\frac{dP_0}{dt_{scaled}}$ is obtained from the network, it was found after testing that transforming $\frac{dP_0}{dt}$ obtained from the numerical model into $\frac{dP_0}{dt_{scaled}}$ is more suitable compared to transferring the scaled version, obtained from the NN, into an unscaled version. The final transformation equation to transform $\frac{dP_0}{dt}$ into $\frac{dP_0}{dt_{scaled}}$ is presented in Equation 19.4.

$$\frac{dP_0}{dt_{scaled}} = \frac{dP_0}{dt}(1+t) \tag{19.4}$$

As was explained in chapter 3, to develop a PINN, a physics and boundary loss need to be added to convert a purely data-driven model into a PINN model. To do so, physics loss functions and boundary conditions are created, making use of the state equations developed for the numerical model as presented in chapter 18. In Code 19.1, the implementation of the physics loss functions for the SRM PINN model are presented. In total, two physics loss functions are presented. The first loss function, called PINN_pressure_loss, is used to compare the derivatives of the PINN with the differential equation for the prediction of chamber pressure. In total, 100 collocation points are being used over the computational domain, which is set to range from $1 \cdot 10^{-16}$ up until 8 [s]. The reason why the computational domain does not start at exactly 0 [s] is that otherwise computational errors will be introduced, causing the cost function of the PINN to go towards infinity. The weight of this loss function for the total cost function is set to $1 \cdot 10^{-6}$. The second physics loss function, called PINN_thrust_loss, compares the predicted generated thrust output with the theoretical thrust output. This loss function is, therefore, different from previously examined loss functions, as it will not calculate and compare derivatives but directly compete with the outputs predicted by the data. This loss function also makes use of 100 collocation points, using the same computational domain as used in the PINN_pressure_loss function.

The weight of this loss function in the total cost function is set to $1 \cdot 10^{-3}$. The boundary loss functions used in the SRM PINN are presented in Code 19.2. In total, 4 boundary loss functions are being used in the PINN, 2 for the physics loss related to the pressure and 2 for the physics loss related to the thrust. All of the individual boundary loss weights have been set to 1 in the cost function. The first function, called Boundary_loss_pressure, is used to set the starting point of the simulation equal to atmospheric pressure conditions. Therefore, this loss makes use of only a single collocation point at $t = 0\,[s]$. The second boundary loss function related to the pressure is called Positive_loss_pressure and is used to make sure that all predicted values over the domain are larger than zero. The reason why this loss function is used is to prevent the simulation from obtaining negative values, which results in nonphysical outcomes of the simulation. This loss function makes use of 50 collocation points within the computational domain, ranging from 0 to 8 [s]. Since only positive and negative values need to be compared, the computational domain can start from 0 [s] rather than $1 \cdot 10^{-16}\,[s]$ as used in the physics loss functions. The third loss function, called Boundary_loss_force, is used to set the prediction for the thrust equal to zero at $t = 0\,[s]$. This loss function also makes use of a single collocation point, at $t = 0\,[s]$. Lastly, the loss function called Positive_loss_force is being used to ensure that the predictions for thrust are larger than zero. This loss function also makes use of 50 collocation points within the computational domain, ranging from 0 to 8 [s].

Code 19.1: Implementation of the physics losses of the solid rocket motor

```
def PINN_pressure_loss(NN_output: torch.nn.Module, used_inputs):
    # Collocation points time array
    start, end = 1e-16, 8
    steps = 100
    time_inputs = torch.linspace(start, end, steps).view(-1, 1).requires_grad_(True)

    # Setup total loss
    total_loss = 0

    for inputs in used_inputs:
        inputs = torch.expm1(inputs)

        # Unzip constant values from inputs
        L_grain = inputs[0]
        d_grain_in = inputs[1]
        d_grain_out = inputs[2]
        t_liner = inputs[3]
        mass_grain = inputs[4]
        N_grains = inputs[5]
        uninhibited_core = inputs[6]
        uninhibited_sides = inputs[7]
        gamma = inputs[9]
        W_g = inputs[10]
        T_0 = inputs[11]
        d_chamber = inputs[12]
        L_chamber = inputs[13]
        d_t = inputs[14]
        #d_e = inputs[15]
        a = inputs[17]
        n = inputs[18]


        # Perform calculations for standard values
        R_g = R_a / W_g
        A_t = np.pi * d_t**2/4  # Nozzle throat area [m]
        rho_grain =  mass_grain / ( L_grain * (np.pi / 4) * (d_grain_out**2 - d_grain_in**2))
    # Density of grain [kg/m3]

        # Repeat the constants for each time step (number of time points)
        # Assuming time_inputs is a tensor with shape (number of time points, 1)
        constant_features_repeated = inputs.unsqueeze(0).repeat(time_inputs.size(0), 1).
    requires_grad_(True)

        # Reshape the time feature to make it compatible for concatenation
        # time_feature is assumed to be of shape (number of time points, 1)
        time_feature = time_inputs.reshape(-1, 1)

```

```
        # Concatenate the constant features with the time feature along the second axis (columns)
        input_data = torch.cat((time_feature, constant_features_repeated), dim=1).requires_grad_(True)

        # Scale
        input_data = torch.log1p(input_data)

        # Obtain predictions from the network (Pc, a, n)
        P_C = NN_output(input_data)[:, 0].unsqueeze(-1)
        #a = NN_output.a
        #n = NN_output.n

        # Obtain derivative of Pc with respect to the scaled time over the network and convert to dPc/dt
        dP_C_dt = gradient(P_C, input_data)[:, 0]

        # Obtain r_integrator
        radius = SRM_helper_tools.r_integrator(P_C, a, n, time_inputs, d_grain_in/2)

        # Check when motor is burning or not
        burning_logic = torch.logical_and(radius <= (d_grain_out/2), (radius - (d_grain_in/2)) <= L_grain/2)

        # Set radius to fixed values if all propellant is burned
        radius[radius >= (d_grain_out/2)] = d_grain_out/2
        radius[(radius - (d_grain_in/2)) >= L_grain/2] = L_grain/2

        # Obtain dPdt from formula
        dPdt_theory = SRM_helper_tools.dPdt_check(radius, P_C, P_a, burning_logic, a, n, rho_grain, R_g, T_0, A_t, gamma, uninhibited_core, uninhibited_sides, L_grain, d_grain_in/2, d_grain_out, N_grains, L_chamber, d_chamber, t_liner) * (1 + time_inputs)

        # Determine loss for specific case and add to total loss
        total_loss += torch.mean((dPdt_theory - dP_C_dt)**2)

    return total_loss / len(used_inputs)

def PINN_thrust_loss(NN_output: torch.nn.Module, used_inputs):
    # Collocation points time array
    start, end = 1e-16, 8
    steps = 100
    time_inputs = torch.linspace(start, end, steps).view(-1, 1).requires_grad_(True)

    # Setup total loss
    total_loss = 0

    for inputs in used_inputs:
        inputs = torch.expm1(inputs)

        # Unzip constant values from inputs
        gamma = inputs[9]
        d_t = inputs[14]
        d_e = inputs[15]

        # Perform calculations for standard values
        A_t = np.pi * d_t**2/4  # Nozzle throat area [m]
        A_e = np.pi * d_e**2/4 # Nozzle exit area [m]

        # Repeat the constants for each time step (number of time points)
        # Assuming time_inputs is a tensor with shape (number of time points, 1)
        constant_features_repeated = inputs.unsqueeze(0).repeat(time_inputs.size(0), 1)

        # Reshape the time feature to make it compatible for concatenation
        # time_feature is assumed to be of shape (number of time points, 1)
        time_feature = time_inputs.reshape(-1, 1)

        # Concatenate the constant features with the time feature along the second axis (columns)
        input_data = torch.cat((time_feature, constant_features_repeated), dim=1)

```

```
        # Scaled input data
        input_data = torch.log1p(input_data)

        # Obtain predictions from the network (Pc, F, a, n)
        outputs_of_network = NN_output(input_data)
        P_C = outputs_of_network[:,0].unsqueeze(-1)
        F_network = outputs_of_network[:,1].unsqueeze(-1)

        # Obtain correct pressure ratio
        pepc = SRM_helper_tools.fsolve_PyTorch(SRM_helper_tools.AeAt_func, 0.001, args=(
    SRM_helper_tools.Gamma(gamma), gamma, A_e, A_t))

        # Calculate nozzle exit pressure
        P_e_sol = P_C*1e5 * pepc  # Units = [Pa]

        # Obtain thrust coefficient
        C_f = SRM_helper_tools.Cf(SRM_helper_tools.Gamma(gamma), gamma, P_e_sol, P_C*1e5, P_a
    *1e5, (A_e / A_t))

        # Calculate theoretical thrust
        F_theory = C_f * P_C*1e5 * A_t # Units = [N]

        # Determine loss for specific case and add to total loss
        total_loss += torch.mean((F_theory - F_network)**2)

    return total_loss / len(used_inputs)
```

Code 19.2: Implementation of the boundary loss of the solid rocket motor

```
# Define pressure boundary loss function (P_a at start)
def Boundary_loss_pressure(NN_output: torch.nn.Module, used_inputs):
    t_boundary = torch.tensor(0.).view(-1,1).requires_grad_(True)

    # Setup total loss
    total_loss = 0

    for inputs in used_inputs:
        inputs = torch.expm1(inputs)

        # Repeat the constants for each time step (number of time points)
        # Assuming time_inputs is a tensor with shape (number of time points, 1)
        constant_features_repeated = inputs.unsqueeze(0).repeat(t_boundary.size(0), 1)

        # Reshape the time feature to make it compatible for concatenation
        # time_feature is assumed to be of shape (number of time points, 1)
        time_feature = t_boundary.reshape(-1, 1)

        # Concatenate the constant features with the time feature along the second axis (
    columns)
        input_data = torch.cat((time_feature, constant_features_repeated), dim=1)

        # Scaled input data
        input_data = torch.log1p(input_data)

        # Obtain predictions from the network (Pc, F, a, n)
        P_C = NN_output(input_data)[:, 0].unsqueeze(-1)

        total_loss += (torch.squeeze(P_C) - P_a)**2

    return total_loss / len(used_inputs)


# Define force boundary loss function (0 at start)
def Boundary_loss_force(NN_output: torch.nn.Module, used_inputs):
    t_boundary = torch.tensor(0.).view(-1,1).requires_grad_(True)

    # Setup total loss
    total_loss = 0

    for inputs in used_inputs:
        inputs = torch.expm1(inputs)
```

```
42
43          # Repeat the constants for each time step (number of time points)
44          # Assuming time_inputs is a tensor with shape (number of time points, 1)
45          constant_features_repeated = inputs.unsqueeze(0).repeat(t_boundary.size(0), 1)
46
47          # Reshape the time feature to make it compatible for concatenation
48          # time_feature is assumed to be of shape (number of time points, 1)
49          time_feature = t_boundary.reshape(-1, 1)
50
51          # Concatenate the constant features with the time feature along the second axis (
        columns)
52          input_data = torch.cat((time_feature, constant_features_repeated), dim=1)
53
54          # Scaled input data
55          input_data = torch.log1p(input_data)
56
57          # Obtain predictions from the network (Pc, F, a, n)
58          F_network = NN_output(input_data)[:,1].unsqueeze(-1)
59
60          total_loss += (torch.squeeze(F_network) - 0)**2
61
62      return total_loss / len(used_inputs)
63
64
65  # Positive loss function
66  def Positive_loss_pressure(NN_output: torch.nn.Module, used_inputs):
67      steps = 50
68      time_inputs = torch.linspace(0, 8, steps).view(-1, 1).requires_grad_(True)
69
70      # Setup total loss
71      total_loss = 0
72
73      for inputs in used_inputs:
74          inputs = torch.expm1(inputs)
75
76          # Repeat the constants for each time step (number of time points)
77          # Assuming time_inputs is a tensor with shape (number of time points, 1)
78          constant_features_repeated = inputs.unsqueeze(0).repeat(time_inputs.size(0), 1)
79
80          # Reshape the time feature to make it compatible for concatenation
81          # time_feature is assumed to be of shape (number of time points, 1)
82          time_feature = time_inputs.reshape(-1, 1)
83
84          # Concatenate the constant features with the time feature along the second axis (
        columns)
85          input_data = torch.cat((time_feature, constant_features_repeated), dim=1)
86
87          # Scale
88          input_data = torch.log1p(input_data)
89
90          # Obtain predictions from the network (Pc, F, a, n)
91          P_C = NN_output(input_data)[:, 0].unsqueeze(-1)
92
93          total_loss += torch.mean(torch.clamp(-P_C, min=0)**2)
94
95      return total_loss / len(used_inputs)
96
97
98  # Positive loss function
99  def Positive_loss_force(NN_output: torch.nn.Module, used_inputs):
100     steps = 50
101     time_inputs = torch.linspace(0, 8, steps).view(-1, 1).requires_grad_(True)
102
103     # Setup total loss
104     total_loss = 0
105
106     for inputs in used_inputs:
107         inputs = torch.expm1(inputs)
108
109         # Repeat the constants for each time step (number of time points)
110         # Assuming time_inputs is a tensor with shape (number of time points, 1)
```

```
        constant_features_repeated = inputs.unsqueeze(0).repeat(time_inputs.size(0), 1)

        # Reshape the time feature to make it compatible for concatenation
        # time_feature is assumed to be of shape (number of time points, 1)
        time_feature = time_inputs.reshape(-1, 1)

        # Concatenate the constant features with the time feature along the second axis (
    columns)
        input_data = torch.cat((time_feature, constant_features_repeated), dim=1)

        # Scale
        input_data = torch.log1p(input_data)

        # Obtain predictions from the network (Pc, F, a, n)
        F_network = NN_output(input_data)[:,1].unsqueeze(-1)

        total_loss += torch.mean(torch.clamp(-F_network, min=0)**2)

    return total_loss / len(used_inputs)
```

To show the structure of the code and how the code will be implemented in Python, making use of the developed code class as presented in chapter 3, use can be made of a flow diagram, which is presented in Figure 19.2.

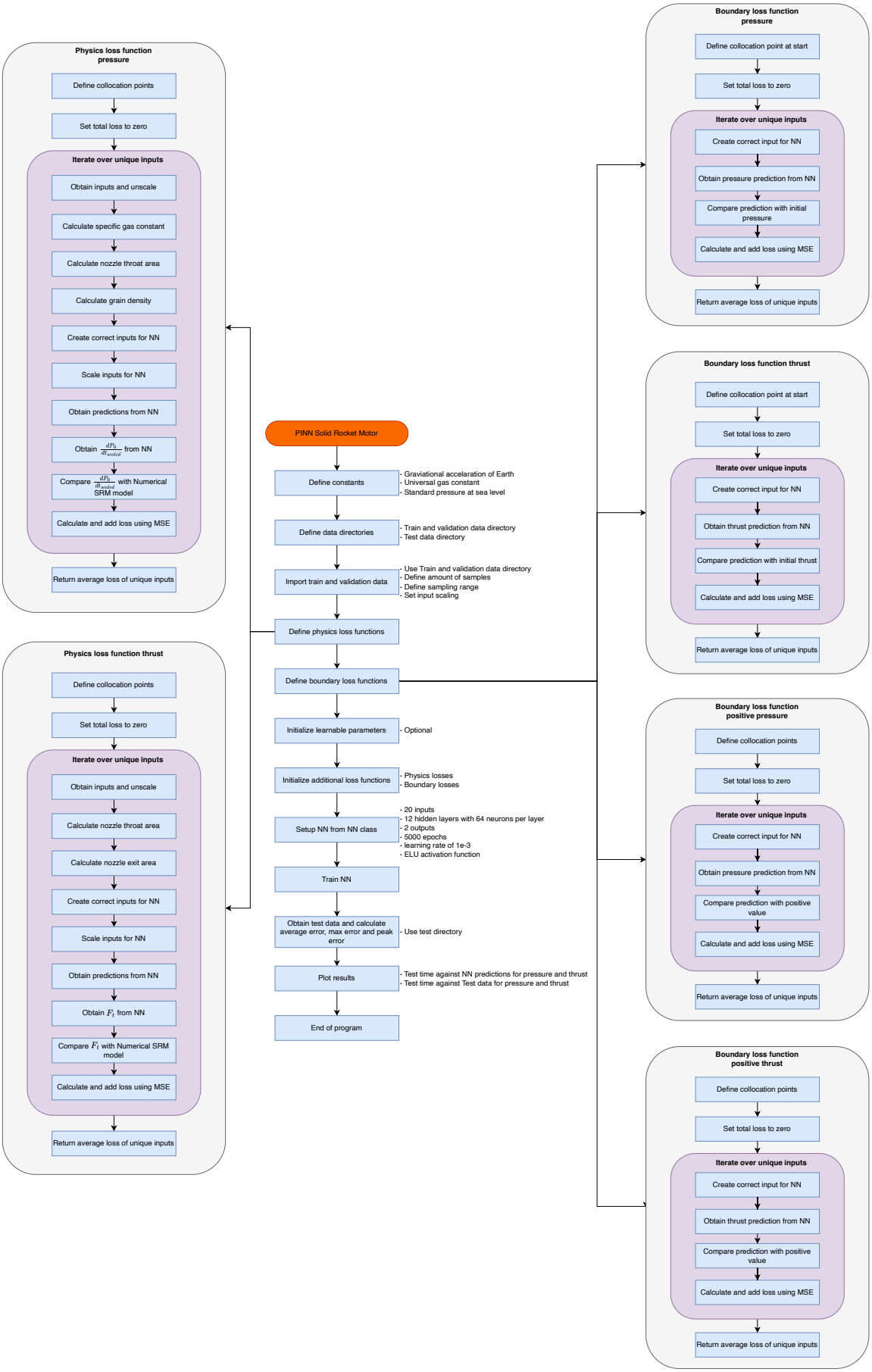


Figure 19.2: Flowchart of the developed PINN program for the modelling of SRM predictions in terms of chamber pressure and generated thrust in Python code

19.2. Results of the Model

The developed PINN predicts chamber pressure and generates thrust for an SRM. An example prediction output generated by the PINN is presented in Figure 19.3.

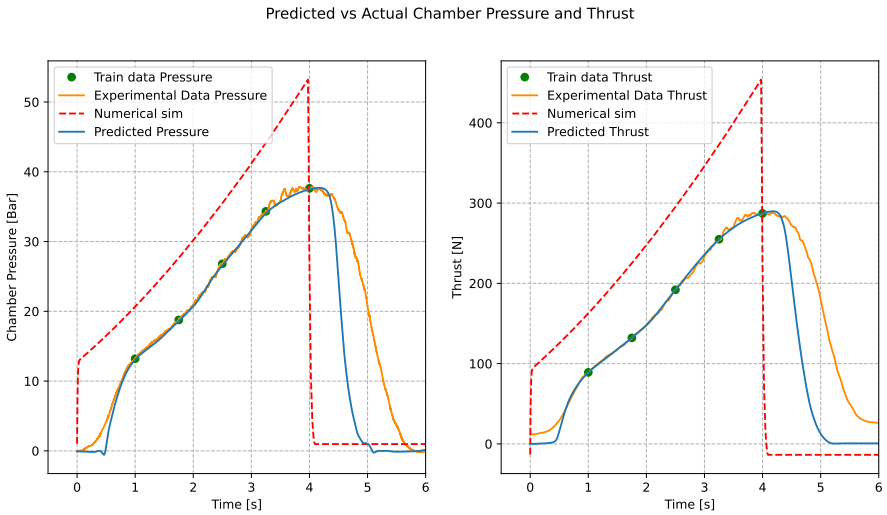


Figure 19.3: Example prediction output of chamber pressure and generated thrust by the SRM PINN model, compared to its training data and the real experimental data

From Figure 19.3, it can be seen that only 5 data points are provided as training data input, ranging from 1 to 4 [s]. Furthermore, it can be seen that the numerical model developed in chapter 18, presented as the dashed line in red, over-predicts both the predictions for chamber pressure and generated thrust compared to the measured experimental data, presented as the orange line. The PINNs prediction, on the other hand, which is presented as the blue line, shows a much better agreement with the measured experimental data, compared to the numerical model. Although only a few data points are provided, the PINN can predict the start-up transient and trail-off, as well as showing close agreement with the peak chamber pressure and generated thrust. However, the PINN is not able to correctly predict the start-up transient time and shape, as well as the correct duration of the burn time and the trail-off shape and direction. However, where the experimental data shows some nonphysical phenomena, such as having a non-zero starting value for thrust and a higher final value of thrust after the test, the PINN does predict a start and end value of 0 [N]. The provided example makes use of 5000 epochs and has a learning rate of $1 \cdot 10^{-3}$.

20

Model Comparison: Solid Rocket Motor

After the PINN model and Numerical model have been developed, as described in chapter 19 and chapter 18, the performance of each modelling method will be compared to be able to answer the research questions as presented in chapter 15. To be able to answer all research questions, a purely Data-Driven model has been developed as well, making use of the same network configuration as the PINN model, as presented in chapter 19. This chapter will examine and compare the performance of the various modelling methods, using the experimental data as presented in chapter 17. This chapter is structured as follows. First, the multiple models used in this comparison will be discussed in section 20.1. Next, the accuracy, data efficiency and computational efficiency of the various modelling methods will be examined, making use of the LOO-CV method in section 20.2. Lastly, the surrogate modelling performance of the various modelling methods for the prediction of SRM performance in terms of chamber pressure and generated thrust will be examined in section 20.3.

20.1. Comparison Model

In this section, the models which will be used for the comparison will be discussed. In this part of the report, a PINN model has been developed which predicts the performance of an SRM in terms of chamber pressure and generated thrust, as described in chapter 19. The developed PINN model for SRM performance predictions will be compared against a Numerical model developed for the same purpose, as was explained in chapter 18. Furthermore, the models will be compared against a purely Data-Driven model, making use of the same network architecture of the PINN, which was described in section 19.1. Furthermore, the models will be compared against an I-PINN, treating the burn rate coefficient, a, and the burning rate exponent, n, as additional network parameters. Since these quantities will obtain new values due to the learning process of the I-PINN, and to be able to make a fair comparison with the Numerical model, the last model, which will be used in the comparison, is called an I-Numerical model. This model has the same properties as the Numerical model described in chapter 18, but makes use of the burn rate coefficient and the burn rate exponent obtained from the trained I-PINN.

20.2. Leave-One-Out Cross Validation Comparison

As described in subsubsection 2.1.1.15, methods using a data-driven component need a different validation strategy to estimate the model's performance compared to numerical methods. To assess the performance of the various modelling methods developed during this chapter and to be able to answer the first 3 sub-research questions, it was decided to make use of the LOO-CV method to estimate the performance of the methods involving a data-driven component. In chapter 12, a total of 20 experimental data sets were available to be able to train and compare the various modelling methods. From these experiments, 1 data set corresponds to a failure event of an SRM. After careful evaluation, it was de-

cided not to make use of this dataset as it does not aid in answering the research questions. However, this dataset could be useful for future studies, where identification of anomalies could be investigated, as was discussed in section 16.7. Therefore, a total of 19 experimental data sets were available to perform the comparison of the various modelling methods predicting the performance of an SRM. When using the LOO-CV method, these 19 experiments are split into two groups. 18 data sets will be used to train the methods which contain a data-driven component and the resulting 1 will be used to asses the performance of each method. This process will be repeated until each of the 19 experimental data sets has been used once as test data to assess the performance of the various modelling methods. During the training process, a certain number of data points will be sampled uniformly over the domain and used to train the PINN, I-PINN and purely Data-Driven method. This means that each method will be trained using the same data points. To assess the performance, all the available data points of the test dataset will be used for the various methods. As with the training data points, the test data points will also be the same for each modelling method. To be able to assess the performance of the various modelling methods, 3 different performance metrics will be used. To obtain an overall estimate of the performance of each model, the RMSE will be used. Furthermore, to be able to assess the correctness of the shape of the predictions, use will be made of the maximum error, as it was found in Part I that this performance metric provides a good estimate of this criterion. Lastly, since the maximum value of the chamber pressure and generated thrust force are important design parameters, the peak error will also be determined. The peak error is defined as the error between the maximum value of the predicted chamber pressure and generated thrust compared to the maximum recorded experimental chamber pressure and generated thrust. For each of the tests, the 3 metrics will be used to assess the performance of each modelling method. Lastly, the performance metrics will be averaged to obtain a single number for the performance of each method. Besides the errors, the average of the training and execution time will also be recorded during the tests, to be able to assess the computational efficiency of each modelling method.

To be able to assess the data efficiency, accuracy and computational efficiency, the LOO-CV procedure will be conducted by varying the number of data points used during the training process. To be able to correctly assess the accuracy, a threshold should be determined, such that if the predictions fall below this accuracy threshold, the error is no longer assessable due to the inconsistency of the measurement equipment. To obtain the thresholds for chamber pressure and generated thrust, use can be made of the datasheets for the sensors used to measure the chamber pressure and the generated thrust force, as presented in Appendix I and Appendix J, respectively. From Appendix I, it can be found that the pressure sensor used to measure the chamber pressure has various sources of error. Making use of Equation 9.1 to obtain the total error, a total of 3 types of errors should be considered, as presented in Appendix I, namely the characteristics deviation, with an error of ± 0.5 [bar], the repeatability error of ± 0.05 [bar] and the temperature coefficient error of ± 0.1 [bar]. The long-term stability is neglected since it is unknown how long the sensor has been in use. Making use of Equation 9.1, this results in a threshold of 0.512 [bar], meaning that if the accuracy of the chamber pressure predictions is below this threshold, the deviations could be introduced due to the sensor error. A similar process can be conducted for the load cell, which measures the generated thrust force delivered by the SRM. When examining Appendix J, the total error of the load cell is based on the non-linearity error of ± 0.085 [N], hysteresis error of ± 0.1 [N], the repeatability error of ± 0.05 [N], the creep error of ± 0.1 [N], the temperature effect on zero of ± 0.085 [N] and the temperature effect on the span of ± 0.07 [N]. Making use of Equation 9.1, this results in a threshold of 0.87 [N] for the thrust force, meaning that predictions below this threshold could be introduced due to sensor inaccuracies.

20.2.1. Computational Efficiency

To examine the computational efficiency of the various modelling methods, the data obtained using the LOO-CV process has been summarised in bar charts. In Figure 20.1, the average training time for the various modelling methods has been presented, and in Figure 20.2, the execution time of the various modelling methods is presented.

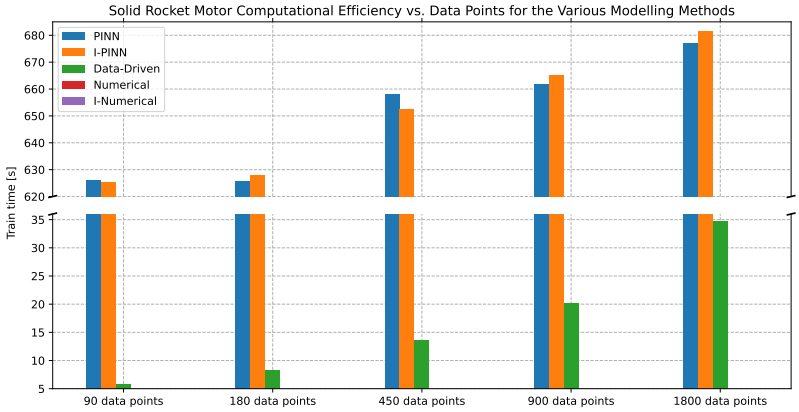


Figure 20.1: Bar chart presenting the average training time against the number of data points used during the training process for the various modelling methods for SRM modelling

Examining Figure 20.1, it can be seen that the training time of the PINN and I-PINN is much larger compared to the training time of the purely Data-Driven method. While the training time of the PINN and I-PINN is ranging between 625 and 682 [s], the training time of the purely Data-Driven method is ranging between 6 and 35 [s], meaning that the PINN and I-PINN methods have a training time of 20 to 125 times larger compared to the purely Data-driven method. The most likely explanation for this observation is that the PINN and I-PINN need to perform additional calculations to obtain the physics loss, while the purely Data-Driven method only needs to minimise the loss between the predictions and the training data. When comparing the PINN with the I-PINN, it can be seen that the two methods have a relatively similar training time, ranging from 625 to 677 [s] for the PINN and 625 to 682 [s] for the I-PINN modelling method. Overall, the training time of the I-PINN is the largest, having a difference in training time of 2 and 5 [s] compared to the PINN. However, sometimes, it was observed that the I-PINN had a slightly lower training time compared to the PINN. For all the modelling methods using a Data-Driven component, it is observed that an increase in data points used during the training process increases the training time and, therefore, computational time. This is expected as more data is available, more calculations need to be performed, which causes an increase in computational time.

Since the training phase only occurs once during the development of the modelling methods involving a data-driven component, a better metric to compare the performance of the various modelling methods in terms of computational efficiency would be the execution time. To be able to assess this metric, use can be made of Figure 20.2.

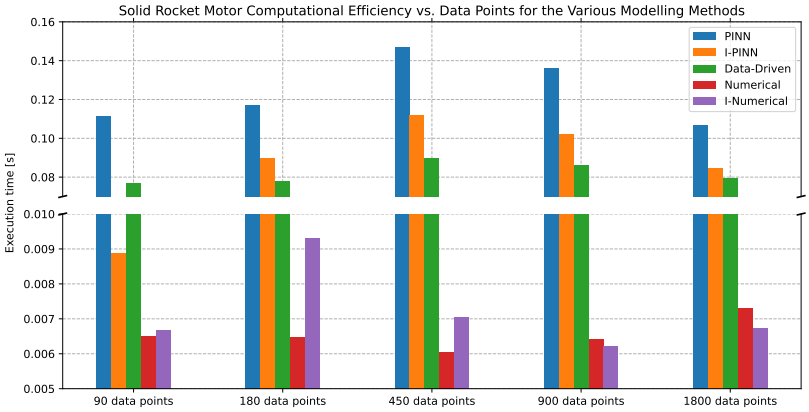


Figure 20.2: Bar chart presenting the average execution time against the number of data points used during the training process for the various modelling methods for SRM modelling

Making use of Figure 20.2, it can be seen that the Numerical methods have the lowest execution times, ranging between 0.006 to 0.0095 [s]. The methods involving a data-driven component, on the other hand, have a higher execution time, ranging from 0.0088 [s] to 0.145 [s]. However, besides the I-PINN having an execution time of approximately 0.0088 [s], making use of 90 data-points using the training phase, the execution time of the methods involving a data-driven component ranges between 0.08 and 0.145 [s]. This means that the execution time of methods involving a data-driven component is between 13 and 15 times higher compared to the Numerical methods. Furthermore, it can be seen that the execution time of the methods involving a data-driven component slightly increases with the number of data points used during the training process, whereafter a slight decay is observed. This is most likely due to an inconsistency in the measuring method, meaning that this observation is most likely not a trend. The PINN and I-PINN have a higher execution time compared to the purely Data-Driven method, with the PINN having the largest execution time. This observation is interesting since the number of layers and neurons is similar, meaning that the forward pass through the network follows the same path in each method involving a data-driven component. However, the most likely explanation for this observation is that the method by which the execution time is obtained is not consistent, causing this difference in execution times. It is expected, however, that the modelling methods involving a data-driven component have a similar order of magnitude in terms of execution time. To conclude, the Numerical methods have a computational efficiency which is about 13 to 15 times higher compared to the methods involving a data-driven component during the execution time. Furthermore, the PINN and I-PINN do have a training time 20 to 125 times larger compared to a purely Data-Driven method, meaning that PINNS and I-PINNS do not improve computational efficiency compared to a Numerical and purely Data-Driven method.

20.2.2. Data Efficiency

To assess the data efficiency, a certain threshold needs to be set from which it can be examined how many data points during the training phase are required to obtain a certain accuracy. To do so, the average maximum error of the chamber pressure and thrust predictions will be used. The predictions of the chamber pressure are presented in Figure 20.3, and the predictions of the generated thrust are presented in Figure 20.4.

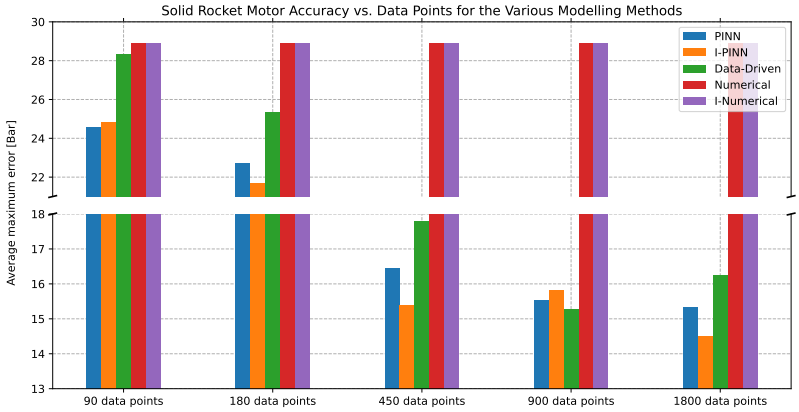


Figure 20.3: Bar chart presenting the average maximum error of the chamber pressure predictions against the number of data points used during the training process for the various modelling methods

Making use of Figure 20.3, it can be seen that the average maximum error of the chamber pressure predictions decreases with an increase in the number of data points for all the methods involving a data-driven component. When comparing all the modelling methods, it can be observed that the PINN and I-PINN need fewer data points to obtain a certain accuracy level for their predictions of chamber pressure in terms of average maximum error. For example, when comparing the PINN and I-PINN with the purely Data-Driven method for 450 data points used to train the models, it can be seen that while the purely Data-Driven method has an error of about 17.6 [Bar], the I-PINN has an average maximum error of approximately 15.3 [Bar]. To reach the same level of accuracy using the purely Data-Driven method, 900 data points are required, meaning that the I-PINN has a data efficiency increase of 50% compared to the purely Data-Driven method. From all the modelling methods, it can be observed that the I-PINN has the best accuracy in terms of average maximum error for chamber pressure predictions, whereafter the PINN and the purely Data-Driven method. The Numerical and I-Numerical methods have the worst performance, almost doubling the average maximum error using 900 data points compared to the modelling methods involving a data-driven component. Furthermore, it can be seen that the difference between the Numerical and I-Numerical methods is relatively small. There could be multiple reasons for this observation, but the most likely explanation is that the additional network parameters a and n were not correctly implemented in the I-Numerical model, or the I-PINN was not able to capture the trend, meaning the numerical values of the additional network parameters were not different compared to the initial standard values. To be able to correctly obtain the data efficiency, the average maximum error of the generated thrust predictions should also be assessed, which is presented in Figure 20.4.

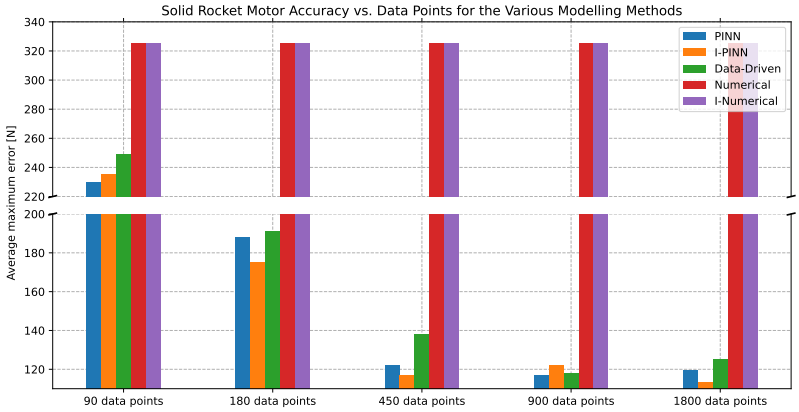


Figure 20.4: Bar chart presenting the average maximum error of the generated thrust predictions against the number of data points used during the training process for the various modelling methods

From Figure 20.4, a similar trend can be observed compared to Figure 20.3. The I-PINN, in general, has the lowest average maximum error, followed by the PINN and the purely Data-Driven method. The 50% increase in data efficiency of the I-PINN compared to the purely Data-Driven method can also be observed when increasing the number of data points used during training from 450 to 900 data points. However, while this trend can also be observed when increasing the amount of data points from 90 data points to 180 data points for the average maximum error for generated thrust, as can be seen in Figure 20.4, the same pattern is not visible for the average maximum error for the chamber pressure, as can be seen from Figure 20.3. However, since there is a clear decreasing trend showing that the PINN and I-PINN need fewer data points to obtain a similar accuracy level compared to a purely Data-Driven method, it can be concluded that the PINN and I-PINN have an improvement in data efficiency of about 50% compared to a purely Data-Driven method.

20.2.3. Accuracy

To be able to assess the accuracy of the various modelling methods, use will be made of the average RMSE and average peak error for the chamber pressure and generated thrust predictions. Firstly, the average RMSE will be presented in Figure 20.5 for the chamber pressure predictions and in Figure 20.6 for the generated thrust predictions.

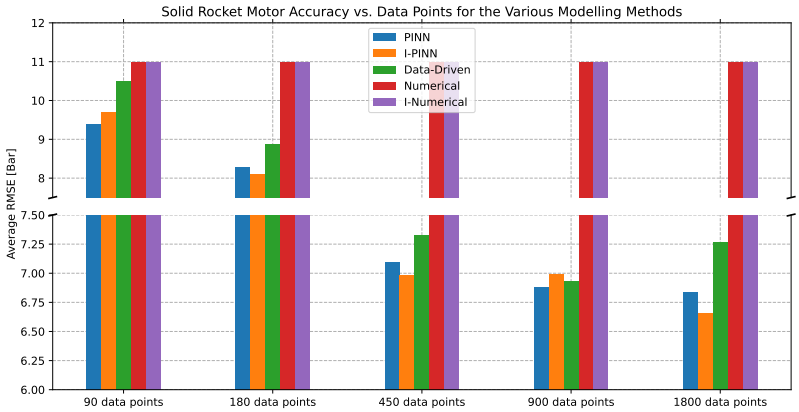


Figure 20.5: Bar chart presenting the average RMSE error of the chamber pressure predictions against the number of data points used during the training process for the various modelling methods

From Figure 20.5, it can be seen that, in general, the PINN and I-PINN are the most accurate in terms of average RMSE for chamber pressure predictions. The Numerical method is the most inaccurate compared to the other methods. However, while the PINN and I-PINN are the most accurate, the purely Data-Driven method has a similar accuracy level compared to the PINN and I-PINN modelling methods. Increasing the number of data points used during the training phase increases the accuracy of the methods, which involve a data-driven component for chamber pressure predictions in terms of average RMSE. However, when using 1800 data points, it can be seen that the purely Data-Driven has a decrease in accuracy, while the PINN and I-PINN remain in their decreasing trend. Next, the average RMSE for generated thrust predictions will be assessed, making use of Figure 20.6.

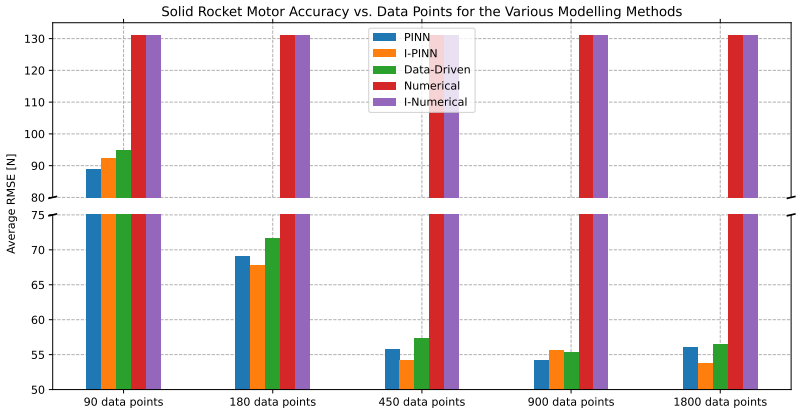


Figure 20.6: Bar chart presenting the average RMSE error of the generated thrust predictions against the number of data points used during the training process for the various modelling methods

From Figure 20.6, it can be seen that a similar trend compared to the average RMSE for chamber pressure predictions is present. The PINN and I-PINN show the highest accuracy level for average RMSE for generated thrust predictions, while the Numerical method is the most inaccurate modelling method. The purely Data-Driven has a similar accuracy level compared to the PINN and I-PINN methods. All the methods involving a data-driven component show an increase in accuracy when increasing the number of data points used during the training process. To be able to fully assess the performance in terms of accuracy, lastly, the average peak error of the chamber pressure and generated thrust predictions will be assessed against the number of data points. In Figure 20.7, the average peak error of the chamber pressure prediction is presented and in Figure 20.8, the average peak error of the generated thrust predictions is presented.

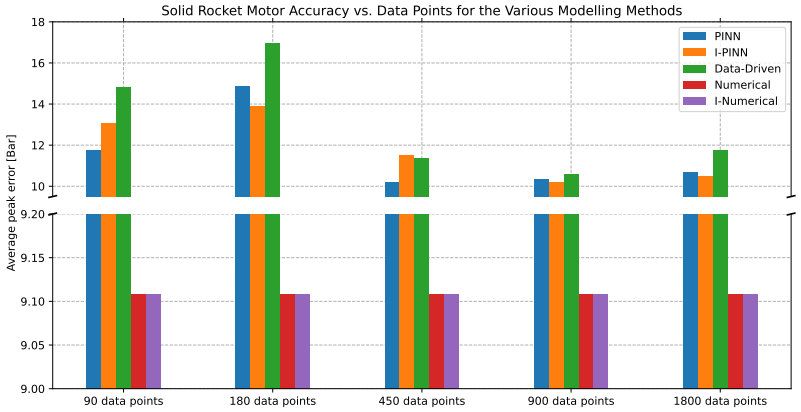


Figure 20.7: Bar chart presenting the average peak error of the chamber pressure predictions against the number of data points used during the training process for the various modelling methods

From Figure 20.7, something interesting can be observed. While the accuracy in terms of average maximum error and average RMSE for chamber pressure is lower for the Numerical methods, in terms of peak error, the Numerical methods have the highest accuracy for chamber pressure predictions. The methods involving a data-driven component are not able to obtain a similar accuracy level compared to the Numerical methods, stabilising at 450 data points used during training, with a difference of about 1 [Bar] compared to the Numerical method. In some situations, such as for 180 data points, the purely Data-Driven method has an accuracy which is almost 2 times less compared to the Numerical methods. When comparing the methods involving a data-driven component, it can be observed that in general, the PINN and I-PINN have the highest accuracy, while the purely Data-Driven method remains closely related in terms of performance to the PINN and I-PINN. Next, the average peak error for the generated thrust prediction will be assessed, as presented in Figure 20.8.

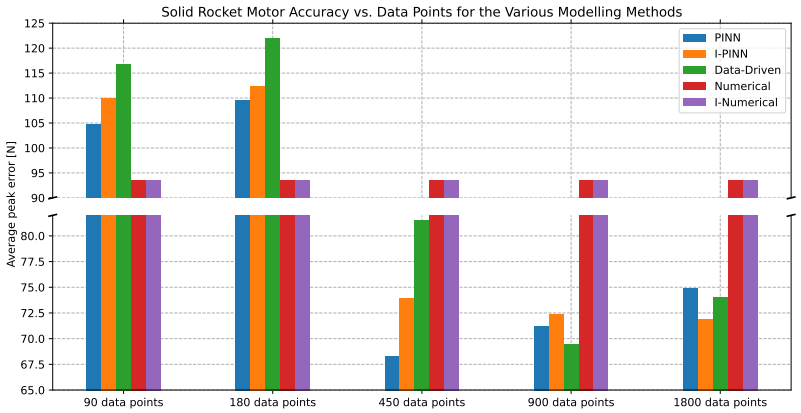


Figure 20.8: Bar chart presenting the average peak error of the generated thrust predictions against the number of data points used during the training process for the various modelling methods

Examining Figure 20.8, it can be observed that, while the Numerical methods have a higher accuracy in terms of average peak error at 90 and 180 data points used during training for generated thrust predilections, this decreases when making use of 450 data points to train the methods involving a data driven component. When increasing the amount of data points from 450 to 900, it can be seen that a similar accuracy is obtained for the purely Data-Driven compared to the accuracy obtained from the PINN and I-PINN when making use of 450 data points.

To conclude, the methods involving a data-driven component have a higher accuracy compared to the Numerical methods. However, while there are indications that the PINN and I-PINN methods are slightly more accurate compared to a purely Data-Driven method, this was found not to be statistically significant. When comparing the peak average level, however, it can be seen that, specifically for the chamber pressure predictions, the Numerical methods have a slightly higher accuracy compared to the methods involving a data-driven component.

20.2.4. Data Table LOO-CV

The detailed numbers obtained during the LOO-CV comparison from the previous section are summarised in Table 20.1.

Table 20.1: Comparison of Solid Rocket Motor: Computational Efficiency and Accuracy across the various used methods

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Computational Efficiency					
Train Time (using 90 data points) [s]	625.971680	625.220151	5.763701	x	x
Train Time (using 180 data points) [s]	625.671486	628.079316	8.166276	x	x
Train Time (using 450 data points) [s]	658.030522	652.483021	13.626438	x	x
Train Time (using 900 data points) [s]	661.606419	665.063915	20.172568	x	x
Train Time (using 1800 data points) [s]	676.931989	681.354580	34.642038	x	x
Execution Time (using 90 data points) [s]	0.111690	0.0088725	0.076965	0.006519	0.006677
Execution Time (using 180 data points) [s]	0.117343	0.089810	0.077773	0.0064787	0.009323
Execution Time (using 450 data points) [s]	0.147069	0.111945	0.089876	0.006040	0.007049
Execution Time (using 900 data points) [s]	0.136122	0.102351	0.086139	0.006410	0.006228
Execution Time (using 1800 data points) [s]	0.107058	0.084711	0.079482	0.007306	0.006736
Accuracy					
Avg. RMSE Chamber pressure (using 90 data points) [bar]	9.385099	9.698429	10.504964	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 90 data points) [bar]	11.734110	13.045510	14.832508	9.108076	9.108077
Avg. Max Error Chamber pressure (using 90 data points) [bar]	24.550128	24.824798	28.310988	28.905221	28.905221
Avg. RMSE Thrust (using 90 data points) [N]	88.950938	92.282979	95.046512	131.211803	131.211806
Avg. Peak Error Thrust (using 90 data points) [N]	104.693971	109.846437	116.744696	93.487317	93.487328
Avg. Max Error Thrust (using 90 data points) [N]	230.114779	235.398981	248.873631	324.990327	324.990339
Avg. RMSE Chamber pressure (using 180 data points) [bar]	8.274266	8.113656	8.865576	10.996256	10.996256

Continued on next page

Table 20.1 – continued from previous page

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Avg. Peak Error Chamber pressure (using 180 data points) [bar]	14.852079	13.900468	16.949333	9.1080767	9.108077
Avg. Max Error Chamber pressure (using 180 data points) [bar]	22.694742	21.683441	25.311923	28.905221	28.905221
Avg. RMSE Thrust (using 180 data points) [N]	69.057836	67.718571	71.666021	131.211803	131.211806
Avg. Peak Error Thrust (using 180 data points) [N]	109.473773	112.282004	121.987764	93.487317	93.487328
Avg. Max Error Thrust (using 180 data points) [N]	187.954493	175.216216	191.205326	324.990327	324.990339
Avg. RMSE Chamber pressure (using 450 data points) [bar]	7.095736	6.980973	7.329081	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 450 data points) [bar]	10.202130	11.491606	11.383740	9.108076	9.108077
Avg. Max Error Chamber pressure (using 450 data points) [bar]	16.453792	15.389970	17.792068	28.905221	28.905221
Avg. RMSE Thrust (using 450 data points) [N]	55.682239	54.215492	57.302056	131.211803	131.211806
Avg. Peak Error Thrust (using 450 data points) [N]	68.218601	73.876301	81.515355	93.487317	93.487328
Avg. Max Error Thrust (using 450 data points) [N]	122.233782	116.784106	138.172854	324.990327	324.990339
Avg. RMSE Chamber pressure (using 900 data points) [bar]	6.878653	6.989631	6.93378	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 900 data points) [bar]	10.346630	10.199791	10.585910	9.108076	9.108077
Avg. Max Error Chamber pressure (using 900 data points) [bar]	15.5357267	15.8263626	15.272270	28.905221	28.905221
Avg. RMSE Thrust (using 900 data points) [N]	54.198997	55.628319	55.245823	131.211803	131.211806
Avg. Peak Error Thrust (using 900 data points) [N]	71.1411727	72.308060	69.390418	93.487317	93.487328
Avg. Max Error Thrust (using 900 data points) [N]	117.092485	122.272280	118.083813	324.990327	324.990339
Avg. RMSE Chamber pressure (using 1800 data points) [bar]	6.838115	6.654954	7.2686840	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 1800 data points) [bar]	10.656086	10.505123	11.729235	9.108076	9.108077
Avg. Max Error Chamber pressure (using 1800 data points) [bar]	15.336492	14.501622	16.251237	28.905221	28.905221

Continued on next page

Table 20.1 – continued from previous page

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Avg. RMSE Thrust (using 1800 data points) [N]	55.986078	53.784242	56.427613	131.211803	131.211806
Avg. Peak Error Thrust (using 1800 data points) [N]	74.909374	71.844447	74.012743	93.487317	93.487328
Avg. Max Error Thrust (using 1800 data points) [N]	119.731380	113.366296	125.075596	324.990327	324.990339

20.3. Surrogate Modelling Comparison

In section 20.2, the performances of the various modelling methods were compared in terms of accuracy, data efficiency and computational efficiency, making use of the LOO-CV method. From this section, it was included that the PINN and I-PINN have a higher data efficiency compared to the other modelling methods and that there are strong indications that the PINN and I-PINN modelling methods have a higher accuracy compared to a Numerical method, while having a similar order of accuracy compared to a purely Data-Driven method. The Numerical method is superior in terms of computational efficiency. In this section, the surrogate-ness of the various modelling methods will be examined, as was explained in subsection 2.1.2. To do so, use will be made of the data partition strategy as presented in section 17.4, where the various data groups will be examined and compared against a baseline LOO-CV test. During the tests, data groups will be excluded during the training phase, thereby assessing the surrogate-ness of the various modelling methods. The baseline LOO-CV case makes use of 900 data points, and to train the groups, each data file will make use of 50 data points per file. The network architecture of the modelling methods containing a data-driven component is the same network architecture used in section 20.2. All methods will be assessed by making use of the average RMSE, average maximum error and average peak error for the chamber pressure predictions and generated thrust predictions.

20.3.1. Nozzle Sizes Group

First, the performance of the various modelling methods in terms of surrogate modelling will be assessed by removing various nozzle size groups from the data sets used during the training phase. In Figure 20.9 and Figure 20.10, the average RMSE for the chamber pressure and thrust predictions are presented.

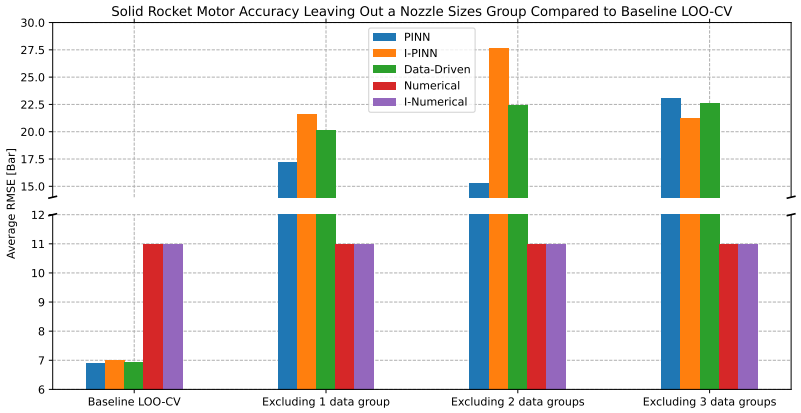


Figure 20.9: Bar chart presenting the average RMSE of the chamber pressure predictions against various data partitioning strategies for the nozzle size groups for the various modelling methods

From Figure 20.9, it can be seen that compared to the baseline, none of the methods involving a data-driven component can accurately provide predictions when excluding data groups from the training process. Furthermore, it can be seen that the I-PINN provides the most inaccurate predictions, while

the PINN is the most accurate modelling method of the methods involving a data-driven component. However, the Numerical methods are continuously outperforming these methods, showing that in terms of average RMSE of chamber pressure predictions, the PINN and I-PINN modelling methods are not able to capture the general trend for surrogate modelling. In Figure 20.10, the RMSE for the thrust predictions is presented.

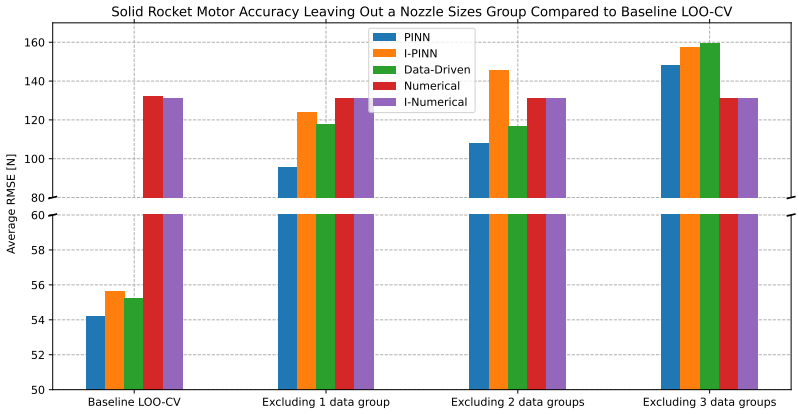


Figure 20.10: Bar chart presenting the average RMSE of the generated thrust predictions against various data partitioning strategies for the nozzle size groups for the various modelling methods

From Figure 20.10, it can be seen that, similarly to the chamber pressure, the methods involving a data-driven component are not able to obtain a similar order of accuracy compared to the baseline. However, when excluding 1 and 2 data groups, the performance is similar to the Numerical methods, where the PINN has a slight increase in performance, providing the most accurate results of all the other methods when excluding 1 or 2 data groups. When excluding 3 data groups, the Numerical methods have a higher accuracy. This shows that, in terms of average RMSE, the methods involving a data-driven method are not able to capture the general trend and provide inaccurate results in terms of surrogate modelling compared to a Numerical method in terms of average RMSE for generated thrust predictions. Next, the average peak error for chamber pressure and generated thrust will be examined in Figure 20.11 and Figure 20.12.

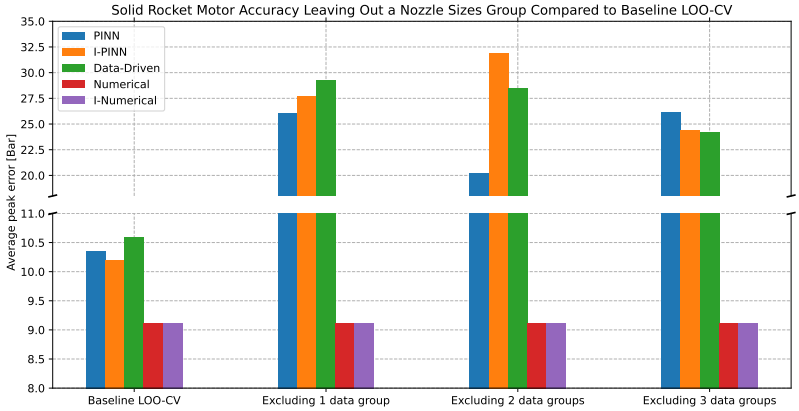


Figure 20.11: Bar chart presenting the average peak error of the chamber pressure predictions against various data partitioning strategies for the nozzle size groups for the various modelling methods

From Figure 20.11, it can be seen that the methods involving a data-driven component have a decrease

in accuracy of almost 3 times compared to the baseline. While the PINN has a slightly higher accuracy of the modelling methods involving a data-driven component when excluding 1 or 2 data groups, it is the least accurate method when excluding 3 data groups. The average peak error in terms of generated thrust is presented in Figure 20.12

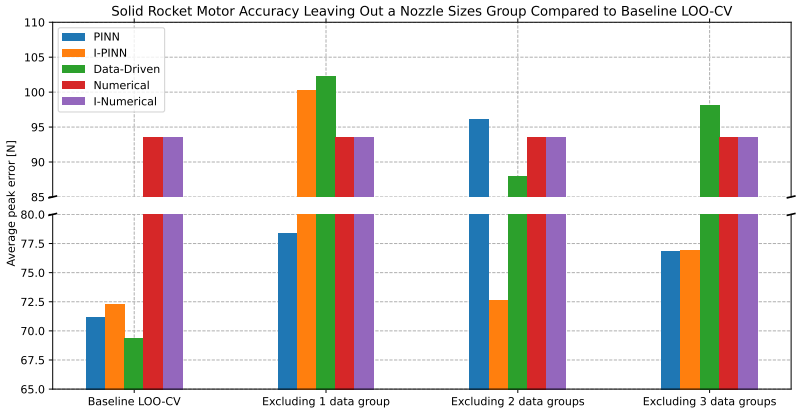


Figure 20.12: Bar chart presenting the average peak error of the generated thrust predictions against various data partitioning strategies for the nozzle size groups for the various modelling methods

From Figure 20.12, it can be seen that, as with the chamber pressure, the accuracy of the models involving a data-driven component becomes less accurate compared to the baseline when excluding data groups. However, something interesting can be observed when excluding more data groups from the training groups: the peak error for the generated thrust prediction seems to decrease for various methods. The PINN and I-PINN overall have the best performance of the various modelling methods involving a data-driven component, outperforming the Numerical methods when more data groups are excluded during the training phase. Compared to the baseline, the performance of the PINN and I-PINN is comparable, however, a consistent trend is not present, meaning that 1 method does not consistently have a higher accuracy compared to the other modelling methods. Although it seems that there are indications that the PINN and I-PINN have an higher accuracy in terms of average peak error for thrust, this is not the case when comparing chamber pressure, which means that it is concluded that the PINN and I-PINN method are not able to be used for surrogate modelling in terms of average peak error for chamber pressure and thrust predictions. Lastly, the average maximum error in chamber pressure and thrust predictions will be examined, making use of Figure 20.13 for chamber pressure and Figure 20.14 for generated thrust predictions.

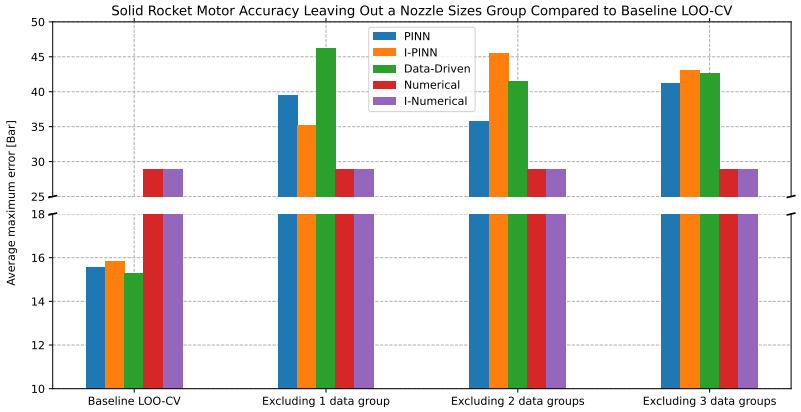


Figure 20.13: Bar chart presenting the average maximum error of the chamber pressure predictions against various data partitioning strategies for the nozzle size groups for the various modelling methods

From Figure 20.13, it can be observed that compared to the baseline, the average maximum errors are almost twice as large when excluding data groups for the methods involving a data-driven component. While the PINN and I-PINN do sometimes have a higher accuracy compared to the purely Data-Driven method, all the methods involving a data-driven component are less accurate compared to the Numerical model. Next, the average maximum error of the generated thrust prediction will be examined in Figure 20.14.

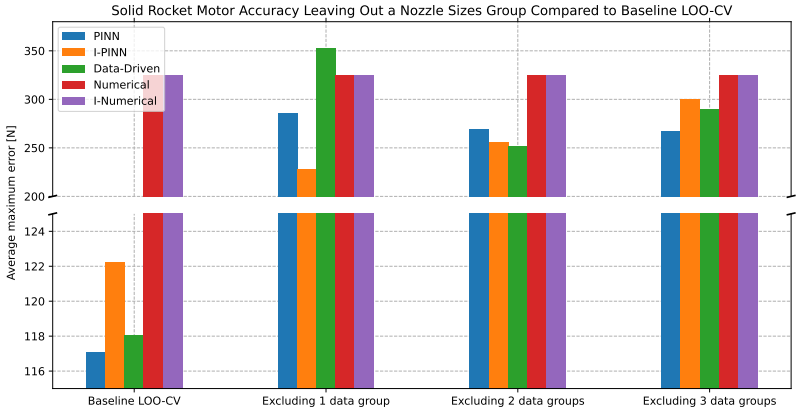


Figure 20.14: Bar chart presenting the average maximum error of the generated thrust predictions against various data partitioning strategies for the nozzle size groups for the various modelling methods

From Figure 20.14, it can be seen that, similarly to the chamber pressure predictions, the generated thrust predictions have an average maximum error almost twice as large compared to the baseline for the methods involving a data-driven component. Although when excluding groups during the training process, the PINN and I-PINN methods have a higher accuracy compared to the Numerical methods, there is no evidence that the PINN and I-PINN methods can act as a surrogate model in terms of average maximum error.

To conclude, when excluding various groups of nozzle sizes during the training process, it was found that the PINN and I-PINN are not able to act as surrogate modelling methods, having errors almost twice as high compared to the standard LOO-CV baseline. The most likely reason for this is that the network architecture has not been sufficiently optimised for surrogate modelling. Although a difference

in nozzle sizes is present in the physics loss, the PINN and I-PINN are most likely not able to capture this trend.

20.3.2. Grain Design Group

After the nozzle group was examined, the various modelling methods' performance for surrogate modelling will be assessed by removing various grain design groups from the data sets used during the training phase. In Figure 20.15 and Figure 20.16, the average RMSE for the chamber pressure and thrust predictions are presented.

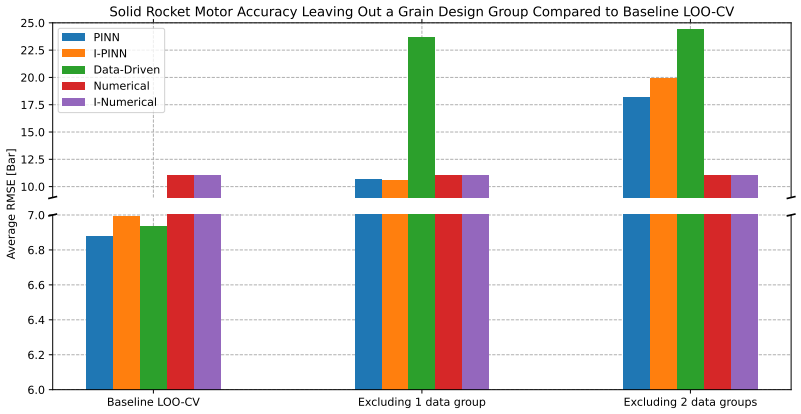


Figure 20.15: Bar chart presenting the average RMSE of the chamber pressure predictions against various data partitioning strategies for the grain design groups for the various modelling methods

From Figure 20.9, it can be seen that when excluding 1 data group, the PINN and I-PINN have a slight decrease in accuracy compared to the baseline. The purely Data-Driven method is the most inaccurate, with a decrease in performance of almost 3 times compared to the baseline LOO-CV. When excluding 2 data groups, the PINN and I-PINN have unsatisfactory accuracy compared to the baseline LOO-CV. Overall, the Numerical methods consistently outperform the methods involving a data-driven component, showing that in terms of average RMSE of chamber pressure predictions, the PINN and I-PINN modelling methods cannot capture the general trend for surrogate modelling. However, compared to the purely Data-Driven model, the PINN and I-PINN can capture the general trend. In Figure 20.10, the RMSE for the thrust predictions is presented.

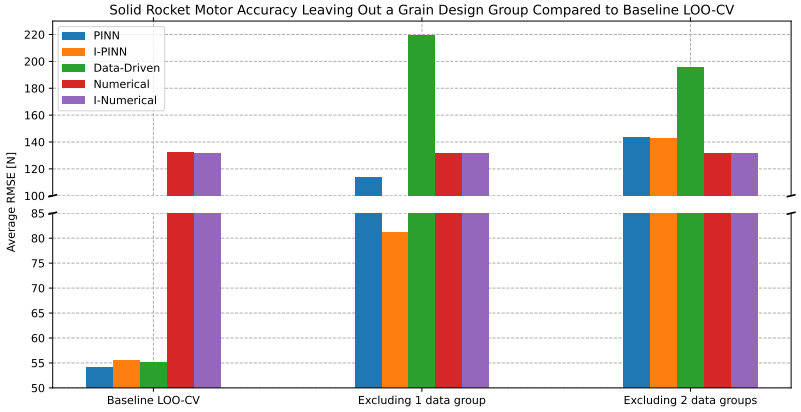


Figure 20.16: Bar chart presenting the average RMSE of the generated thrust predictions against various data partitioning strategies for the grain design groups for the various modelling methods

From Figure 20.16, it can be seen that, similarly to the chamber pressure, the methods involving a data-driven component are not able to provide a similar level of accuracy compared to the baseline LOO-CV. Furthermore, it can be observed that the purely Data-Driven method is the most inaccurate modelling method, while the PINN and I-PINN do have a slightly better accuracy compared to the Numerical methods. Overall, in terms of average RMSE, the methods involving a data-driven method are not able to capture the general trend and are more inaccurate in terms of surrogate modelling compared to a Numerical method. However, as previously discussed, the purely Data-Driven method is most likely not able to capture the general trend, while the PINN and I-PINN seem to be able to better capture the general trend. Next, the average peak error for chamber pressure and generated thrust will be examined in Figure 20.17 and Figure 20.18.

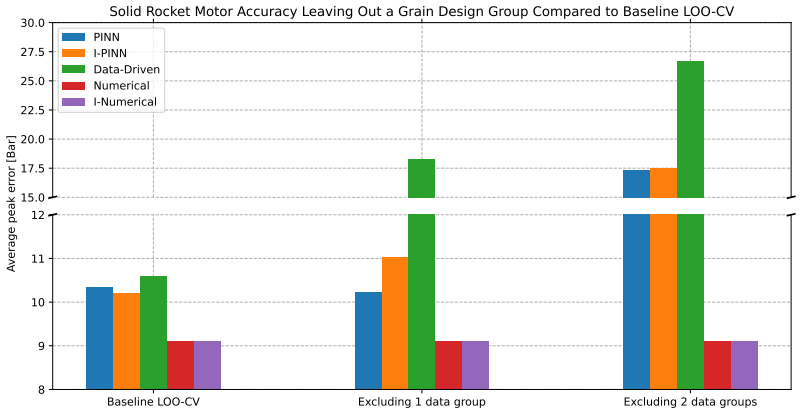


Figure 20.17: Bar chart presenting the average peak error of the chamber pressure predictions against various data partitioning strategies for the grain design groups for the various modelling methods

From Figure 20.11, it can be seen that when leaving out 1 data group, the accuracy of the PINN and I-PINN modelling method does not decrease a lot compared to the baseline LOO-CV. The purely Data-Driven method, on the other hand, has a decrease in accuracy of almost 2 times compared to the baseline LOO-CV. When excluding 2 data groups, the PINN and I-PINN methods are not able to provide satisfactory accuracy compared to the baseline LOO-CV. However, the PINN and I-PINN modelling methods are better at capturing the general trend compared to the purely Data-Driven model. This means that, in terms of average maximum error for the chamber pressure, the PINN and I-PINN are not suitable for surrogate modelling. However, the PINN and I-PINN can capture the general trend compared to the purely Data-Driven model, which is not able to capture this trend. The average peak error in terms of generated thrust is presented in Figure 20.18

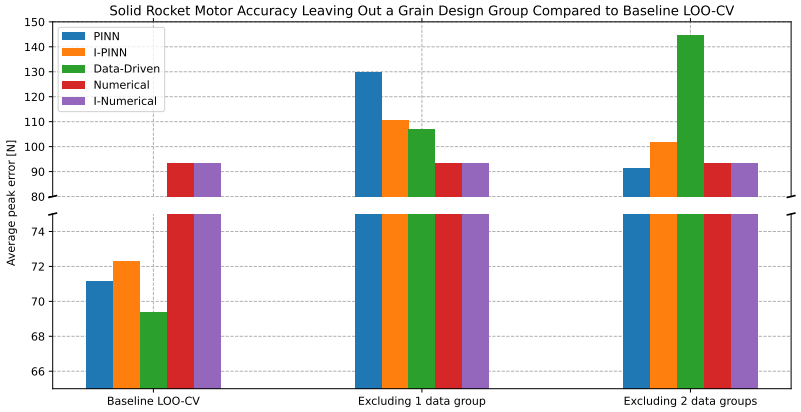


Figure 20.18: Bar chart presenting the average peak error of the generated thrust predictions against various data partitioning strategies for the grain design groups for the various modelling methods

From Figure 20.18, it can be seen that, as with the chamber pressure, the accuracy of the models involving a data-driven component decreases compared to the baseline LOO-CV. However, something interesting can be observed when excluding more data groups from the training groups: the peak error for the generated thrust prediction seems to decrease for the modelling methods involving a data-driven component. When excluding 1 data group, the PINN is the most inaccurate modelling method. However, when excluding 2 data groups, the purely Data-Driven method is the most inaccurate modelling method compared to the baseline LOO-CV, while the PINN and I-PINN seem to have an increase in accuracy when leaving out more data groups. Although there are indications that the PINN and I-PINN have a higher accuracy in terms of average peak error for generated thrust predictions, it is concluded that the PINN and I-PINN methods cannot able to be used for surrogate modelling in terms of average peak error for chamber pressure and thrust predictions. Lastly, the average maximum error in chamber pressure and thrust predictions will be examined, making use of Figure 20.19 for chamber pressure and Figure 20.20 for generated thrust predictions.

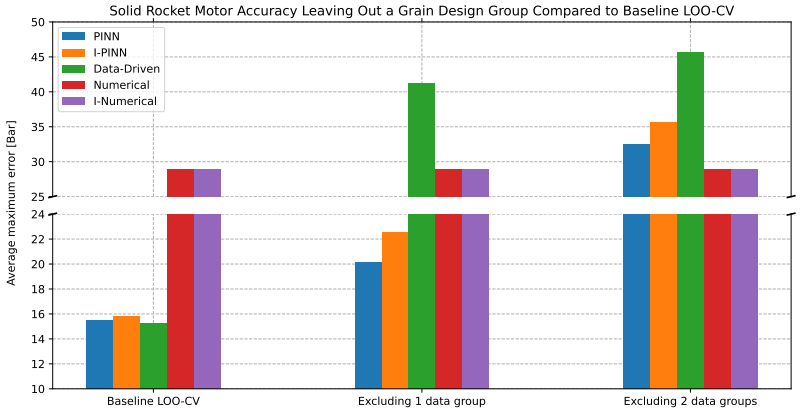


Figure 20.19: Bar chart presenting the average maximum error of the chamber pressure predictions against various data partitioning strategies for the grain design groups for the various modelling methods

From Figure 20.19, it can be seen that compared to the baseline, the PINN and I-PINN only have a slight decrease in accuracy when excluding 1 data group compared to the baseline LOO-CV. Furthermore, it can be seen that, while the PINN and I-PINN have relatively similar performance compared to the Numerical methods, the purely Data-Driven method is the most inaccurate modelling method compared

to the baseline LOO-CV. Lastly, the average maximum error of the generated thrust prediction will be examined in Figure 20.20.

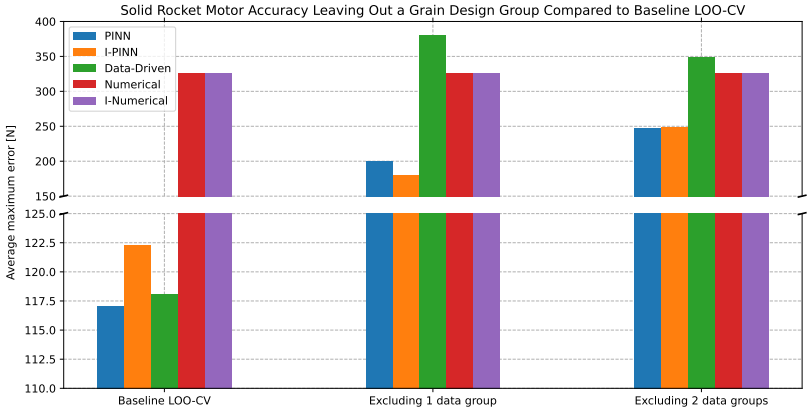


Figure 20.20: Bar chart presenting the average maximum error of the generated thrust predictions against various data partitioning strategies for the grain design groups for the various modelling methods

From Figure 20.20, it can be seen that the generated thrust predictions for the PINN and I-PINN have a decrease in accuracy when excluding data groups from the training process, where the purely Data-Driven method has a decrease in performance of almost 3 times compared to the baseline. While the PINN and I-PINN do have a higher accuracy compared to the Numerical methods, the purely Data-Driven method can be considered the most inaccurate modelling method of all of the modelling methods in terms of average maximum error for generated thrust predictions.

To conclude, when excluding various groups of grain designs during the training process, it was found that the PINN and I-PINN are not able to act as surrogate modelling methods, having errors almost twice as high for the average RMSE and average maximum error compared to the standard LOO-CV baseline. Although the performances of the PINN and I-PINN show a better ability to surrogate modelling compared to the nozzle sizes group discussed in subsection 20.3.1, it is concluded that the PINN and I-PINN are not able to act as surrogate models. The most likely explanation for this is that the network architecture has not been sufficiently optimised for surrogate modelling, lacking the ability for a good generalisation model. Although varying grain designs are part of the equations used in the physics loss, the PINN and I-PINN are most likely not able to capture this trend.

20.3.3. Chamber Lengths Group

After the grain design group was examined, the various modelling methods' performance in surrogate modelling will be assessed by removing various chamber length groups from the data sets used during the training phase. In Figure 20.21 and Figure 20.22, the average RMSE for the chamber pressure and thrust predictions are presented.

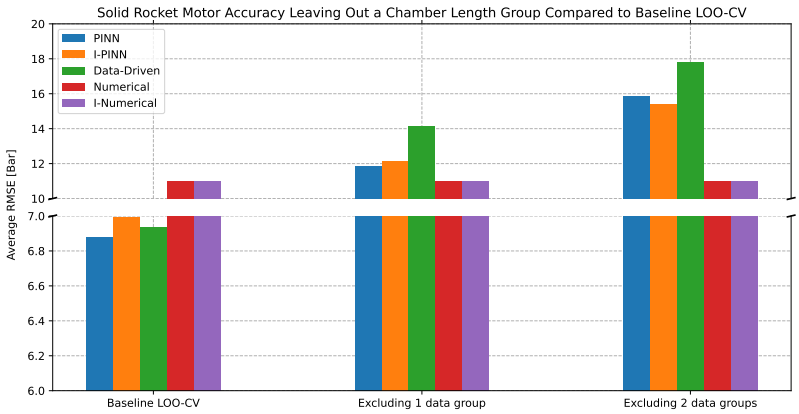


Figure 20.21: Bar chart presenting the average RMSE of the chamber pressure predictions against various data partitioning strategies for the chamber length groups for the various modelling methods

From Figure 20.21, it can be seen that when excluding 1 data group, the PINN and I-PINN have a decrease in accuracy compared to the baseline LOO-CV. The PINN and I-PINN have an unsatisfactory accuracy when excluding 2 data groups compared to the baseline LOO-CV, with a similar performance compared to the purely Data-Driven model, which has an accuracy decrease of more than 2 compared to the baseline LOO-CV. Overall, the Numerical methods consistently outperform the methods involving a data-driven component, showing that in terms of average RMSE of chamber pressure predictions, the PINN and I-PINN modelling methods are not able to capture the general trend for surrogate modelling. In Figure 20.22, the RMSE for the thrust predictions is presented.

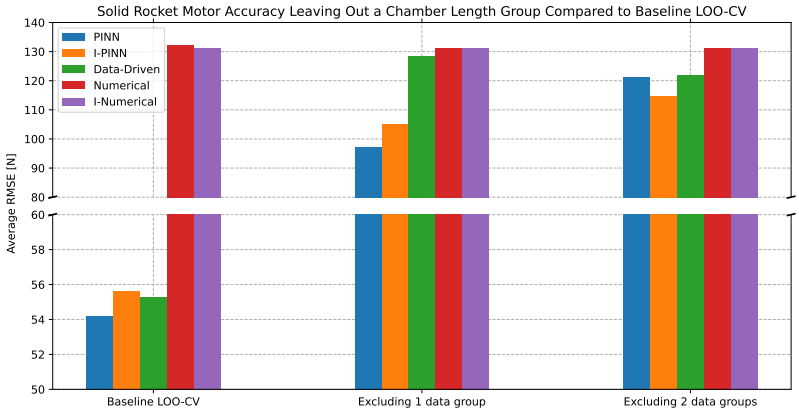


Figure 20.22: Bar chart presenting the average RMSE of the generated thrust predictions against various data partitioning strategies for the chamber length groups for the various modelling methods

From Figure 20.22, it can be seen that, similarly to the chamber pressure, the methods involving a data-driven component are not able to provide a similar accuracy level compared to the baseline LOO-CV. Furthermore, it can be observed that the purely Data-Driven method has a similar performance compared to the Numerical methods, while the PINN and I-PINN do have a slightly higher accuracy when excluding 1 data group during the training process. However, the decrease in accuracy is almost twice when excluding data groups during the training process for methods involving a data-driven component, as compared to the baseline LOO-CV. Overall, in terms of average RMSE for generated thrust predictions, the methods involving a data-driven approach are not able to capture the general trend and are not able to be used for surrogate modelling. Next, the average peak error for chamber

pressure and generated thrust will be examined in Figure 20.23 and Figure 20.24.

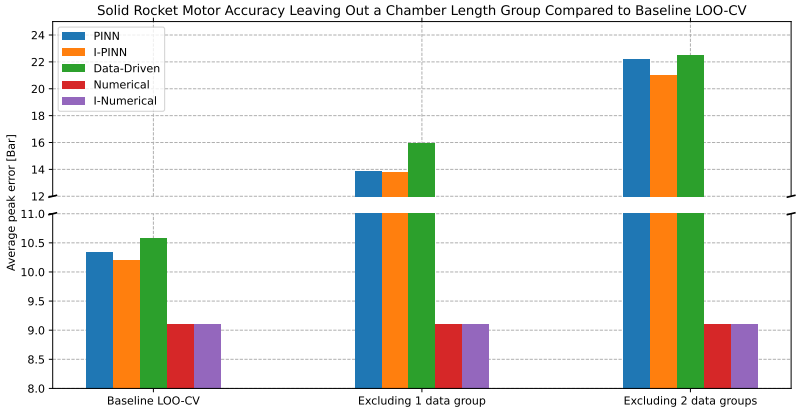


Figure 20.23: Bar chart presenting the average peak error of the chamber pressure predictions against various data partitioning strategies for the chamber length groups for the various modelling methods

From Figure 20.23, it can be seen that when leaving out 1 data group, the accuracy of the methods involving a data-driven component decreases compared to the baseline LOO-CV. When excluding 2 data groups, the methods involving a data-driven component do not have a satisfactory performance in terms of accuracy compared to the baseline LOO-CV. The Numerical methods consistently outperform the methods involving a data-driven component, as was the case with the baseline LOO-CV in terms of average peak error for chamber pressure predictions. This means that, in terms of average peak error for the chamber pressure predictions, the PINN and I-PINN are not suitable modelling methods for surrogate modelling. The average peak error in terms of generated thrust is presented in Figure 20.24

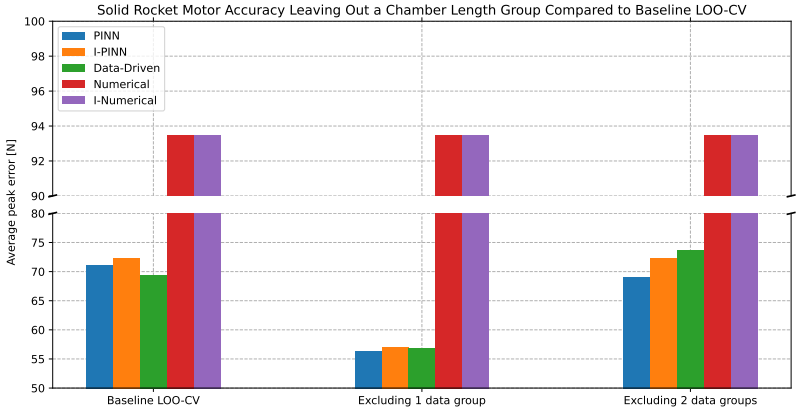


Figure 20.24: Bar chart presenting the average peak error of the generated thrust predictions against various data partitioning strategies for the chamber length groups for the various modelling methods

From Figure 20.24, a surprising observation is made that the methods involving a data-driven component have a similar or increase in accuracy compared to the baseline for the average peak error for generated thrust predictions. When excluding 1 data group during the training process, the accuracy of the methods increases, while the accuracy remains similar when excluding 2 data groups compared to the baseline LOO-CV. This means that in terms of surrogate modelling for the average peak error for generated thrust predictions, the PINN and I-PINN can be used for surrogate modelling operations. The purely Data-Driven model has a similar performance compared to the PINN and I-PINN, which

is likely caused by the fact that the PINN and I-PINN predictions are mainly being dictated by their data-driven part. Lastly, the average maximum error in chamber pressure and thrust predictions will be examined, making use of Figure 20.25 for chamber pressure and Figure 20.26 for generated thrust predictions.

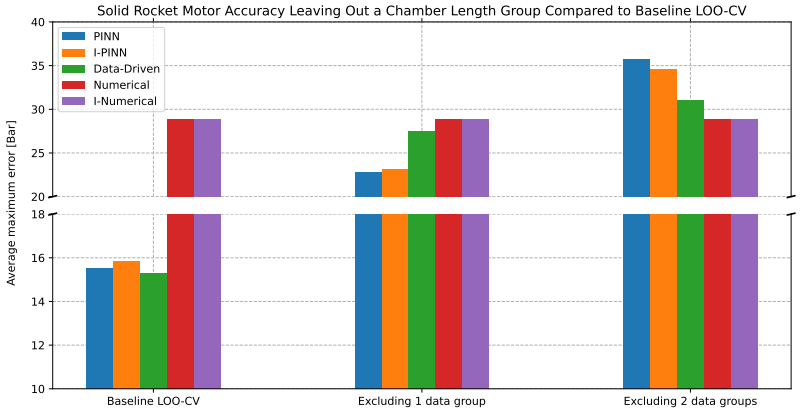


Figure 20.25: Bar chart presenting the average maximum error of the chamber pressure predictions against various data partitioning strategies for the chamber length groups for the various modelling methods

From Figure 20.25, it can be seen that compared to the baseline, the PINN and I-PINN have a decreased accuracy when excluding 1 data group. Furthermore, it can be seen that, when excluding 2 data groups during the training process, the PINN and I-PINN methods are the most inaccurate modelling methods, with a decrease in accuracy of more than 2 times compared to the baseline LOO-CV. The purely Data-Driven method has a slightly higher accuracy compared to the PINN and I-PINN when excluding 2 data groups, while the PINN and I-PINN do perform better when excluding a single data group. Therefore, it can be concluded that in terms of average maximum error for chamber pressure predictions, the PINN and I-PINN cannot be used for surrogate modelling operations. Lastly, the average maximum error of the generated thrust prediction will be examined in Figure 20.26.

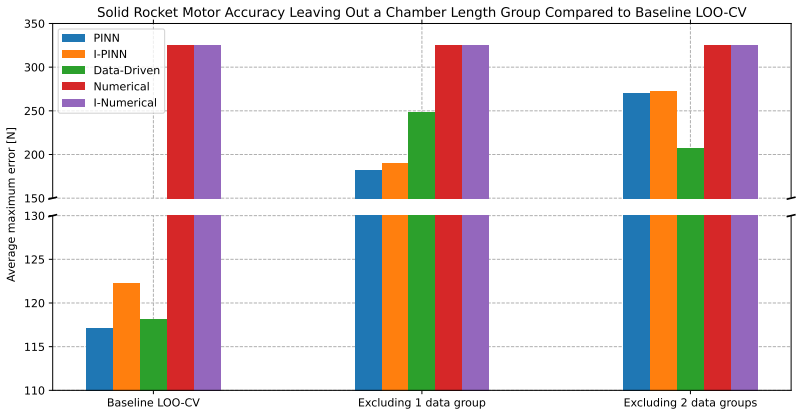


Figure 20.26: Bar chart presenting the average maximum error of the generated thrust predictions against various data partitioning strategies for the chamber length groups for the various modelling methods

From Figure 20.26, it can be seen that, similarly to the chamber pressure predictions, the accuracy of the PINN and I-PINN decreased when excluding data groups. Furthermore, the purely Data-Driven method has a higher accuracy when excluding 2 data groups compared to a single data group, and has

a higher accuracy compared to the PINN and I-PINN when excluding 2 data groups. However, overall, the methods involving a data-driven component have a decrease in accuracy of almost 2 times compared to the baseline LOO-CV. However, the methods involving a data-driven component do exceed the accuracy of the Numerical method. Overall, it was concluded that in terms of average maximum error for generated thrust predictions, the PINN and I-PINN are not suitable methods for surrogate modelling operations.

To conclude, when excluding various groups of chamber lengths during the training process, it was found that the PINN and I-PINN modelling methods are not able to act as surrogate modelling methods, having errors almost twice as high for the average RMSE and average maximum error compared to the standard LOO-CV baseline. Although the performances of the PINN and I-PINN show a better ability for surrogate modelling operations, with the average peak error for generated thrust predictions, having a very similar or sometimes even higher accuracy compared to the baseline, the total performance of the PINN and I-PINN has not been found satisfactory for surrogate modelling operations. The most likely explanation for this is that the network architecture has not been sufficiently optimised for surrogate modelling and accurate generalisation capability. Although varying chamber lengths are part of the equations used in the physics loss, the PINN and I-PINN are most likely not able to capture this trend, being dictated by their data-driven component, since the PINN and I-PINN seem to follow the purely Data-Driven model's trend.

20.3.4. Chamber Diameter Group

After the chamber length group was examined, the various modelling methods' performance for surrogate modelling will be assessed by removing various chamber diameter groups from the data sets used during the training phase. In Figure 20.27 and Figure 20.28, the average RMSE for the chamber pressure and thrust predictions are presented.

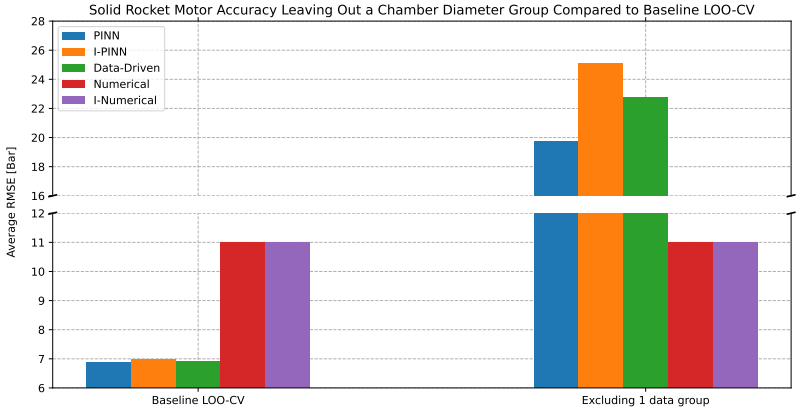


Figure 20.27: Bar chart presenting the average RMSE of the chamber pressure predictions against various data partitioning strategies for the chamber diameter groups for the various modelling methods

From Figure 20.27, it can be seen that when excluding 1 data group, all the methods involving a data-driven component have a decreased accuracy of more than 3 times compared to the baseline LOO-CV. While the PINN has a slightly higher accuracy compared to the I-PINN and purely Data-Driven method, it can be concluded that the Numerical methods consistently outperform the methods involving a data-driven component. This means that in terms of average RMSE of chamber pressure predictions, the PINN and I-PINN modelling methods are not suitable for use as a surrogate modelling method. In Figure 20.28, the RMSE for the thrust predictions is presented.

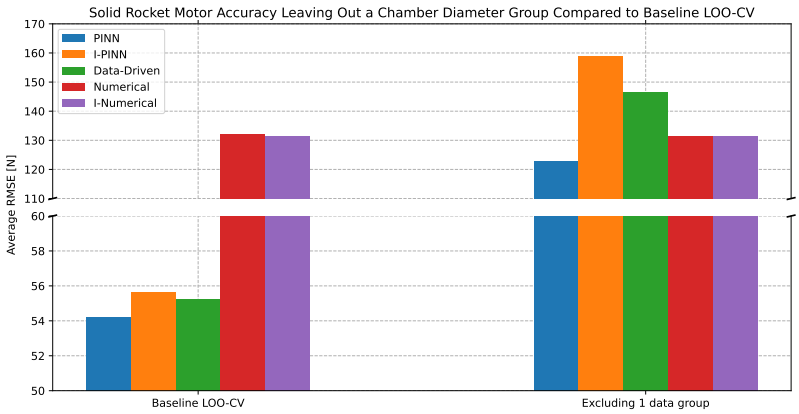


Figure 20.28: Bar chart presenting the average RMSE of the generated thrust predictions against various data partitioning strategies for the chamber diameter groups for the various modelling methods

From Figure 20.28, it can be seen that, similarly to the chamber pressure, the methods involving a data-driven component are not able to provide similar levels of accuracy compared to the baseline. Furthermore, it can be observed that the I-PINN is the most inaccurate modelling method, while the PINN has the highest accuracy. However, the decrease in accuracy is almost twice when excluding data groups during the training process for methods involving a data-driven component, as compared to the baseline. Overall, in terms of average RMSE, the methods involving a data-driven approach are not able to capture the general and can therefore not be used for surrogate modelling operations in terms of average RMSE for generated thrust prediction. Next, the average peak error for chamber pressure and generated thrust will be examined in Figure 20.29 and Figure 20.30.

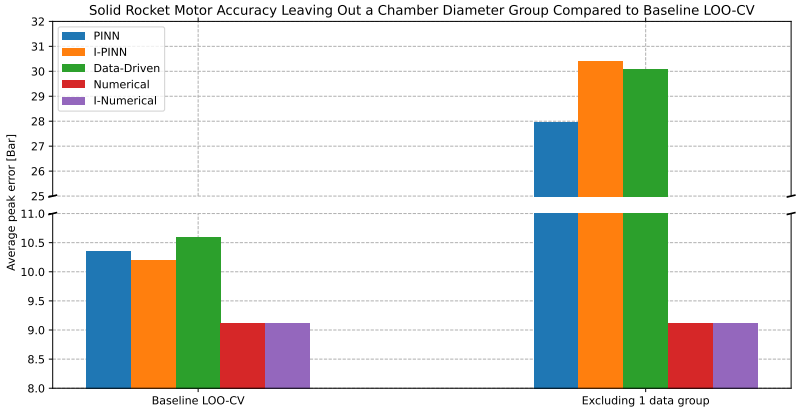


Figure 20.29: Bar chart presenting the average peak error of the chamber pressure predictions against various data partitioning strategies for the chamber diameter groups for the various modelling methods

From Figure 20.29, it can be seen that when leaving out 1 data group, the accuracy of the modelling methods involving a data-driven component decreases almost 2 times compared to the baseline LOO-CV in terms of average peak error for chamber pressure predictions. Furthermore, the Numerical methods consistently outperform the methods involving a data-driven component, as was the case with the baseline LOO-CV in terms of average peak error for chamber pressure and generated thrust predictions. This means that, in terms of average peak error for the chamber pressure predictions, the PINN and I-PINN are not suitable for surrogate modelling. The average peak error in terms of generated thrust is presented in Figure 20.30

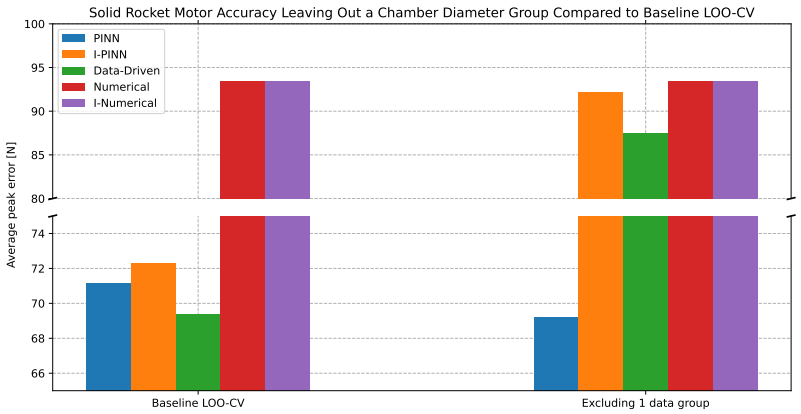


Figure 20.30: Bar chart presenting the average peak error of the generated thrust predictions against various data partitioning strategies for the chamber diameter groups for the various modelling methods

From Figure 20.30, a surprising observation can be made that while the I-PINN and purely Data-Driven methods' accuracy decreases to a similar accuracy level as the Numerical methods, the PINN has an increase in accuracy compared to the baseline LOO-CV. This means that in terms of surrogate modelling for the average peak error for generated thrust predictions, the PINN can be used for surrogate modelling operations. Lastly, the average maximum error in chamber pressure and thrust predictions will be examined, making use of Figure 20.31 for chamber pressure and Figure 20.32 for generated thrust predictions.

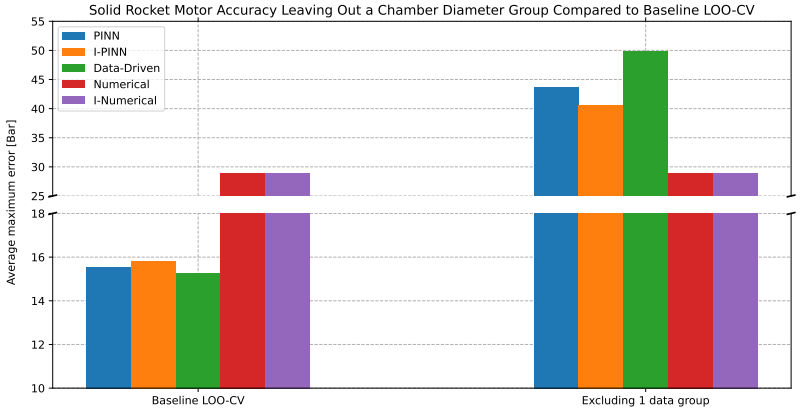


Figure 20.31: Bar chart presenting the average maximum error of the chamber pressure predictions against various data partitioning strategies for the chamber diameter groups for the various modelling methods

From Figure 20.31, it can be seen that compared to the baseline, the methods involving a data-driven component have a decrease in accuracy of almost 2 times when excluding data groups during the training process. While the Numerical methods are the most inaccurate modelling method of the baseline LOO-CV, the Numerical methods are the most accurate when excluding a data group. Therefore, it can be concluded that in terms of average maximum error for chamber pressure predictions, the PINN and I-PINN cannot be used for surrogate modelling operations. Lastly, the average maximum error of the generated thrust prediction will be examined in Figure 20.32.

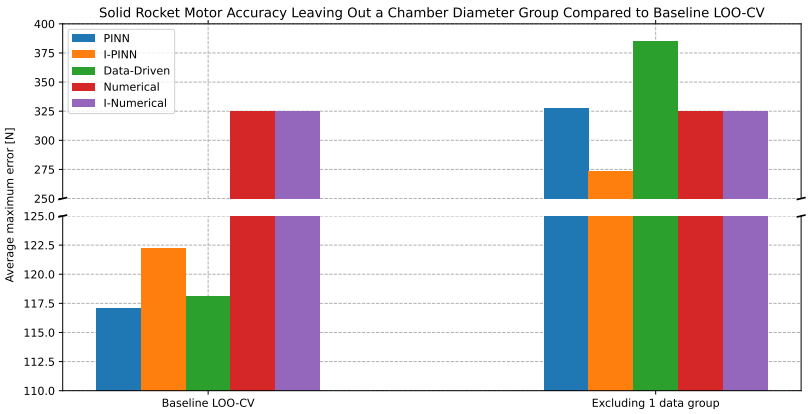


Figure 20.32: Bar chart presenting the average maximum error of the generated thrust predictions against various data partitioning strategies for the chamber diameter groups for the various modelling methods

From Figure 20.32, it can be seen that, similarly to the chamber pressure predictions, the accuracy of the methods involving a data-driven component decreases when excluding data groups. The accuracy decreases almost twice compared to the baseline, and even 3 times for the purely Data-Driven method. Overall, the methods involving a data-driven component have a decrease in accuracy of almost 2 times compared to the baseline. The methods involving a data-driven component have, however, a slightly decreased level of accuracy compared to the Numerical method. Overall, it was concluded that in terms of average maximum error for generated thrust predictions, the PINN and I-PINN are not suitable methods for surrogate modelling operations.

To conclude, when excluding various groups of chamber diameters during the training process, it was found that the PINN and I-PINN are not able to act as surrogate modelling methods, having errors almost twice as high for the average RMSE and average maximum error compared to the standard LOO-CV baseline. Although the performances of the PINN show a better ability for surrogate modelling operations, with the average peak error for generated thrust predictions having a higher accuracy compared to the baseline, the total performance of the PINN and I-PINN has not been found satisfactory for surrogate modelling operations. The most likely explanation for this is that the network architecture has not been sufficiently optimised for surrogate modelling. Although varying chamber diameters are part of the equations used in the physics loss, the PINN and I-PINN are most likely not able to capture this trend, being dictated by their data-driven component compared to their physics part, since the PINN and I-PINN seem to follow the purely Data-Driven model's trend.

20.3.5. Igniter Types Group

After the chamber diameter group was examined, the various modelling methods' performance in surrogate modelling will be assessed by removing various igniter type groups from the data sets used during the training phase. It should be noted that there is no physical model present in the physics loss, which models various igniter types, meaning that this will be captured by making use of the data-driven component of the PINN and I-PINN. In Figure 20.33 and Figure 20.34, the average RMSE for the chamber pressure and thrust predictions are presented.

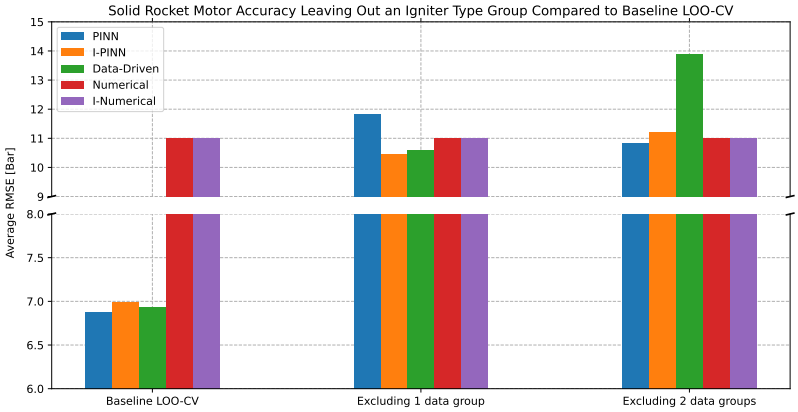


Figure 20.33: Bar chart presenting the average RMSE of the chamber pressure predictions against various data partitioning strategies for the igniter type groups for the various modelling methods

From Figure 20.33, it can be seen that when excluding 1 data group, all the methods involving a data-driven component have a decreased accuracy compared to the baseline LOO-CV. However, when excluding 2 groups from the training process, the purely Data-Driven method has a much larger decrease in accuracy compared to the PINN and I-PINN, which have a similar accuracy when compared against the Numerical methods. Overall, in terms of average RMSE of chamber pressure predictions, the PINN and I-PINN modelling are not suitable for use as surrogate modelling methods. In Figure 20.34, the RMSE for the thrust predictions is presented.

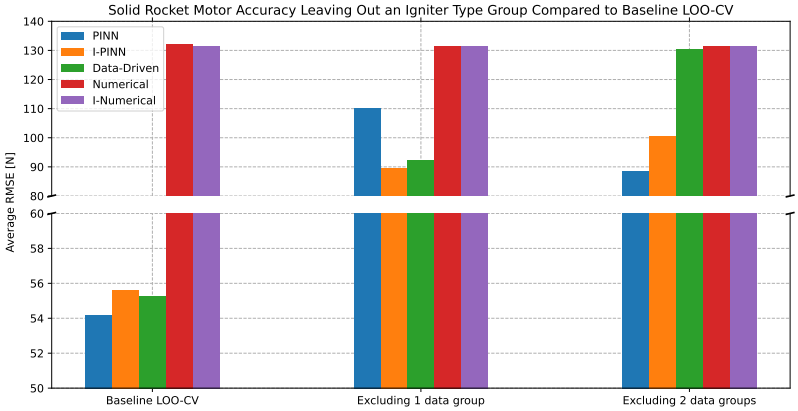


Figure 20.34: Bar chart presenting the average RMSE of the generated thrust predictions against various data partitioning strategies for the igniter type groups for the various modelling methods

From Figure 20.34, it can be seen that, similarly to the chamber pressure, the methods involving a data-driven component are not able to provide similar accuracy compared to the baseline LOO-CV. Furthermore, it can be observed that while the PINN is the most inaccurate modelling method when excluding 1 data group, the purely Data-Driven method is the most inaccurate modelling method when excluding 2 data groups, while the PINN is the most accurate modelling method when excluding 2 data groups compared to 1 data group. However, the methods involving a data-driven component do have a better performance compared to the Numerical models. Overall, in terms of average RMSE, the methods involving a data-driven approach cannot able to be used for surrogate modelling. Next, the average peak error for chamber pressure and generated thrust will be examined in Figure 20.35 and Figure 20.36.

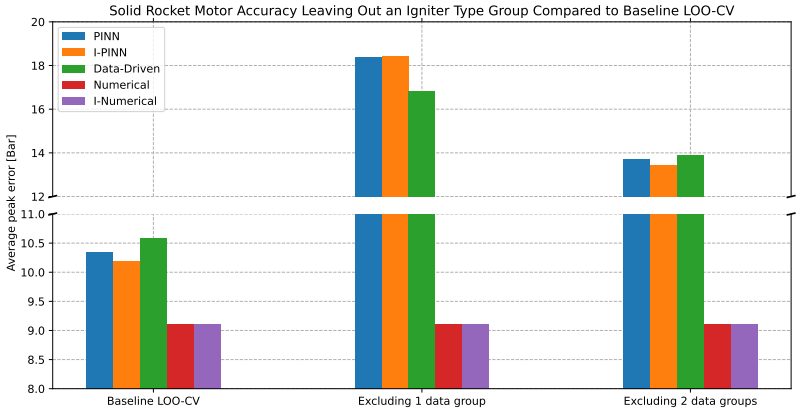


Figure 20.35: Bar chart presenting the average peak error of the chamber pressure predictions against various data partitioning strategies for the igniter type groups for the various modelling methods

From Figure 20.35, it can be seen that when leaving out 1 data group, the performance of the methods involving a data-driven component decreases almost 2 times compared to the baseline LOO-CV in terms of accuracy. Furthermore, the Numerical methods consistently outperform the methods involving a data-driven component, as was the case with the baseline LOO-CV in terms of average peak error for chamber pressure and generated thrust predictions. However, when comparing the methods involving a data-driven component, excluding 2 data groups has a higher performance compared to excluding 1 data group. Overall, in terms of average peak error for the chamber pressure predictions, the PINN and I-PINN are not suitable modelling methods for surrogate modelling operations. The average peak error in terms of generated thrust is presented in Figure 20.36

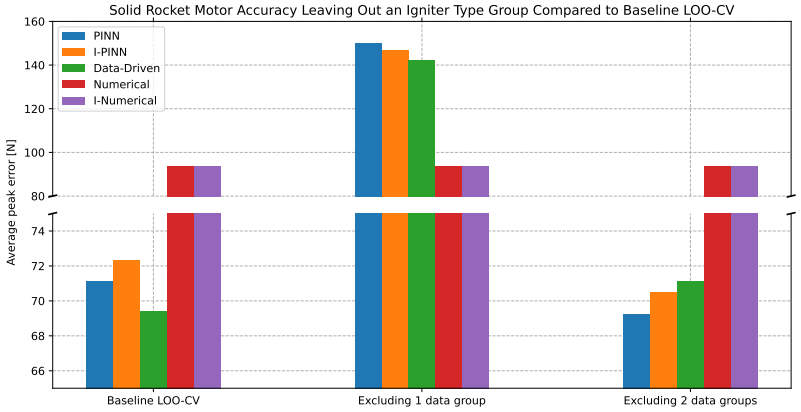


Figure 20.36: Bar chart presenting the average peak error of the generated thrust predictions against various data partitioning strategies for the igniter type groups for the various modelling methods

From Figure 20.36, it can be seen that, while the performance of all the methods involving a data-driven component has a decrease in accuracy when excluding 1 data group compared to the baseline LOO-CV, when excluding 2 data groups, the accuracy is similar to the baseline LOO-CV. While the PINN is the most inaccurate modelling method when excluding 1 data group, it has the best performance when excluding 2 data groups. Overall, in terms of surrogate modelling for the average peak error for generated thrust predictions, it can be concluded that the PINN and I-PINN cannot be used for surrogate modelling operations. Lastly, the average maximum error in chamber pressure and thrust predictions will be examined, making use of Figure 20.37 for chamber pressure and Figure 20.38 for

generated thrust predictions.

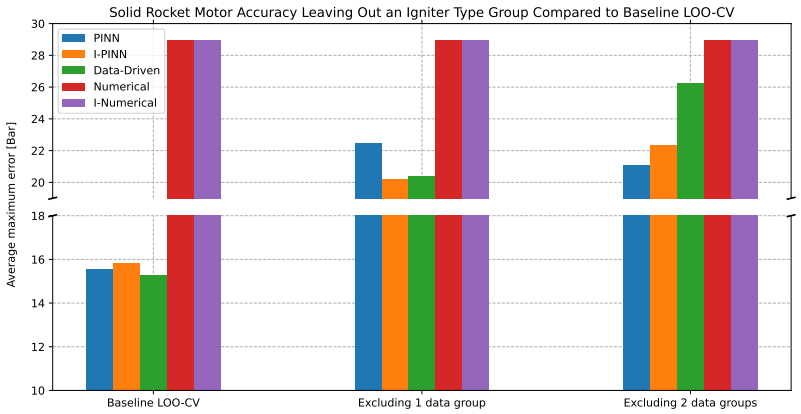


Figure 20.37: Bar chart presenting the average maximum error of the chamber pressure predictions against various data partitioning strategies for the igniter type groups for the various modelling methods

From Figure 20.37, it can be seen that compared to the baseline, the methods involving a data-driven component have a decrease in accuracy when excluding 1 and 2 data groups. The PINN and I-PINN performances stay relatively similar, while the purely Data-Driven method has a decrease in accuracy when excluding additional data groups. The Numerical methods are the most inaccurate modelling methods of all the modelling methods. Therefore, it can be concluded that in terms of average maximum error for chamber pressure predictions, the PINN and I-PINN can be used for surrogate modelling operations, since their performance only slightly decreases when excluding data groups during the training phase. Lastly, the average maximum error of the generated thrust prediction will be examined in Figure 20.38.

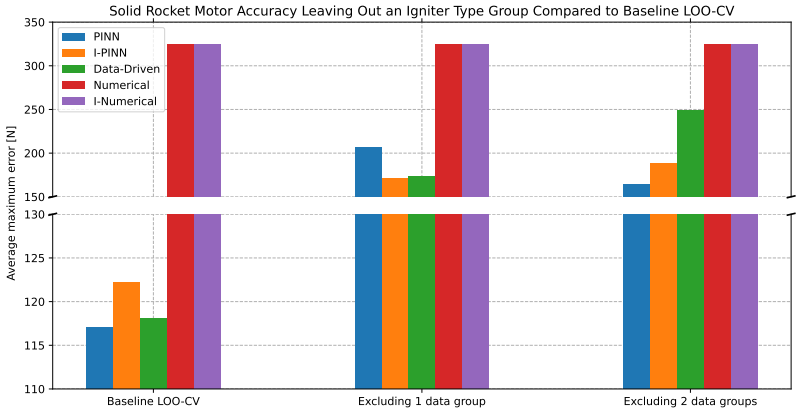


Figure 20.38: Bar chart presenting the average maximum error of the generated thrust predictions against various data partitioning strategies for the igniter type groups for the various modelling methods

From Figure 20.38, it can be seen that, similarly to the chamber pressure predictions, the accuracy of the methods involving a data-driven component decreases when excluding data groups. While the PINN has the worst performance of the methods involving a data-driven component when excluding a single data group, the purely Data-Driven method is the most inaccurate modelling method when excluding 2 data groups from the training phase. The methods involving a data-driven component do exceed the accuracy of the Numerical methods. Overall, it was concluded that in terms of average max-

imum error for generated thrust predictions, the PINN and I-PINN are suitable methods for surrogate modelling operations.

To conclude, when excluding various groups of igniter types during the training process, it was found that the PINN and I-PINN are not able to act as surrogate modelling methods, even though they are in terms of average maximum error for chamber pressure and generated thrust predictions. It is interesting to note that, even though the PINN and I-PINN do not have a physical model for igniter types, they do outperform the purely Data-Driven model most of the time. The most likely reason for this is that while the Data-Driven model does not perform well over the entire domain, the PINN and I-PINN will most likely be able to better model the parts of the domain not covered by the training data points, which results in better predictions.

20.3.6. Coating Strategy Group

After the igniter type group was examined, the various modelling methods' performance in surrogate modelling will be assessed by removing various coating strategy groups from the data sets used during the training phase. It should be noted that there is no physical model present in the physics loss, which models various coating strategies, meaning that this will be captured by making use of the data-driven component of the PINN and I-PINN. In Figure 20.39 and Figure 20.40, the average RMSE for the chamber pressure and thrust predictions are presented.

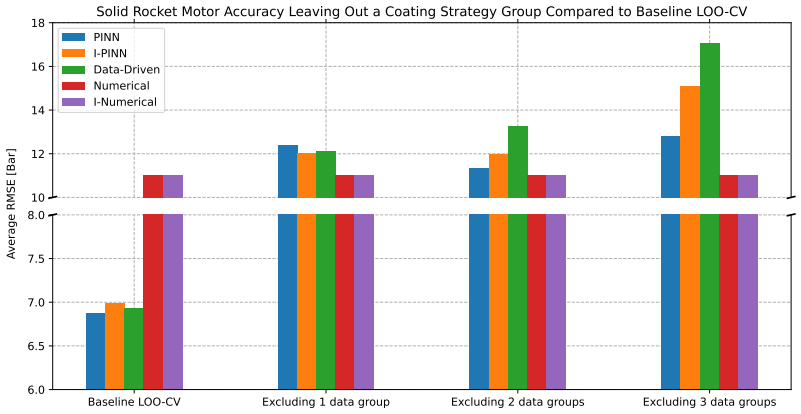


Figure 20.39: Bar chart presenting the average RMSE of the chamber pressure predictions against various data partitioning strategies for the coating strategy groups for the various modelling methods

From Figure 20.39, it can be seen that when excluding 1 data group, all the methods involving a data-driven component only have a slight decrease in accuracy compared to the baseline LOO-CV, having a similar accuracy level compared to the Numerical Methods. However, when excluding more groups from the training process, the purely Data-Driven method has a much larger decrease in accuracy compared to the PINN and I-PINN, which have a similar performance compared to the Numerical methods. Overall, in terms of average RMSE of chamber pressure predictions, the PINN and I-PINN modelling methods are not suitable for use as a surrogate modelling method. In Figure 20.40, the RMSE for the thrust predictions is presented.

Figure 20.40: Bar chart presenting the average RMSE of the generated thrust predictions against various data partitioning strategies for the coating strategy groups for the various modelling methods

From Figure 20.40, it can be seen that, similarly to the chamber pressure, the methods involving a data-driven component are not able to provide similar accuracy compared to the baseline LOO-CV, having a decrease in accuracy of about 2 times. Furthermore, it can be observed that while the PINN performs the worst when excluding 1 data group, the purely Data-Driven method is the most inaccurate modelling method when excluding more data groups, while the PINN will have a higher accuracy when excluding more data groups compared to 1 data group. However, the methods involving a data-driven component do have a better performance compared to the Numerical models. Overall, in terms of average RMSE, the methods involving a data-driven component cannot be used for surrogate modelling operations. Next, the average peak error for chamber pressure and generated thrust will be examined in Figure 20.41 and Figure 20.42.

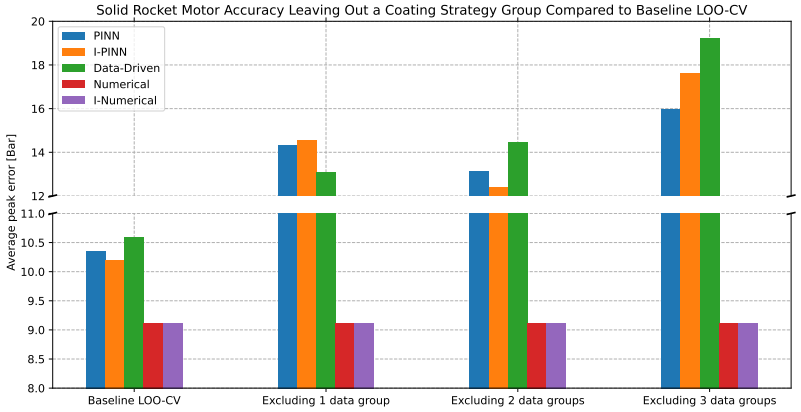


Figure 20.41: Bar chart presenting the average peak error of the chamber pressure predictions against various data partitioning strategies for the coating strategy groups for the various modelling methods

From Figure 20.41, it can be seen that when leaving out 1 data group, the performance of the methods involving a data-driven component has only a slight decrease in accuracy compared to the baseline LOO-CV. Only when excluding 3 data groups, the methods involving a data-driven component have an unsatisfactory decrease in accuracy. Furthermore, the Numerical methods consistently outperform the methods involving a data-driven component, as was the case with the baseline LOO-CV in terms of average peak error for chamber pressure and generated thrust predictions. However, when comparing the methods involving a data-driven component, excluding 2 data groups has a higher performance

compared to excluding 1 data group, while excluding 3 data groups leads to the most inaccurate modelling results. Overall, in terms of average peak error for the chamber pressure predictions, the PINN and I-PINN are not suitable for surrogate modelling. The average peak error in terms of generated thrust is presented in Figure 20.42

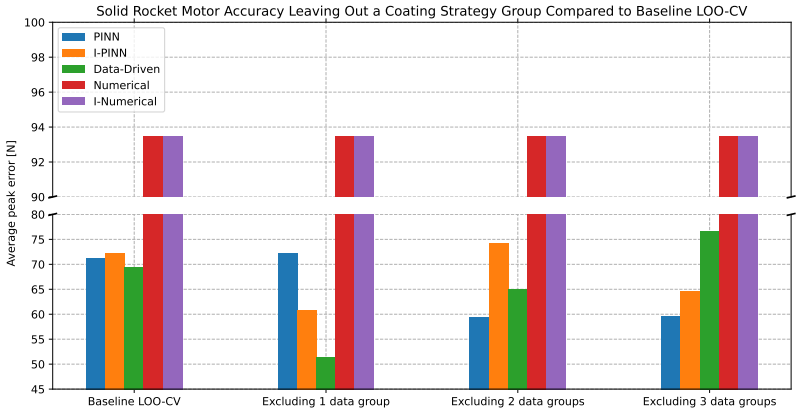


Figure 20.42: Bar chart presenting the average peak error of the generated thrust predictions against various data partitioning strategies for the coating strategy groups for the various modelling methods

From Figure 20.42, some interesting observations can be made. When excluding 1 data group during the training phase, it can be seen that the accuracy compared to the baseline LOO-CV increases for all the modelling methods, except for the PINN model. However, when increasing the number of data groups excluded during the training phase, the accuracy of the PINN increases. The purely Data-Driven method, on the other hand, has the best accuracy when including 1 data group, while the performance decreases when excluding more data groups. As with the baseline, the Numerical methods have the worst performance. Overall, in terms of surrogate modelling for the average peak error for generated thrust predictions, it can be concluded that the PINN and I-PINN can be used for surrogate modelling operations. Lastly, the average maximum error in chamber pressure and thrust predictions will be examined, making use of Figure 20.43 for chamber pressure and Figure 20.44 for generated thrust predictions.

Figure 20.43: Bar chart presenting the average maximum error of the chamber pressure predictions against various data partitioning strategies for the coating strategy groups for the various modelling methods

From Figure 20.43, it can be seen that compared to the baseline, the methods involving a data-driven

component have a decrease in accuracy when excluding data groups during the training process. When 3 data groups are excluded during the training process, the decrease in accuracy is almost twice compared of the baseline LOO-CV. While the PINN and I-PINN do show a better accuracy level when excluding 2 data groups compared to the purely Data-Driven method, it can be concluded that the methods involving a data-driven component are coupled. Furthermore, it can be seen that the Numerical methods are the most inaccurate modelling methods of all the modelling methods. Overall, it can be concluded that in terms of average maximum error for chamber pressure predictions, the PINN and I-PINN cannot be used for surrogate modelling operations. Lastly, the average maximum error of the generated thrust prediction will be examined in Figure 20.44.

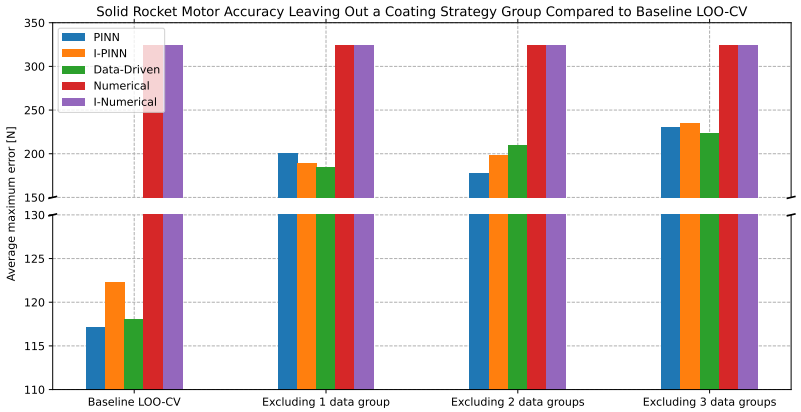


Figure 20.44: Bar chart presenting the average maximum error of the generated thrust predictions against various data partitioning strategies for the coating strategy groups for the various modelling methods

From Figure 20.44, it can be seen that, similarly to the chamber pressure predictions, the accuracy of the methods involving a data-driven component decreases compared to the baseline. However, when excluding more data groups, it can be seen that this decrease remains relatively constant, varying between 180 and 225 [N] for the various methods. Furthermore, it can be seen that each method will have the highest and lowest accuracy compared to each other when excluding more data groups for the methods involving a data-driven component. However, comparing the methods involving a data-driven component to the Numerical methods, the Numerical methods are the most inaccurate modelling method of the considered modelling methods. Overall, it was concluded that in terms of average maximum error for generated thrust predictions, the PINN and I-PINN are not suitable methods for surrogate modelling operations.

To conclude, when excluding various groups of coating strategies during the training process, it was found that the PINN and I-PINN are not able to act as surrogate modelling methods.

20.3.7. Number of Grains Group

After the coating strategy group was examined, the various modelling methods' performance in surrogate modelling will be assessed by removing various number of grains groups from the data sets used during the training phase. In Figure 20.45 and Figure 20.46, the average RMSE for the chamber pressure and thrust predictions are presented.

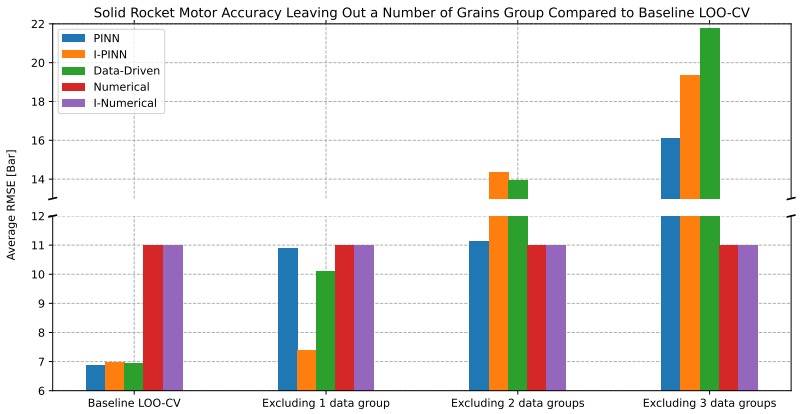


Figure 20.45: Bar chart presenting the average RMSE of the chamber pressure predictions against various data partitioning strategies for the number of grains groups for the various modelling methods

From Figure 20.45, it can be seen that when excluding 1 data group, the I-PINN shows only a slight decrease in accuracy level, while the other methods involving a data-driven component have a larger decrease in accuracy. When excluding more data groups, the methods involving a data-driven component have a higher decrease in accuracy, having a much lower accuracy level compared to the Numerical methods. When excluding 3 data groups, the purely Data-Driven method has a decrease in accuracy level of almost 3 times compared to the baseline LOO-CV. Overall, in terms of average RMSE of chamber pressure predictions, the PINN and I-PINN modelling methods are not suitable for use as a surrogate modelling method. In Figure 20.46, the RMSE for the thrust predictions is presented.

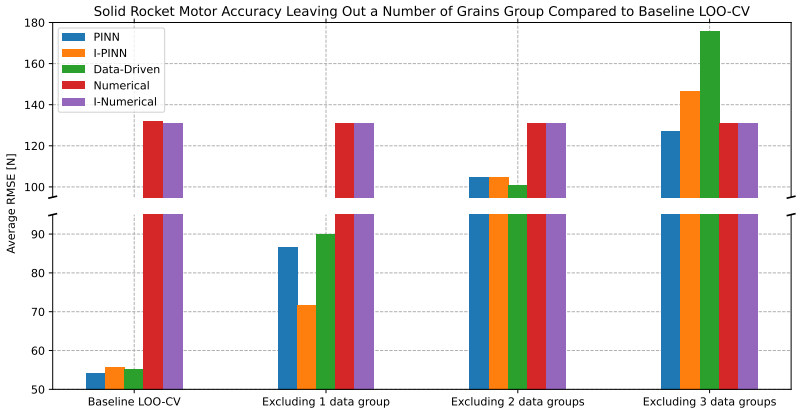


Figure 20.46: Bar chart presenting the average RMSE of the generated thrust predictions against various data partitioning strategies for the number of grains groups for the various modelling methods

From Figure 20.46, it can be seen that, similarly to the chamber pressure, the methods involving a data-driven component only have a slight decrease in accuracy when excluding 1 data group. However, when excluding more data groups, the methods involving a data-driven component have a higher accuracy level compared to the Numerical models. However, when excluding 3 data groups, the PINN and purely Data-Driven model are the most inaccurate modelling methods. Similarly to the chamber pressure predictions, the purely Data-Driven method has a decrease in accuracy of about 3 times compared to the baseline LOO-CV when excluding 3 data groups during the training process. Overall, in terms of average RMSE, the methods involving a data-driven component cannot able to be used for surrogate modelling. Next, the average peak error for chamber pressure and generated thrust will be

examined in Figure 20.47 and Figure 20.48.

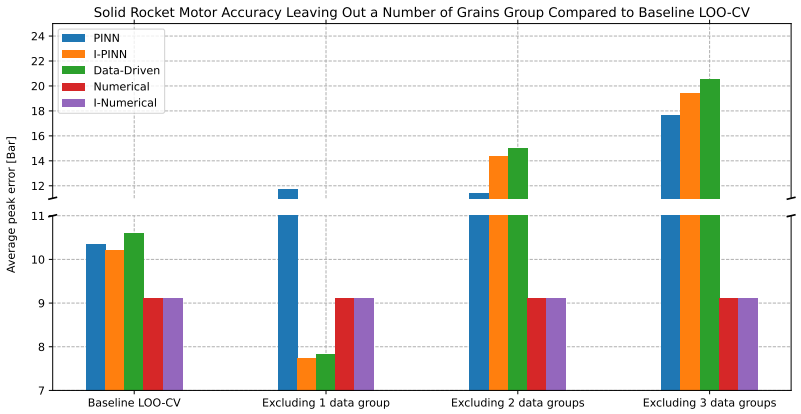


Figure 20.47: Bar chart presenting the average peak error of the chamber pressure predictions against various data partitioning strategies for the number of grains groups for the various modelling methods

From Figure 20.47, it can be seen that when leaving out 1 data group, the PINN and purely Data-Driven method show an increase in accuracy compared to the baseline. However, the PINN, on the other hand, shows a decrease in accuracy. When excluding more data groups, the methods involving a data-driven component have a lower accuracy level compared to the Numerical methods, having a decrease in accuracy of almost two times compared to the baseline LOO-CV when comparing it to the purely Data-Driven model. Overall, in terms of average peak error for the chamber pressure predictions, the PINN and I-PINN are not suitable for surrogate modelling. The average peak error in terms of generated thrust is presented in Figure 20.48.

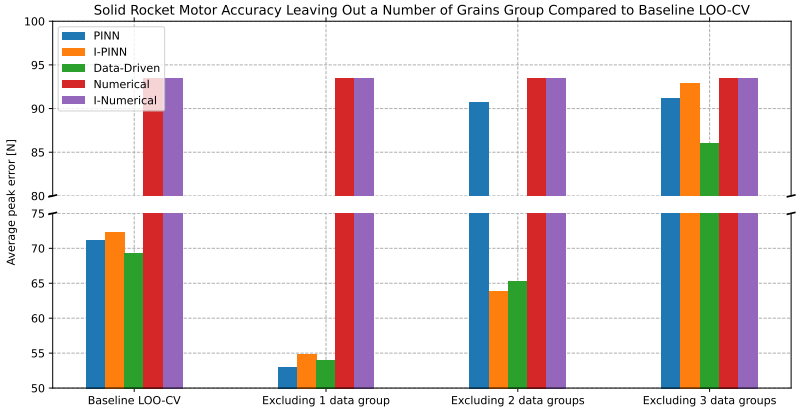


Figure 20.48: Bar chart presenting the average peak error of the generated thrust predictions against various data partitioning strategies for the number of grains groups for the various modelling methods

From Figure 20.48, some interesting observations can be made. When excluding 1 data group during the training phase, it can be seen that the accuracy levels compared to the baseline increase for all the modelling methods. When increasing the number of data groups, the I-PINN and purely Data-Driven models have a slight decrease in accuracy compared to the exclusion of 1 data group, but have an improved accuracy compared to the baseline LOO-CV. The PINN, on the other hand, is the most inaccurate modelling method compared to the baseline LOO-CV, having a similar accuracy level when compared to the Numerical Models. When excluding 3 data groups, the accuracy of the methods involv-

ing a data-driven component are similar to the accuracy level of the Numerical model. However, overall, there is only a slight decrease in accuracy level for the modelling methods involving a data-driven component, meaning that in terms of surrogate modelling for the average peak error for generated thrust predictions, it can be concluded that the I-PINN can be used for surrogate modelling operations. Lastly, the average maximum error in chamber pressure and thrust predictions will be examined, making use of Figure 20.49 for chamber pressure and Figure 20.50 for generated thrust predictions.

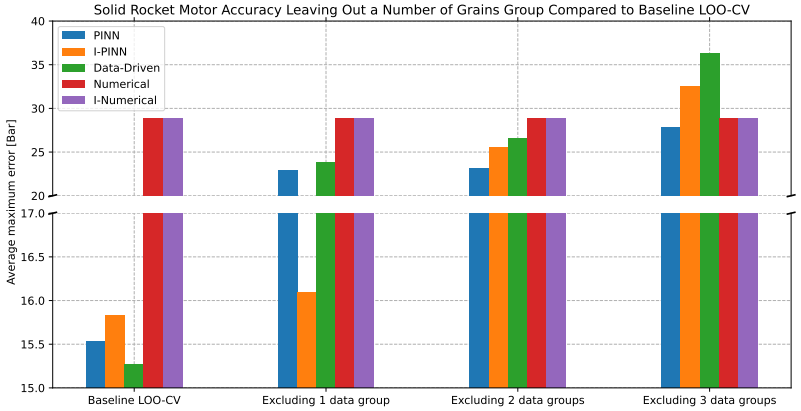


Figure 20.49: Bar chart presenting the average maximum error of the chamber pressure predictions against various data partitioning strategies for the number of grains groups for the various modelling methods

From Figure 20.49, it can be seen that compared to the baseline, the methods involving a data-driven component have a decrease in accuracy when excluding data groups during the training process. When 3 data groups are excluded during the training process, the decrease in performance is almost twice compared of the baseline LOO-CV. While the I-PINN does have a similar performance compared to the baseline LOO-CV when excluding 1 data group, it can overall be concluded that the methods involving a data-driven component are coupled to each other. Furthermore, it can be seen that the Numerical methods are the most inaccurate of all the modelling methods, except when 3 data groups are excluded from the training process. Overall, it can be concluded that in terms of average maximum error for chamber pressure predictions, the PINN and I-PINN cannot be used for surrogate modelling operations. Lastly, the average maximum error of the generated thrust prediction will be examined in Figure 20.50.

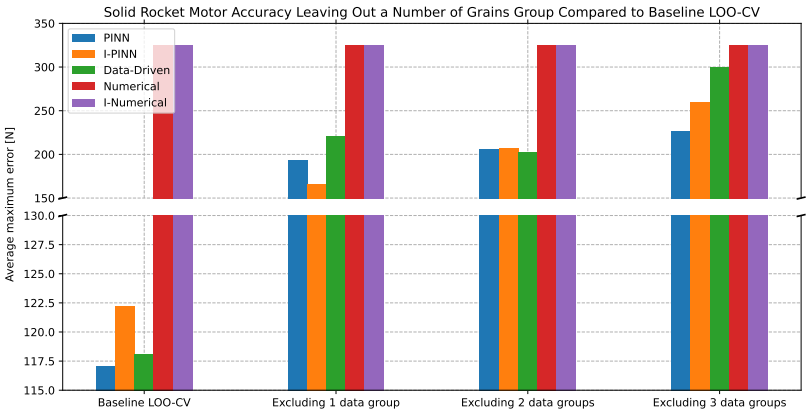


Figure 20.50: Bar chart presenting the average maximum error of the generated thrust predictions against various data partitioning strategies for the number of grains groups for the various modelling methods

From Figure 20.50, it can be seen that, similarly to the chamber pressure predictions, the accuracy of the methods involving a data-driven component decreases compared to the baseline LOO-CV. While excluding 1 data group during the training phase results in a relatively low level of decrease in accuracy for the I-PINN, the accuracy of the PINN and purely Data-Driven method decreases almost twice compared to the baseline LOO-CV. When excluding 2 data groups, the accuracy levels remain relatively similar compared to the exclusion of a single data group, however, when excluding 3 data groups, the purely Data-Driven model has a decreases in accuracy of almost 3 times compared to the baseline LOO-CV, while the PINN has a similar accuracy level when excluding 1 and 2 data groups during the training phase. However, comparing the methods involving a data-driven component to the Numerical methods, the Numerical methods are the most inaccurate modelling methods. Overall, it was concluded that in terms of average maximum error for generated thrust predictions, the PINN and I-PINN are not suitable methods for surrogate modelling operations.

To conclude, when excluding various groups of number of grains during the training process, it was found that the PINN and I-PINN are not able to act as surrogate modelling methods, even though the I-PINN does show this ability in terms of average peak error for generated thrust predictions.

To sum up, when examining all the surrogate modelling runs, as described in section 20.3, it can be concluded that the PINN and I-PINN are currently not suitable to be used as surrogate models for SRM modelling. This conclusion is mainly drawn by the fact that the accuracy of the PINN and I-PINN decreases when leaving out data groups during the training phase compared to a standard LOO-CV baseline. There could be various reasons for these observations, but the most likely ones are summed up in the list below:

- NN architecture optimisation
  The NN architecture has been created by making use of a trial-and-error approach, as well as making use of educated guesses. However, while this showed promising results for the LOO-CV process in terms of accuracy compared to the Numerical models, it was found that there was not much difference compared to the purely Data-Driven model. Therefore, it is likely that the PINN is mostly being dictated by the data-driven part of the PINN, meaning that if a purely Data-Driven method is not able to capture the general trend, the PINN would likely also not be able to perform well in terms of surrogate modelling.
- Scaling
  While the PINN developed for the Cooling Down of Object showed promising results in terms of surrogate modelling, as presented in section 9.1, there are no indications that the PINN method can perform sufficiently well in terms of surrogate modelling for SRM. It should be noted, however, that a difference between the two methods is that the methods involving a data-driven component for the SRM comparison have scaled inputs, while the inputs for the Cooling Down of Objects methods involving a data-driven component are unscaled. Although it is verified that scaling has been performed correctly, it could be that by scaling, issues are being introduced in the physics and boundary losses compared to an unscaled method. It is therefore advised to perform additional research investigating scaling factors for surrogate modelling of PINNs.
- Amount of inputs
  While the inputs for the PINN model used for the Cooling Down of Objects only make use of 7 inputs, as was explained in chapter 8, the SRM PINN model makes use of 20 inputs. Besides the fact that it was shown that an unscaled version was not able to converge, it could be possible that there is a limit to the number of inputs that can provide satisfactory results. There possibly is a relation between the number of inputs and the NN architecture, however, it is advised to perform more research related to the number of inputs and surrogate modelling capabilities of PINNs.
- Data used during training
  While certain predictions of experiments provide correct predictions, making use of the LOO-CV method, others are not able to provide satisfactory results. The most likely reason for this is that the model has not captured the general trend and is therefore not able to generalise correctly for new unseen inputs. This means that, most likely, the reason why certain predictions for tests work and others are not able to provide meaningful predictions could be caused by the training data used. While certain parameters have been seen during the training, being able to generalise the

model, in other cases, the model has not seen parts as inputs, therefore not able to generalise, which causes unsatisfactory results.

20.3.8. Data Table Surrogate Modelling

The detailed outcomes of the surrogate test runs for SRM modelling are summarised in Table 20.2.

Table 20.2: Comparison of Cooling Down of Objects for surrogate modelling: Computational Efficiency, Data Efficiency, and Accuracy across the various used methods

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Computational Efficiency					
Train Time (50 data points per file, excluding 1 group of nozzle sizes) [s]	411.358460	408.556584	15.189051	x	x
Train Time (50 data points per file, excluding 2 groups of nozzle sizes) [s]	284.876157	280.646706	12.002722	x	x
Train Time (50 data points per file, excluding 3 groups of nozzle sizes) [s]	140.684531	138.625663	8.247657	x	x
Train Time (50 data points per file, excluding 1 group of grain designs) [s]	377.525802	373.126276	14.156336	x	x
Train Time (50 data points per file, excluding 2 groups of grain designs) [s]	184.348224	185.395806	9.402566	x	x
Train Time (50 data points per file, excluding 1 group of chamber lengths) [s]	412.153429	405.516001	22.874637	x	x
Train Time (50 data points per file, excluding 2 groups of chamber lengths) [s]	209.160551	203.943100	9.807385	x	x
Train Time (50 data points per file, excluding 1 group of chamber diameter) [s]	360.435110	358.902224	17.710201	x	x
Train Time (50 data points per file, excluding 1 group of igniter types) [s]	237.361738	236.089137	12.603852	x	x
Train Time (50 data points per file, excluding 2 groups of igniter types) [s]	120.210394	119.341456	7.955949	x	x
Train Time (50 data points per file, excluding 1 group of coating strategy) [s]	456.808200	458.375523	24.197470	x	x
Train Time (50 data points per file, excluding 2 groups of coating strategy) [s]	308.729031	308.594508	13.247327	x	x
Train Time (50 data points per file, excluding 3 groups of coating strategy) [s]	156.702514	155.214874	8.701397	x	x
Train Time (50 data points per file, excluding 1 group of number of grains) [s]	444.635712	445.558052	20.858796	x	x

Continued on next page

Table 20.2 – continued from previous page

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Train Time (50 data points per file, excluding 2 groups of number of grains) [s]	299.813111	298.831693	15.904169	x	x
Train Time (50 data points per file, excluding 3 groups of number of grains) [s]	152.253842	151.786563	8.429067	x	x
Execution Time (50 data points per file, excluding 1 group of nozzle sizes) [s]	0.128274	0.084186	0.078968	0.007035	0.007923
Execution Time (50 data points per file, excluding 2 groups of nozzle sizes) [s]	0.141895	0.096694	0.096316	0.008591	0.008880
Execution Time (50 data points per file, excluding 3 groups of nozzle sizes) [s]	0.116177	0.085634	0.085367	0.008077	0.006637
Execution Time (50 data points per file, excluding 1 group of grain designs) [s]	0.146252	0.089760	0.080335	0.009169	0.007973
Execution Time (50 data points per file, excluding 2 groups of grain designs) [s]	0.112109	0.089487	0.083339	0.006864	0.007518
Execution Time (50 data points per file, excluding 1 group of chamber lengths) [s]	0.103605	0.083219	0.082298	0.009749	0.007998
Execution Time (50 data points per file, excluding 2 groups of chamber lengths) [s]	0.098024	0.084075	0.081163	0.007489	0.006750
Execution Time (50 data points per file, excluding 1 group of chamber diameter) [s]	0.105247	0.093462	0.086880	0.007391	0.006763
Execution Time (50 data points per file, excluding 1 group of igniter types) [s]	0.113819	0.094032	0.093198	0.007763	0.007759
Execution Time (50 data points per file, excluding 2 groups of igniter types) [s]	0.120733	0.102895	0.092918	0.007204	0.007985
Execution Time (50 data points per file, excluding 1 group of coating strategy) [s]	0.107994	0.086328	0.080917	0.009336	0.007193
Execution Time (50 data points per file, excluding 2 groups of coating strategy) [s]	0.117189	0.093854	0.087546	0.008465	0.008573
Execution Time (50 data points per file, excluding 3 groups of coating strategy) [s]	0.107845	0.086837	0.082792	0.008812	0.008720

Continued on next page

Table 20.2 – continued from previous page

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Execution Time (50 data points per file, excluding 1 group of number of grains) [s]	0.094733	0.083520	0.080219	0.007062	0.006804
Execution Time (50 data points per file, excluding 2 groups of number of grains) [s]	0.099650	0.086019	0.082304	0.007441	0.007070
Execution Time (50 data points per file, excluding 3 groups of number of grains) [s]	0.117279	0.092992	0.090965	0.007083	0.007510
Accuracy					
Avg. RMSE Chamber pressure (using 50 data points per file) [bar]	6.878653	6.989631	6.93378	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file) [bar]	10.346630	10.199791	10.585910	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file) [bar]	15.5357267	15.8263626	15.272270	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file) [N]	54.198997	55.628319	55.245823	132.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file) [N]	71.1411727	72.308060	69.390418	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file) [N]	117.092485	122.272280	118.083813	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 1 group of nozzle sizes) [bar]	17.161296	21.625395	20.083075	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 1 group of nozzle sizes) [bar]	25.999967	27.732080	29.274979	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 1 group of nozzle sizes) [bar]	39.463253	35.131035	46.156847	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 1 group of nozzle sizes) [N]	95.392229	123.659306	117.535013	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 1 group of nozzle sizes) [N]	78.386621	100.182160	102.194104	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 1 group of nozzle sizes) [N]	286.180612	228.593958	353.039264	324.990327	324.990339

Continued on next page

Table 20.2 – continued from previous page

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 2 groups of nozzle sizes) [bar]	15.302988	27.614073	22.369539	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 2 groups of nozzle sizes) [bar]	20.202694	31.895262	28.495320	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 2 groups of nozzle sizes) [bar]	35.800327	45.539753	41.553196	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 2 groups of nozzle sizes) [N]	108.150584	145.437535	116.915103	131.211803	131.211906
Avg. Peak Error Thrust (using 50 data points per file, excluding 2 groups of nozzle sizes) [N]	96.101544	72.649534	87.874934	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 2 groups of nozzle sizes) [N]	269.686517	255.564758	251.642294	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 3 groups of nozzle sizes) [bar]	23.028272	21.253427	22.626081	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 3 groups of nozzle sizes) [bar]	26.139866	24.339445	24.144891	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 3 groups of nozzle sizes) [bar]	41.179832	43.043000	42.579023	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 3 groups of nozzle sizes) [N]	147.910511	157.239970	159.215145	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 3 groups of nozzle sizes) [N]	76.822576	76.893655	98.043682	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 3 groups of nozzle sizes) [N]	267.613064	300.243891	290.060612	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 1 group of grain designs) [bar]	10.640867	10.560724	23.643269	10.996256	10.996256

Continued on next page

Table 20.2 – continued from previous page

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 1 group of grain design) [bar]	10.224448	11.016989	18.204477	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 1 group of grain design) [bar]	20.137765	22.582818	41.251629	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 1 group of grain design) [N]	113.548153	81.292479	219.701800	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 1 group of grain design) [N]	129.741296	110.459786	106.988478	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 1 group of grain design) [N]	199.830367	178.921254	379.952263	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 2 groups of grain designs) [bar]	18.205647	19.876443	24.394089	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 2 groups of grain designs) [bar]	17.259492	17.481376	26.641102	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 2 groups of grain designs) [bar]	32.581764	35.633407	45.643654	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 2 groups of grain designs) [N]	143.613546	142.777736	195.228698	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 2 groups of grain designs) [N]	91.251890	101.656001	144.737309	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 2 groups of grain designs) [N]	247.540438	247.919992	348.300333	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 1 group of chamber lengths) [bar]	11.853360	12.126783	14.138275	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 1 group of chamber lengths) [bar]	13.870732	13.829910	15.942167	9.108076	9.108077

Continued on next page

Table 20.2 – continued from previous page

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 1 group of chamber lengths) [bar]	22.818291	23.124409	27.506149	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 1 group of chamber lengths) [N]	97.023010	104.916149	128.456064	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 1 group of chamber lengths) [N]	56.325631	57.028385	56.884107	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 1 group of chamber lengths) [N]	181.355265	189.878730	248.064779	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 2 groups of chamber lengths) [bar]	15.865748	15.405590	17.802625	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 2 groups of chamber lengths) [bar]	22.221721	21.041846	22.528234	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 2 groups of chamber lengths) [bar]	35.768879	34.596094	31.067074	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 2 groups of chamber lengths) [N]	121.277578	114.487203	121.739604	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 2 groups of chamber lengths) [N]	69.093361	72.267138	73.720664	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 2 groups of chamber lengths) [N]	270.247791	272.336766	206.822511	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 1 group of chamber diameter) [bar]	19.751183	25.132901	22.803544	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 1 group of chamber diameter) [bar]	27.965062	30.376352	30.081978	9.108076	9.108077

Continued on next page

Table 20.2 – continued from previous page

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 1 group of chamber diameter) [bar]	43.785219	40.668459	49.831073	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 1 group of chamber diameter) [N]	122.891482	158.890614	146.427547	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 1 group of chamber diameter) [N]	69.198574	92.173805	87.497782	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 1 group of chamber diameter) [N]	327.769156	273.869447	385.042240	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 1 group of igniter types) [bar]	11.834527	10.470212	10.582644	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 1 group of igniter types) [bar]	18.389863	18.415302	16.830044	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 1 group of igniter types) [bar]	22.444657	20.198648	20.377446	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 1 group of igniter types) [N]	110.199906	89.460871	92.122975	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 1 group of igniter types) [N]	150.079761	146.880585	142.124390	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 1 group of igniter types) [N]	206.785728	170.862083	173.124178	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 2 groups of igniter types) [bar]	10.842972	11.231010	13.894163	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 2 groups of igniter types) [bar]	13.708427	13.438124	13.914770	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 2 groups of igniter types) [bar]	21.081731	22.351831	26.203971	28.905221	28.905221

Continued on next page

Table 20.2 – continued from previous page

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Avg. RMSE Thrust (using 50 data points per file, excluding 2 groups of igniter types) [N]	88.511036	100.482657	130.182149	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 2 groups of igniter types) [N]	69.215079	70.486157	71.153194	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 2 groups of igniter types) [N]	164.354599	188.009468	248.584660	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 1 group of coating strategy) [bar]	12.374169	12.008924	12.100626	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 1 group of coating strategy) [bar]	14.296614	14.554694	13.056068	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 1 group of coating strategy) [bar]	23.542246	22.869799	22.981238	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 1 group of coating strategy) [N]	108.351816	103.957183	98.782396	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 1 group of coating strategy) [N]	72.233281	60.794345	51.437368	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 1 group of coating strategy) [N]	200.369400	188.864824	185.032701	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 2 groups of coating strategy) [bar]	11.335417	11.982703	13.276404	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 2 groups of coating strategy) [bar]	13.140694	12.386185	14.426287	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 2 groups of coating strategy) [bar]	22.109613	22.690672	25.151308	28.905221	28.905221

Continued on next page

Table 20.2 – continued from previous page

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Avg. RMSE Thrust (using 50 data points per file, excluding 2 groups of coating strategy) [N]	94.461562	103.740305	109.349724	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 2 groups of coating strategy) [N]	59.504002	74.289045	64.932980	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 2 groups of coating strategy) [N]	177.423097	198.766442	210.489998	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 3 groups of coating strategy) [bar]	12.819457	15.104532	17.041599	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 3 groups of coating strategy) [bar]	15.940703	17.600854	19.233457	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 3 groups of coating strategy) [bar]	28.567854	29.313559	29.113902	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 3 groups of coating strategy) [N]	108.415505	115.166484	128.656455	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 3 groups of coating strategy) [N]	59.677871	64.601861	76.538641	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 3 groups of coating strategy) [N]	229.967190	235.141158	224.072180	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 1 group of number of grains) [bar]	10.889547	7.393663	10.106777	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 1 group of number of grains) [bar]	11.752688	7.724500	7.836237	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 1 group of number of grains) [bar]	22.882446	16.093715	23.857086	28.905221	28.905221

Continued on next page

Table 20.2 – continued from previous page

Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Avg. RMSE Thrust (using 50 data points per file, excluding 1 group of number of grains) [N]	86.654689	71.597017	90.128865	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 1 group of number of grains) [N]	53.000768	54.851306	54.055055	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 1 group of number of grains) [N]	193.368612	165.301970	220.709307	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 2 groups of number of grains) [bar]	11.133324	14.357306	13.912879	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 2 groups of number of grains) [bar]	11.432014	14.388070	15.030171	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 2 groups of number of grains) [bar]	23.208623	25.605141	26.625655	28.905221	28.905221
Avg. RMSE Thrust (using 50 data points per file, excluding 2 groups of number of grains) [N]	104.709849	104.723904	100.821882	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 2 groups of number of grains) [N]	90.761776	63.862658	65.280910	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 2 groups of number of grains) [N]	205.379410	207.332788	202.126490	324.990327	324.990339
Avg. RMSE Chamber pressure (using 50 data points per file, excluding 3 groups of number of grains) [bar]	16.095618	19.345863	21.787425	10.996256	10.996256
Avg. Peak Error Chamber pressure (using 50 data points per file, excluding 3 groups of number of grains) [bar]	17.645766	19.468632	20.564072	9.108076	9.108077
Avg. Max Error Chamber pressure (using 50 data points per file, excluding 3 groups of number of grains) [bar]	27.889382	32.596787	36.299881	28.905221	28.905221

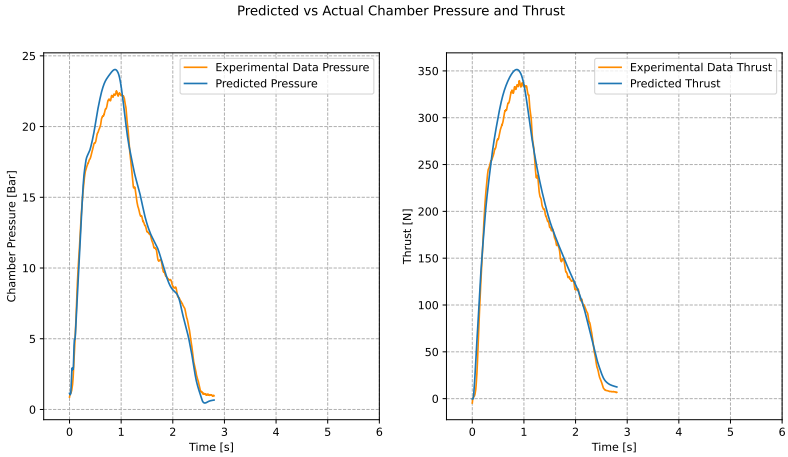
Continued on next page

Table 20.2 – continued from previous page

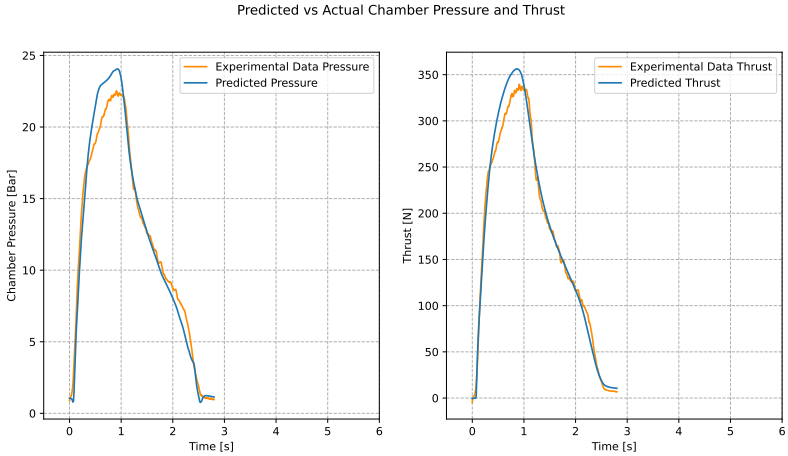
Metric	PINN	I-PINN	Data-Driven	Numerical	I-Numerical
Avg. RMSE Thrust (using 50 data points per file, excluding 3 groups of number of grains) [N]	127.201966	146.868895	175.953411	131.211803	131.211806
Avg. Peak Error Thrust (using 50 data points per file, excluding 3 groups of number of grains) [N]	91.241433	92.963967	86.061666	93.487317	93.487328
Avg. Max Error Thrust (using 50 data points per file, excluding 3 groups of number of grains) [N]	225.852954	259.830174	299.458561	324.990327	324.990339

20.3.9. Visual Comparison Example

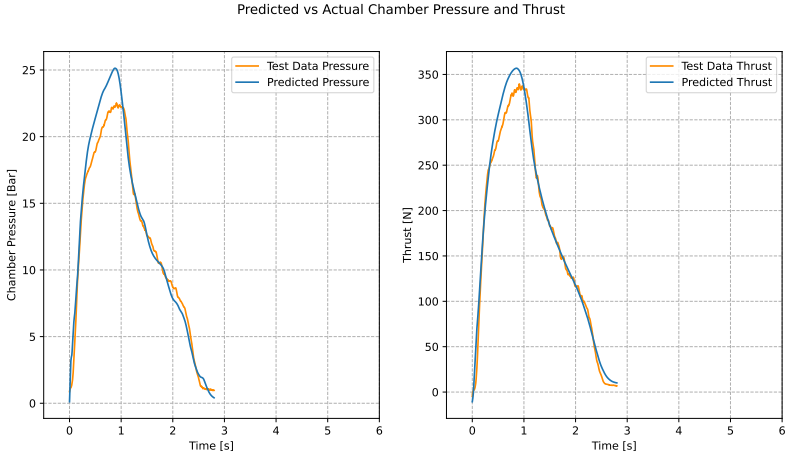
After the tests for the LOO-CV process were completed, and discussed in section 20.2 and the tests to determine the surrogate modelling capability as discussed in section 20.3, it was concluded that the components involving a data-driven component seem to provide more accurate results compared to Numerical methods in terms of average RMSE, peak error and maximum error for chamber pressure and generated thrust predictions. However, it was concluded that the PINN and I-PINN methods are currently not suitable as surrogate modelling methods. While the average RMSE, peak error and maximum error are a good starting position to assess the performance of the various modelling methods, a visual comparison is also required to verify the previous findings and see if the model predictions are indeed able to provide sufficiently accurate results. In Figure 20.51, the predictions of the various modelling methods will be presented against a test data set, which has not been seen during the training process.



(a) PINN trained on 900 data points with Engine3_test7_fullcoating.csv as test file. The RMSE of the chamber pressure prediction is 0.947711 [Bar], the RMSE of the generated thrust prediction is 13.166321 [N], the maximum error of the chamber pressure prediction is 2.157026 [Bar], the maximum error of the generated thrust prediction is 35.418072 [N], the peak error of the chamber pressure prediction is 1.518626 [Bar], the peak error of the generated thrust prediction is 11.924167 [N] and the training time is 671.666092 [s]

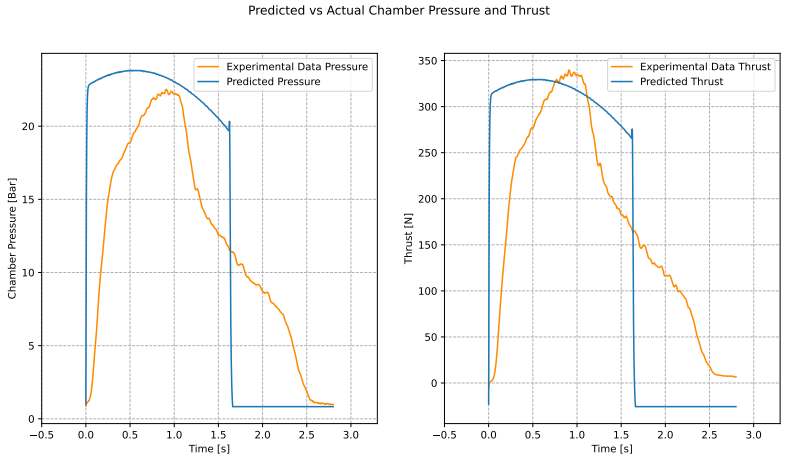


(b) I-PINN trained on 900 data points with Engine3_test7_fullcoating.csv as test file. The RMSE of the chamber pressure prediction is 1.209563 [Bar], the RMSE of the generated thrust prediction is 13.847637 [N], the maximum error of the chamber pressure prediction is 2.910129 [Bar], the maximum error of the generated thrust prediction is 32.809627 [N], the peak error of the chamber pressure prediction is 1.548873 [Bar], the peak error of the generated thrust prediction is 16.708133 [N] and the training time is 712.490046 [s]

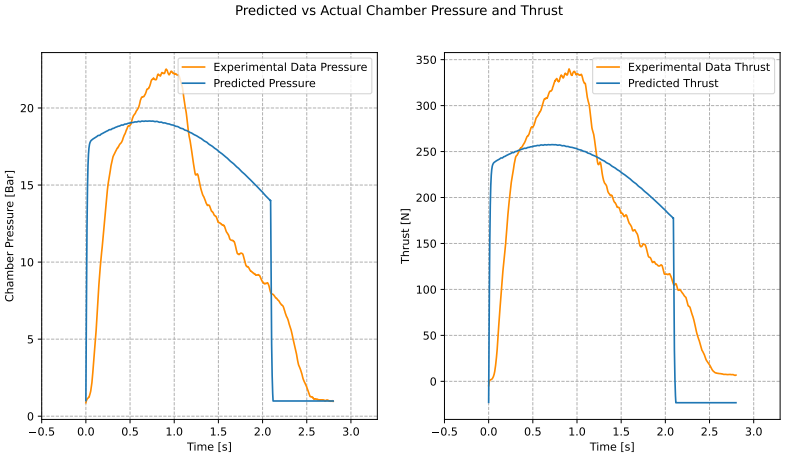


(c) Data-driven method trained on 900 data points with Engine3_test7_fullcoating.csv as test file. The RMSE of the chamber pressure prediction is 1.380030 [Bar], the RMSE of the generated thrust prediction is 14.939422 [N], the maximum error of the chamber pressure prediction is 2.947139 [Bar], the maximum error of the generated thrust prediction is 43.419973 [N], the peak error of the chamber pressure prediction is 2.619578 [Bar], the peak error of the generated thrust prediction is 17.310672 [N] and the training time is 33.457455 [s]

Figure 20.51: Comparison between a trained PINN, I-PINN, Data-driven, Numerical and I-Numerical method against a test file (Part 1/2)



(d) Numerical method with Engine3_test7_fullcoating.csv as test file. The RMSE of the chamber pressure prediction is 7.345650 [Bar], the RMSE of the generated thrust prediction is 111.630186 [N], the maximum error of the chamber pressure prediction is 21.568786 [Bar], the maximum error of the generated thrust prediction is 311.788806 [N], the peak error of the chamber pressure prediction is 1.313439 [Bar] and the peak error of the generated thrust prediction is 9.812800 [N]



(e) I-Numerical method with Engine3_test7_fullcoating.csv as test file. The RMSE of the chamber pressure prediction is 4.921434 [Bar], the RMSE of the generated thrust prediction is 74.950850 [N], the maximum error of the chamber pressure prediction is 16.328446 [Bar], the maximum error of the generated thrust prediction is 232.299457 [N], the peak error of the chamber pressure prediction is 3.355919 [Bar] and the peak error of the generated thrust prediction is 81.978360 [N]

Figure 20.51: Comparison between a trained PINN, I-PINN, Data-driven, Numerical and I-Numerical method against a test file(Part 2/2)

From Figure 20.51, it can be seen that, while the Numerical methods as presented in Figure 20.51d and Figure 20.51e are not able to capture the general trend of the test data set, the methods involving a data-driven component, as presented in Figure 20.51c for the purely Data-Driven method, Figure 20.51a for the PINN and Figure 20.51b for the I-PINN, can predict the shape of the test data set accurately. However, the PINN, I-PINN, and purely data-driven method slightly over-predict the peak value of chamber pressure and generated thrust. Furthermore, it can be seen that the PINN, as presented in Figure 20.51a, and the I-PINN, as presented in Figure 20.51b, have a slightly more accurate prediction in terms of peak error compared to the purely Data-Driven method, as presented in Figure 20.51c. It should be noted that the I-Numerical method, as presented in Figure 20.51e, has a much smaller maximum value compared to the Numerical model, as presented in Figure 20.51d. The most likely reason for this is that the numerical values for the burning rate coefficient and burning rate exponent are smaller compared to the values presented in the literature. The reason for this is that the I-PINN, as presented in Figure 20.51b, has identified during the training process that the experiments which have been used needed to have smaller numerical values for a more accurate fit compared to

the baseline values.

While Figure 20.51a shows promising results for PINN modelling of SRM performance predictions, it should be noted that when changing test files, the accuracy of the various modelling methods involving a data-driven component decreases compared to their experimental counterpart. In Figure 20.52, this is shown for the PINN modelling method.

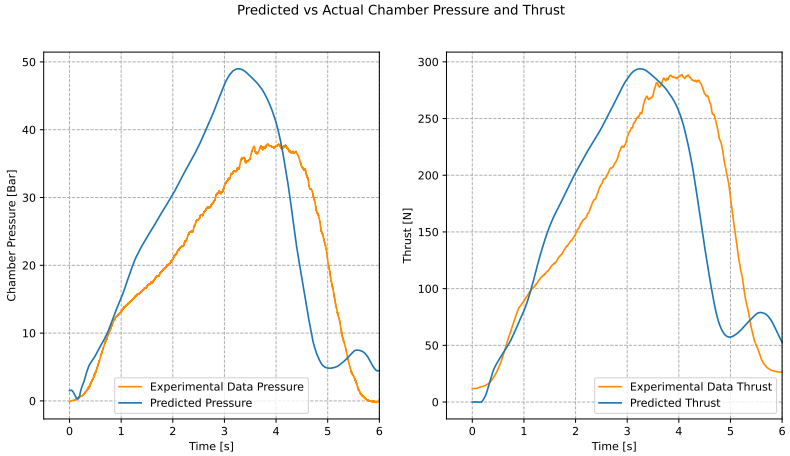


Figure 20.52: PINN trained on 900 data points with ReferenceMotor2_211222_092347.csv as test file. The RMSE of the chamber pressure prediction is 10.068910 [Bar], the RMSE of the generated thrust prediction is 62.057593 [N], the maximum error of the chamber pressure prediction is 21.651085 [Bar], the maximum error of the generated thrust prediction is 167.514054 [N], the peak error of the chamber pressure prediction is 11.134276 [Bar], the peak error of the generated thrust prediction is 5.153868 [N] and the training time is 602.563807 [s]

The most likely reason for this deviation in performance lies in the fact that, when comparing the experimental data from chapter 17, it can be seen that more data files are present for certain training conditions compared to others. Most likely, the NN architecture is therefore able to capture the trend for these data files, but not for other under-represented data files. Therefore, it can be concluded that while the PINN and I-PINN do show some increase in performance when making predictions for certain data files, this method is currently not sufficient to model all types of tests, meaning that the model has a poor generalisation property. This most likely is due to the NN architecture and training process, which has not been optimised during this research. However, currently, this modelling method can be interesting to obtain certain parameters from experimental test data with only a few data points, as will be presented in Figure 20.53.

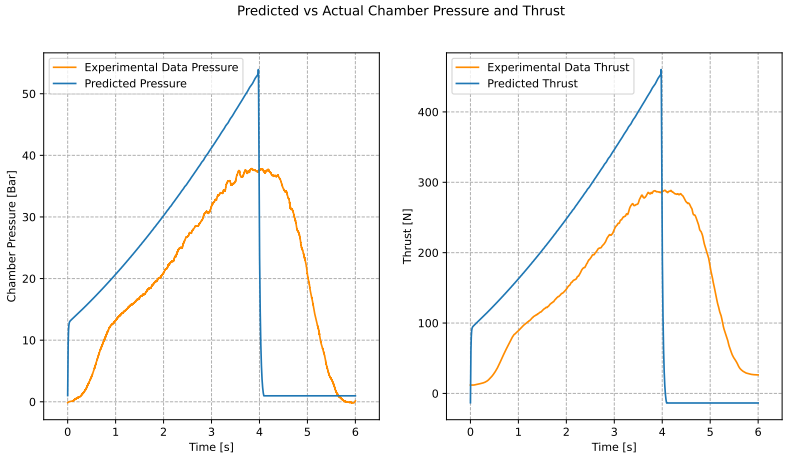


Figure 20.53: Numerical method with ReferenceMotor2_211222_092347.csv as test file. The RMSE of the chamber pressure prediction is 15.754673 [Bar], the RMSE of the generated thrust prediction is 144.788857 [N], the maximum error of the chamber pressure prediction is 36.740806 [Bar], the maximum error of the generated thrust prediction is 302.096926 [N], the peak error of the chamber pressure prediction is 16.076009 [Bar] and the peak error of the generated thrust prediction is 171.487457 [N]

When using a standard Numerical model, as presented in Figure 20.53, it can be seen that the peak error is too high, and the shape and starting position are not correctly modelled. Furthermore, it can be seen that the startup transient has not been modelled correctly. Now, making use of an I-PINN, the burning rate coefficient and burning rate exponent can be treated as learnable parameters, such that these quantities can be extracted for the numerical model. Making use of the results from the I-PINN, an I-Numerical model can be generated, which is presented in Figure 20.54. The additional network parameters, also called learnable parameters, and their numerical values used to model the Numerical model in Figure 20.53 and the I-Numerical model in Figure 20.54 are summarised in Table 20.3.

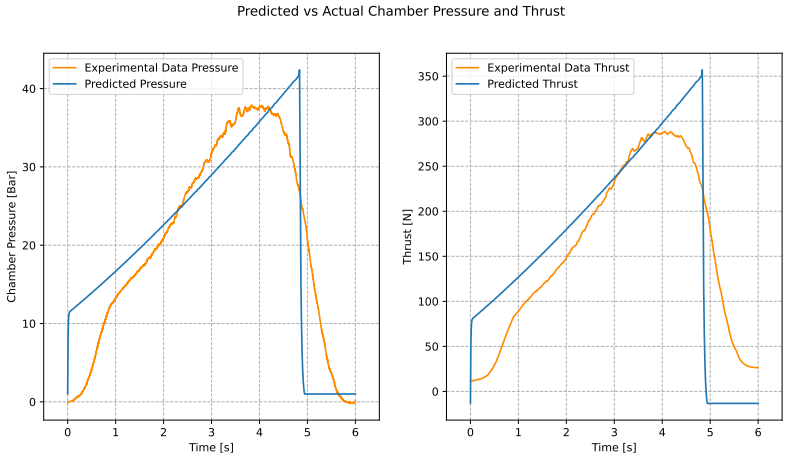


Figure 20.54: Numerical method with ReferenceMotor2_211222_092347.csv as test file. The RMSE of the chamber pressure prediction is 6.590843 [Bar], the RMSE of the generated thrust prediction is 62.098164 [N], the maximum error of the chamber pressure prediction is 22.377792 [Bar], the maximum error of the generated thrust prediction is 211.759303 [N], the peak error of the chamber pressure prediction is 4.528954 [Bar] and the peak error of the generated thrust prediction is 68.176417 [N]

Table 20.3: Comparison of learnable parameters between standard values and I-PINN method

Quantity	Standard Value	I-PINN Value
Burning rate coefficient (a) [mm/s - Pa^n]	5.13	4.7037086
Burning rate constant (n) [-]	0.222	0.14675261

From Figure 20.54 it can be seen that the Numerical model has now a higher accuracy when predicting the maximum values compared to the previous numerical model as presented in Figure 20.53. This means that, while currently not proven to be a fully functional and useful modelling and surrogate modelling method for SRM performance predictions, PINNs and especially I-PINNS could still be a useful tool to enhance Numerical models by obtaining values from limited experimental data. The I-PINN made use of only 5 data points, evenly distributed between 1 and 4 [s], showing the data efficiency of the model. The NN architecture is the same as presented in chapter 19. The data points used during the training process are presented in Figure 20.55.

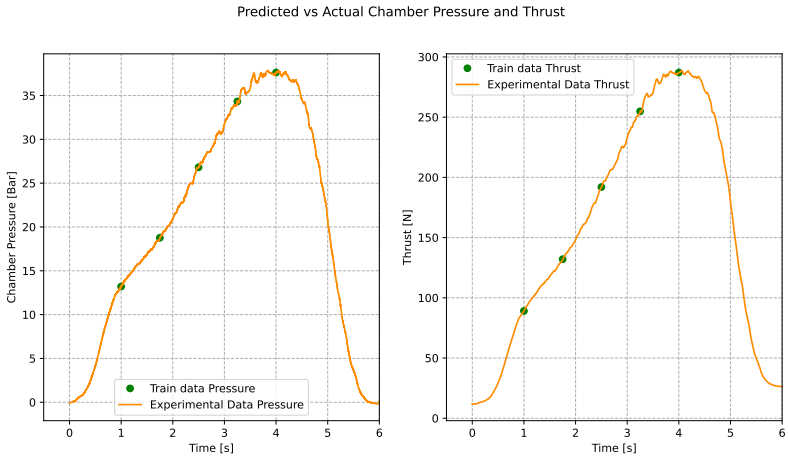


Figure 20.55: Data points used for the generation of the I-PINN using 5 data points, evenly distributed over a domain size between 1 and 4 [s], making use of the ReferenceMotor2_211222_092347.csv experimental data file

Lastly, it should be noted that during this part of the report, a relatively simple underlying physical model has been used to model the performance of SRMs. In this simplified model, as discussed in chapter 18, a lot of assumptions have been made, excluding certain phenomena from being captured in this model. However, certain phenomena could be captured using more extensive models. When doing so, the performance of the PINN and I-PINN would likely increase as well. Therefore, it should be noted that a PINN or I-PINN are not meant as a replacement for numerical models, but rather an addition to have data-driven models embed known physical models rather than mathematical relations obtained from the provided training data, without physical meaning.

21

Conclusion

In this thesis, the performance of PINNs in terms of accuracy, data and computational efficiency against numerical and purely data-driven methods was investigated, as well as their ability to act as a surrogate modelling method. To do so, a software tool was developed using the Python coding language and the PyTorch framework, enabling the generation of customised FNN architectures, loss functions and additional network parameters. The performance of the developed tool was investigated by using three modelling topics related to aerospace, examining the various capabilities of PINNs. For each topic, sub-research questions were formulated, and tests were conducted to address them, thereby contributing to the answer of the main research question. First, the sub-research questions for the first topic, the cooling down of objects use case, will be examined and discussed.

Cooling Down of Objects Modelling

1. ***To what extent can a hybrid model, using a Physics-Informed Neural Network, improve the accuracy and computational efficiency of an object's surface temperature predictions compared to a Numerical method?***

   For this use case, it was observed that a PINN model, developed to predict an object's surface temperature, can improve the accuracy of surface temperature predictions compared to a Numerical method. While the Numerical method consistently has an average RMSE between 8.5-8.75 [°C] and an average maximum error between 10.9-11.3 [°C], the PINN shows an increase in accuracy compared to the Numerical method. The average RMSE of the PINN predictions has found to be between 0.75-0.83 [°C] and an average maximum error between 1.6-2.5 [°C], depending on the training methodology used (e.g. the files and number of data points). However, the PINN has a decreased computational efficiency compared to the Numerical model. The Numerical method has an execution time ranging between 0.0004 to 0.0009 [s]. Besides the fact that the PINN has a training time ranging between 270 to 366 [s], once being trained, the execution time of the PINN ranges between 0.0013 to 0.0016 [s], meaning that the PINN has only a slight decrease in computational efficiency compared to the Numerical method. However, this decrease in computational efficiency does not outweigh the benefit of the increased accuracy when making use of the PINN model method. Concluding, the PINN has an increased accuracy level while having a decreased computational efficiency compared to a Numerical method.

2. ***To what extent can a hybrid model, using a Physics-Informed Neural Network, improve the accuracy and data efficiency of an object's surface temperature predictions compared to a purely Data-Driven method?***

   For this use case, it was observed that a PINN model can achieve a data reduction of 2 to 3 times compared to a purely Data-Driven method for similar performance in terms of

average maximum error. Although the average RMSE has a similar accuracy level for the PINN and purely Data-Driven methods, the PINN has a higher accuracy level in average maximum error given similar conditions. Only when the number of data points reaches a certain threshold, the average RMSE and average maximum errors are of a similar order of magnitude. However, although the accuracy and data efficiency of the PINN are higher compared to the purely Data-Driven method, the training time for the PINN is approximately a factor of 5-6 times higher compared to the purely Data-Driven method. Execution time, on the other hand, has a similar order of magnitude, which is expected when making use of the same NN architecture. Concluding, the PINN has a similar order of accuracy but a higher data efficiency compared to a purely Data-Driven method. However, the purely Data-Driven method is superior in terms of computational efficiency.

3. ***To what extent does treating the convective heat transfer coefficient as a learnable parameter in a Physics-Informed Neural Network affect the accuracy, data efficiency, and computational efficiency of an object's surface temperature predictions, compared to a Numerical and purely Data-Driven method?***

   For this use case, it was observed that by making use of an I-PINN, the data efficiency slightly increases compared to the PINN, while having a similar accuracy level. However, when increasing the number of data points, the accuracy of the I-PINN decreases compared to the PINN and purely Data-Driven method, especially when examining the average maximum error as a performance metric. This showcases that for this situation, increasing the number of data points does not, by definition, help in improving accuracy for the I-PINN when making use of the convective heat coefficient as an additional parameter of the network. Furthermore, the I-PINN has a training time of 1 to 5 [s] larger compared to the PINN, while having a similar execution time. Concluding, the I-PINN does show a slight improvement in data efficiency and accuracy compared to a Numerical and purely Data-Driven method. However, when increasing the number of data points, the I-PINN has a slightly worse performance compared to the PINN. The I-PINN does not improve computational efficiency compared to a Numerical and purely Data-Driven method for similar conditions. Furthermore, the I-PINN show a slight increase in data efficiency compared to the PINN.

4. ***To what extent can a Physics-Informed Neural Network accurately predict an object's surface temperature when presented with new design parameters (for example, object's geometry or ambient conditions) not seen during training, compared to a Numerical and purely Data-Driven method, when validated against experimental data? What are potential limitations?***

   For this use case, it was observed that a PINN can be used as a surrogate modelling method, even when most of the data groups are not being used during the training phase, obtaining similar performance compared to a standard LOO-CV procedure. Although the purely Data-Driven method has a similar performance in terms of average RMSE, the average maximum error for the PINN is lower, given similar conditions. It is found that especially, the I-PINN excels in terms of accuracy when making use of fewer data groups during the training phase compared to the other modelling methods. However, when certain features are not present during the training phase and not provided as additional loss to the PINN's cost function in terms of a physical model or boundary loss, the accuracy reduces for all methods except the Numerical methods. This means that, to create a surrogate model, the problem should be well defined, or the training data should contain the missing problem information. Concluding, a PINN and specifically an I-PINN can act as a surrogate modelling method, obtaining higher accuracy levels compared to a baseline LOO-CV method, when compared against a purely Data-Driven method and a Numerical method. However, the physical model should be well defined, meaning that all the information of the physical problem should be present in either the training data or additional loss functions.

From the first topic, the use case cooling down of objects, promising results were obtained, which showed an increase in accuracy and data-efficiency for a PINN compared to a Numerical and purely

Data-Driven method. Furthermore, it was shown that a PINN and, specifically, an I-PINN, can act as a surrogate modelling method, having only a slight decrease in performance when leaving out certain groups of training data, compared to a baseline LOO-CV. However, it was also shown that the model should be fully described or missing information from the model should be present in the training data, to be able to generate a surrogate model. The second use case investigates a different aspect of a PINN, namely the ability to obtain derivatives analytically rather than numerically. To examine this ability, a PINN was developed for CFD simulations, of which the next sub-research questions are examined and discussed.

Computational Fluid Dynamics Modelling

1. ***To what extent can Physics-Informed Neural Networks serve as a general framework for solving the governing equations in Computational Fluid Dynamics without relying on data using shallow neural networks, compared to a traditional Numerical method?***

   For this use case, it was found that a PINN can provide comparable results to a traditional Numerical method when solving the underlying physical model for low Mach and Reynolds number flows. The PINN has a difference of maximum and minimum values within 10% compared to a traditional Numerical model, making use of a shallower network configuration compared to literature. However, the position and size of the PINN's predictions for pressure and velocity components slightly differ for the various predicted quantities, where the PINN is unable to capture the rapidly changing boundary regions compared to the traditional Numerical method. Besides this, it was found that the computational domain size of the PINN should be smaller compared to the traditional Numerical model, and the inlet and outlet boundary conditions for the PINN need to be different as well to obtain satisfactory results. To conclude, a PINN can be used as a framework to solve the underlying physical model for Computational Fluid Dynamics simulation, but does not exceed a traditional Numerical model for simplified problems using a shallow NN architecture in terms of accuracy and computational efficiency.

2. ***To what extent can a hybrid model, using a Physics-Informed Neural Network without relying on data, improve the accuracy of a Computational Fluid Dynamics simulation compared to a traditional Numerical method when validated against experimental data for the section lift coefficient?***

   For this use case, it was found that while the PINN showed promising results for low Mach and Reynolds number flows, high Mach and Reynolds number flows can not be solved accurately, making use of the same shallow NN architecture. This resulted in insufficiently accurate predictions of the section lift coefficient of the airfoil, while the maximum difference between the experimental data and the traditional Numerical method was found to be within 3%. Therefore, it can be concluded that a PINN is not able to improve accuracy compared to a traditional Numerical method when trained without relying on experimental data using a shallow NN.

3. ***To what extent can a hybrid model using a Physics-Informed Neural Network, informed on experimental data of the pressure coefficient, improve the accuracy of a Computational Fluid Dynamics simulation compared to a traditional Numerical method when validated against experimental data for the section lift coefficient?***

   For this use case, it was found that while introducing experimental data does show some slight improvements in prediction capability, the PINN is not able to provide sufficiently accurate predictions of the section lift coefficient compared to a traditional Numerical method. However, the ability of a PINN to include experimental data at specified locations within the simulation or obtain certain parameters from the flow using an I-PINN opens up possibilities that are currently not feasible using a traditional numerical method. To conclude, a PINN introducing experimental data in the form of pressure values at specified locations does not provide sufficiently accurate results to improve accuracy when validated against experimental data for the section lift coefficient using a shallow NN architecture.

While the first use case showed promising results, the second use case showed that a PINN also has limitations, and additional research is required to obtain better prediction capabilities for CFD simulations. The third and last use case, which will be investigated before the main research question can be answered, is related to the modelling of Solid Rocket Motor performance. This use case is an extension of the first use case, making use of more inputs and outputs and a more extensive physical model. The sub-research questions for the SRM use case will be examined and discussed in the part below.

Solid Rocket Motor Modelling

1. ***To what extent can a hybrid model, using a Physics-Informed Neural Network, improve the accuracy and computational efficiency of chamber pressure and generated thrust predictions in Solid Rocket Motor modelling compared to a Numerical method?***

   For this use case, it was observed that a PINN has a higher accuracy for chamber pressure and generated thrust predictions, compared to a Numerical model, having a decrease in average RMSE of 2-4 [Bar] for chamber pressure and 40-65 [N] for generated thrust, depending on the amount of data points used during the training phase. In terms of average peak error, while the chamber pressure error increases with 1-6 [Bar] when using a PINN, it decreases with 20-28 [N], depending on the number of data points used, compared to the Numerical methods. When examining the average maximum error, a PINN increases accuracy by 5-13 [Bar] for chamber pressure and 100-200 [N] for generated thrust compared to the Numerical methods, depending on the number of data points used. From these observations, it can be concluded that a PINN has an improved accuracy compared to a Numerical method for chamber pressure and generated thrust predictions. However, while the execution time of a Numerical method ranges between 0.006-0.0095 [s], the execution time of the PINN ranges between 0.11-0.15 [s], meaning that the PINN has a decrease in computational efficiency. To conclude, a PINN does have an increased accuracy for chamber pressure and generated thrust predictions, while having a decrease in computational efficiency.

2. ***To what extent can a hybrid model, using a Physics-Informed Neural Network, improve the accuracy and data efficiency of chamber pressure and generated thrust predictions in Solid Rocket Motor modelling compared to a purely Data-Driven method?***

   For this use case, it was observed that a PINN has a similar accuracy level compared to a purely Data-Driven method, having an accuracy increase of about 1-4 [bar] for chamber pressure and 5-20 [N] for generated thrust predictions for average RMSE, average peak error and average maximum error. Furthermore, an increase in data efficiency of 50% has been observed for the PINN compared to the purely data-driven method. However, while the PINN has a training time ranging between 625 to 677 [s], the training time of the purely Data-Driven method ranges from 6-35 [s], meaning that the PINN has a decrease in computational efficiency of 20 to 125 times compared to a purely Data-Driven method. To conclude, while the PINN has a similar accuracy level compared to a purely Data-Driven method and a higher data efficiency, the PINN has a decreased computational efficiency.

3. ***To what extent does treating solid propellant regression rate characteristics (a and n in $\dot{r} = a \cdot p^n$) as learnable parameters in a Physics-Informed Neural Network affect the accuracy, data efficiency, and computational efficiency of chamber pressure and generated thrust predictions in Solid Rocket Motor modelling, compared to a Numerical and purely Data-Driven method?***

   For this use case, it was found that when making use of an I-PINN as a modelling method, the accuracy of chamber pressure and generated thrust predictions is similar for all methods involving a data-driven component in terms of average RMSE and average maximum error. Compared to a Numerical method, the I-PINN shows a similar improvement when compared against the other methods involving a data-driven component. However, the I-PINN has a higher data efficiency compared to the purely Data-Driven method and a slightly higher data efficiency compared to the PINN. In terms of computational efficiency, the I-PINN has

a training time which is only 2-5 [s] higher compared to a PINN, meaning that the training time of the I-PINN is also 20-125 times larger compared to a purely Data-Driven method. In terms of execution time, although using the same NN architecture as the other methods involving a data-driven component, the I-PINN shows a slightly lower execution time, in the range of 0.009-0.11 [s]. To conclude, the I-PINN has a similar order of accuracy compared to the methods involving a data-driven component. Furthermore, the I-PINN has an increased data-efficiency compared to a purely Data-Driven method, to a similar order of magnitude compared to the PINN. However, the I-PINN has a decrease in computational efficiency compared to a Numerical method and an increased training time of 20-125 [s] compared to a purely Data-Driven method, meaning that the I-PINN has a worse computational efficiency.

4. ***To what extent can a Physics-Informed Neural Network accurately predict chamber pressure and generated thrust in a Solid Rocket Motor when presented with new design parameters (for example, grain and nozzle geometries) not seen during training, compared to Numerical and purely Data-Driven methods, when validated against experimental data? What are potential limitations?***

   For this use case, it was found that the PINN method is not able to act as a surrogate modelling method for the prediction of SRM performance. This conclusion is supported by the fact that, compared to a LOO-CV baseline case, excluding data groups from the training process does not obtain a similar level of accuracy. Comparing the PINN to the Numerical and purely Data-Driven method, while the PINN sometimes has a better accuracy compared to these two methods, overall the PINN has a similar level or even decreased level of accuracy compared to the Numerical and purely Data-Driven method. The most likely reason for this observation is the fact that the PINN is unable to provide sufficient generalisation capabilities.

After all the sub-research questions have been answered for the various use cases, the main research question can be answered. The main research question will be examined and discussed in the part below.

For aerospace simulation and surrogate modelling applications, to what extent can Physics-Informed Neural Networks enhance computational and data efficiency while maintaining accuracy, compared to traditional numerical methods and purely data-driven models?

After examining various use cases and implementations of a PINN as a modelling method, it can be concluded that while a purely Data-Driven method has a similar order of accuracy increase as a PINN when compared to a Numerical model, a PINN has a higher data efficiency, meaning that less data points are required to obtain a similar accuracy level. A major downside of PINNs is that, while having an increased accuracy and data-efficiency, the computational efficiency decreases compared to a purely Data-Driven method in terms of training time, and execution time when compared against a Numerical model. However, a PINN could be used as a surrogate modelling method, only requiring a few data groups to provide general predictions, in case the relevant information is embedded correctly in the physics loss or present in the training data and an optimised NN. Overall, the ability of PINNs to embed physical models and obtain parameters of these physical models from only a few data points provides an interesting additional modelling method for Aerospace simulation and surrogate modelling purposes.

22

Recommendations

During this thesis, various aspects of PINNs have been examined, making use of three modelling topics related to aerospace. To do so, a dedicated software class was developed in the Python coding language, using the PyTorch framework. During each of the individual use cases, various possible improvements and ways to gain new insights into the subject of using a PINN for simulation and surrogate modelling have been identified. This chapter provides an overview of these improvements in terms of recommendations for future research projects related to the simulation and surrogate modelling capability of PINNs. First, General recommendations will be presented in section 22.1, whereafter the individual recommendations for each use case will be examined in section 22.2 for the cooling down of objects modelling, section 22.3 for computational fluid dynamics modelling and section 22.4 for solid rocket motor Modelling. It should be noted that the order of the recommendations is not connected to the importance level, meaning that later mentioned recommendations could be equally important as earlier mentioned recommendations.

22.1. General Recommendations

- **Optimisation of the NN architecture**
  While in this research, to obtain the most optimal NN architecture in terms of the number of hidden layers, neurons and type of activation function and initialisation method, use was made of an educated trial-and-error process, it is recognised that the performance of the PINN could be improved when these quantities are optimised. Therefore, it is recommended to perform more research related to the optimisation of the NN architecture for the various use cases.
- **Software class optimisation for GPU**
  The developed software class during this thesis performed sufficiently well for this research. However, especially when performing tests for the second use case related to CFD simulations, it was observed that the developed software class needs to be optimised such that it can be used on a GPU instead of a CPU due to the large computational times observed during the training phase. Therefore, it is recommended to optimise the code for training using GPU hardware.
- **Software class optimisation**
  In the previous point, related to GPU optimisation, it was discussed that the developed software class should be optimised for increased performance during the training phase, making use of GPU hardware rather than CPU hardware. However, the software class could also be optimised in terms of coding practice, since this optimisation process has not been performed during this thesis. For future studies using this software class, it is recommended to enhance the efficiency of the coding class by optimising the software class with more efficient coding practices.
- **Dedicated hardware during training phase**
  Currently, the training of the various modelling methods containing a data-driven component is performed by making use of a CPU on a laptop. However, the PyTorch framework could have an increased performance when being optimised and executed on a dedicated GPU during the

training phase. Therefore, it is recommended for future studies to make use of dedicated GPU hardware during the training phase of the PINN, thereby decreasing the computational time required.

- **Additional use cases**
  While in this thesis only 3 use cases have been examined, it is recommended to identify more use cases for PINNs by domain-specific experts. This way, a generalised conclusion related to PINN simulations and surrogate modelling capabilities can be drawn, as well as opening up new areas of research in which a PINN can be used as an additional modelling or design tool.
- **Fine-tune PINN on data while being trained on physical model**
  An interesting concept to investigate would be to investigate the effects of initially train a PINN on the physical model, followed by fine-tuning using experimental data. This process could optimise initial training time and enhance the generalisation capabilities of the PINN model.
- **Perform experiments using data on fixed domain only to test extrapolation capability**
  To exploit the full potential of PINNs, it would be recommended to investigate the performance of a Numerical and purely Data-Driven method outside the training domain. This way, the ability of PINNs to provide useful predictions outside the training domain can be examined.
- **A more sophisticated method for measuring execution time**
  Currently, the method to measure the training and execution time is performed in such a way that the computer obtains the time before and after the training/execution phases of the model. To obtain the training and execution times, these two set times will be subtracted from each other, obtaining the time difference and, therefore, the training/execution time. However, it was found that when using this method, differences in execution or training time are observed under unchanged conditions. While the differences between the measurements are small, the fact that there is a difference makes the part of the study related to computational efficiency hard to justify. Therefore, it is recommended to implement a more versatile method to obtain the training/execution time for each modelling method.

22.2. Cooling Down of Objects Modelling Recommendations

- **Fully describe the underlying physical model**
  While the Numerical model fully described the physics of the problem related to the cooling down of objects use case, it was shown that, when excluding an ambient temperature group from the training process, the performance of the PINN decreased and was no longer sufficient to provide temperature predictions for surrogate modelling operations. Therefore, it is recommended for future studies to implement an additional far field boundary condition in the cost function, which enforces the predicted temperature to become equal to the measured ambient temperature when time reaches infinity, thereby introducing additional underlying information to the PINN. This could potentially improve the PINN model for surrogate modelling when excluding ambient temperature groups.
- **Additional experimental data**
  While 8 experimental data sets were obtained during this thesis for various geometries, ambient temperatures and materials, it is recommended for future studies to obtain more experimental data sets, with a larger variation in material types, geometries and ambient temperatures. This leads to a better assessment of the performance of the PINN model compared to a Numerical and purely Data-Driven method for surrogate modelling capability.

22.3. Computational Fluid Dynamics Modelling Recommendations

- **Extended version of Navier-Stokes equations**
  While in literature it was shown that the currently used simplified version of the Navier-Stokes equations could be sufficient for CFD simulations, it is recommended for future research to include a more extended version of the Navier-Stokes equations, such as the energy equations and a turbulence model, to better describe the underlying physical problem.
- **Collocation density around object**
  In this research, randomly sampled collocation points were used in the computational domain

for the CFD simulation use case, not considering increasing the density of the collocation points closer to the object. For future studies, it is recommended to investigate if a denser layer around an object of interest leads to increased accuracy or if random sampling is sufficient. When there is a denser collocation layer around the object, there is a possibility that, in total, fewer collocation points are sufficient to obtain similar results, potentially decreasing the computational time required to perform a CFD simulation using a PINN.

- **Additional tailored experimental data**
  In this research, it was investigated if and how experimental data can be incorporated in CFD simulations, making use of a PINN and the performance of a PINN compared to a traditional Numerical method for the section lift coefficient. However, only limited experimental data were available, meaning that for this use case, making use of high Mach and Reynolds number flows was mostly dictated by the availability of experimental data. For future research, it is recommended to perform or obtain experimental data for specific use cases which will be investigated, rather than having the use cases dictated by the experimental data.
- **Section lift coefficient determination**
  The section lift coefficient in this thesis has been obtained by making use of an oversimplified assumption, introducing errors up to ±2.6% when examining the section lift coefficient. For future research, it is recommended to obtain the section lift coefficient by making use of a more reliable and accurate method.
- **Influence of domain size on predictions**
  When decreasing the computational domain size of the PINN when performing the flow over a 2D cylinder experiment, it was observed that the prediction accuracy of the various predicted quantities increases compared to a traditional numerical method. Therefore, it is recommended for future studies to investigate if this observation was an incident or a consistent finding, which means that there exists a possibility that using a PINN results in smaller computational domain sizes.

22.4. Solid Rocket Motor Modelling Recommendations

- **Improved physical model**
  Currently, this thesis makes use of a relatively simple physical model to predict the performance of SRMs. However, more sophisticated physical models could improve the accuracy of the various modelling methods. Therefore, it is recommended for future studies to make use of a more extended physical model in the physics loss of the PINN.
- **Implementation of the physical model**
  Currently, the used physical model is implemented differently compared to the first use case and how it has been done in the literature. The physical model makes use of additional numerical integration steps, and does not solely rely on the integration of the PINN, since the regression rate, chamber volume and surface burn area are not direct outputs of the PINN. This means that the benefit of a PINN, which removes the need for numerical integration, is not being fully used. Therefore, it is recommended for future studies to change the amount of outputs of the PINN to include the other quantities which need to be integrated.
- **Single performance characteristic as output**
  For the PINN model developed for the modelling of the performance of an SRM, two performance outputs were used, namely chamber pressure and generated thrust. For future studies, it is advised to investigate whether predicting a single performance parameter, such as chamber pressure, provides more accurate prediction capabilities for simulations and can generalise better, potentially improving the surrogate modelling capability of the PINN.
- **Data processing and filtering**
  The experimental data used for the SRM PINN contained a lot of noise before filtering. Currently, the Savitzky-Golay filtering method is being used. However, while this method is less generic compared to Butterworth filtering, preventing the removal of desired high-frequency components of the signal, there are still some generic parts when making use of this filtering method, such as the window length and polynomial order. While there are general guidelines, there is no correct

method to determine the required settings for the window length and polynomial order. Therefore, it is recommended to investigate other methods to filter the experimental data and remove the introduced noise, making use of an improved version of a cross-correlation filter or other filtering methods, of which tailored AI filters could be an interesting option.

References

[1] *A Gentle Introduction to torch.autograd — PyTorch Tutorials 2.5.0+cu124 documentation*. URL: https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html.

[2] Ira H.. Abbott and A. E. von. Doenhoff. *Theory of wing sections : including a summary of airfoil data*. Dover Publications, 1959. ISBN: 978-1-62198-621-8.

[3] Nor Hafizah Abdullah, Tengku Farah Wahida Ku Chik, and Fairul Azmin Zaraini. "Data Filter Performance for Rocket Thrust Measurement System". In: *AEROTECH V: Progressive Aerospace Research*. Vol. 629. Applied Mechanics and Materials. Trans Tech Publications Ltd, Oct. 2014, pp. 240–245. DOI: 10.4028/www.scientific.net/AMM.629.240.

[4] *Activation Function - an overview | ScienceDirect Topics*. URL: https://www.sciencedirect.com/topics/engineering/activation-function.

[5] *Adam meyer | Arduino + ds18b20-arduino*. URL: http://adam-meyer.com/arduino/ds18b20-arduino.

[6] *Aerodynamics - Wikipedia*. URL: https://en.wikipedia.org/wiki/Aerodynamics.

[7] Saahil Afaq and Smitha Rao. "Significance Of Epochs On Training A Neural Network". In: *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH* Volume 9 (June 2020). ISSN: 2277-8616. URL: www.ijstr.org.

[8] *AI: What are the different Domains/Subsets of Artificial Intelligence? | by Antonello Sale | Medium*. URL: https://medium.com/@a.sale/ai-what-are-the-different-domains-subsets-of-artificial-intelligence-4cfd5477584.

[9] *Airfoil - Wikipedia*. URL: https://en.wikipedia.org/wiki/Airfoil.

[10] John D.. Anderson. *Fundamentals of aerodynamics*. McGraw-Hill, 2001, p. 892. ISBN: 0-07-237335-0.

[11] Andre Bakker, Ahmad Haidari, and Lanre Oshinowo. "Realize greater benefits from CFD". English. In: *Chemical Engineering Progress* (Mar. 2001), pp. 45–53.

[12] Elijah Hao Wei Ang, Guangjian Wang, and Bing Feng Ng. "Physics-Informed Neural Networks for Low Reynolds Number Flows over Cylinder". In: *Energies* 16.12 (June 2023). ISSN: 19961073. DOI: 10.3390/en16124558.

[13] Ltd. Aosong(Guangzhou) Electronics Co. *Temperature and humidity module DHT11 Product Manual*. 2016. URL: https://www.makerhero.com/img/files/download/DHT11-Datasheet.pdf?srsltid=AfmBOoqaiUfIkJw6hTpwFpGN3IuSaAqUtZpsm9pnwEY5i8LK91ujRj3d.

[14] *Arduino Dht11 Sensor Interfacing With Arduino Uno | Arduino*. URL: https://www.electronicwings.com/arduino/dht11-sensor-interfacing-with-arduino-uno.

[15] *Artificial Neuron - an overview | ScienceDirect Topics*. URL: https://www.sciencedirect.com/topics/psychology/artificial-neuron.

[16] *Autograd mechanics — PyTorch 2.5 documentation*. URL: https://pytorch.org/docs/stable/notes/autograd.html.

[17] João Victor Barros dos Santos et al. "Analysis of Parametrization Methods for Aerodynamic Profiles". In: Associacao Brasileira de Engenharia e Ciencias Mecanicas - ABCM, Dec. 2020. DOI: 10.26678/abcm.encit2020.cit20-0659.

[18] Karsten Berns, Alexander Köpper, and Bernd Schürmann. *Lecture Notes in Electrical Engineering 732 Technical Foundations of Embedded Systems Electronics, System theory, Components and Analysis*. Tech. rep. URL: http://www.springer.com/series/7818.

[19] *Biot number - Wikipedia*. URL: https://en.wikipedia.org/wiki/Biot_number.

[20] Wadii Boulila et al. *WEIGHT INITIALIZATION TECHNIQUES FOR DEEP LEARNING ALGORITHMS IN REMOTE SENSING: RECENT TRENDS AND FUTURE PERSPECTIVES*. Tech. rep.

[21] Tyler J. Bradshaw et al. *A Guide to Cross-Validation for Artificial Intelligence in Medical Imaging*. July 2023. DOI: 10.1148/ryai.220232.

[22] C. Brischke and M. Humar. "Performance of the bio-based materials". In: *Performance of Bio-based Building Materials*. Elsevier Inc., July 2017, pp. 249–333. ISBN: 9780081009925. DOI: 10.1016/B978-0-08-100982-6.00005-7.

[23] A Cervone -Ir BTC Zandbergen, by R Klein, and M Fernandez Jimenez. *Rocket Propulsion*. Tech. rep. 2017.

[24] Clickworker. *How to Validate Machine Learning Models*. 2023. URL: https://www.clickworker.com/customer-blog/how-to-validate-machine-learning-models/ (visited on 10/01/2024).

[25] *Comparison of Supervised, Unsupervised, Semi-Supervised and Reinforcement - IFoA Data Science*. URL: https://ifoadatascienceresearch.github.io/tutorial/comparison/.

[26] *Compressible Flow vs Incompressible Flow in Fluid Mechanics*. URL: https://www.simscale.com/docs/simwiki/cfd-computational-fluid-dynamics/compressible-flow-vs-incompressible-flow/.

[27] *Computational fluid dynamics - Wikipedia*. URL: https://en.wikipedia.org/wiki/Computational_fluid_dynamics.

[28] *Computational Fluids Laboratory 3-Dimensional (CFL3D)(LAR-16003-1) | NASA Software Catalog*. 2025. URL: https://software.nasa.gov/software/LAR-16003-1.

[29] Saeed Damadi, Golnaz Moharrer, and Mostafa Cham. "The Backpropagation algorithm for a math student". In: *International Joint Conference on Neural Networks* (Jan. 2023). URL: http://arxiv.org/abs/2301.09977.

[30] *de Laval nozzle - Wikipedia*. URL: https://en.wikipedia.org/wiki/De_Laval_nozzle.

[31] NumPy Developers. *numpy.log1p — NumPy v2.0 Manual*. 2024. URL: https://numpy.org/doc/2.0/reference/generated/numpy.log1p.html (visited on 10/08/2024).

[32] *Directed acyclic graph - Wikipedia*. URL: https://en.wikipedia.org/wiki/Directed_acyclic_graph.

[33] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*. Sept. 2021. URL: http://arxiv.org/abs/2109.14545.

[34] *Dynamics of Flight*. URL: https://www.grc.nasa.gov/www/k-12/UEET/StudentSite/dynamicsofflight.html.

[35] ElectronicTutorials. *Butterworth Filter Design*. 2014. URL: https://www.electronics-tutorials.ws/filter/filter_8.html (visited on 09/20/2024).

[36] Xiao Fei et al. "Digital Twin of Solid Rocket Motor, Problem and Challenge". In: *Proceedings - 2018 11th International Symposium on Computational Intelligence and Design, ISCID 2018*. Vol. 2. Institute of Electrical and Electronics Engineers Inc., July 2018, pp. 7–11. ISBN: 9781538685266. DOI: 10.1109/ISCID.2018.10103.

[37] *Finite difference method - Wikipedia*. URL: https://en.wikipedia.org/wiki/Finite_difference_method.

[38] *Finite element method - Wikipedia*. URL: https://en.wikipedia.org/wiki/Finite_element_method.

[39] *Finite volume method - Wikipedia*. URL: https://en.wikipedia.org/wiki/Finite_volume_method.

[40] *Finite Volume Method (FVM) - Literature Review (ITCFDUML&OF/Challenge/Week-9)*. URL: https://skill-lync.com/student-projects/finite-volume-method-fvm-literature-review-itcfdumlofchallengeweek-9.

[41] Neal B Gallagher. *Savitzky-Golay Smoothing and Differentiation Filter*. Tech. rep. URL: www.Eigenvector.com.

[42] Dhruv Gamdha et al. “Automated Defect Recognition on X-ray Radiographs of Solid Propellant Using Deep Learning Based on Convolutional Neural Networks”. In: *Journal of Nondestructive Evaluation* 40.1 (Mar. 2021). ISSN: 15734862. DOI: 10.1007/s10921-021-00750-4.

[43] ifm electronic gmbh. *PT5402*. 2014. URL: https://docs.rs-online.com/b08b/0900766b813361b1.pdf.

[44] Ian. Goodfellow, Yoshua. Bengio, and Aaron. Courville. *Deep learning*. The MIT Press, 2016, p. 775. ISBN: 9780262337373.

[45] Vinay K. Goyal et al. *Applications of Machine Learning to Launch Vehicles*. Jan. 2024. DOI: 10.2514/6.2024-1209.

[46] Su-Hyun Han et al. “Artificial Neural Network: Understanding the Basic Concepts without Mathematics”. In: *Dementia and Neurocognitive Disorders* 17.3 (2018), p. 83. ISSN: 1738-1495. DOI: 10.12779/dnd.2018.17.3.83.

[47] *Heat transfer coefficient - Wikipedia*. URL: https://en.wikipedia.org/wiki/Heat_transfer_coefficient.

[48] Rob Hermsen and C Margaritis. “SECTION A-A SCALE 2 : 1 A A TRP SRM2020 Nozzle Delft Aerospace Rocket Engineering / TUDelft LR”. In: (). DOI: 10.50{${}^\circ$}`.50{${}^\circ$}`.50{${}^\circ$}.

[49] Stefan Hickel. *This lecture: Physical Modeling Mathematical Modeling Numerical Modeling Solution Interpretation*. 2022.

[50] Stefan Hickel -Cfd and Aerospace Engineers. *This lecture: Physical Modeling Mathematical Modeling Numerical Modeling Solution Interpretation*. Tech. rep.

[51] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (Jan. 1989), pp. 359–366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8.

[52] *How to calculate the accuracy of pressure transmitter and what is the total probable error ? | Fuji Electric*. URL: https://www.fujielectric.fr/en/blog/how-calculate-accuracy-pressure-transmitter-and-what-total-probable-error/.

[53] Beichao Hu and Dwayne McDaniel. “Applying Physics-Informed Neural Networks to Solve Navier–Stokes Equations for Laminar Flow around a Particle”. In: *Mathematical and Computational Applications* 28.5 (Oct. 2023), p. 102. ISSN: 2297-8747. DOI: 10.3390/mca28050102. URL: https://www.mdpi.com/2297-8747/28/5/102.

[54] Dobbin Huang. *Ad Physics-Informed Neural Networks for Fluid Mechanics*. Tech. rep. 2022.

[55] institut für technische verbrennung. *Lecture 2 Adiabatic Flame Temperature and Chemical Equilibrium 2.-1*. Tech. rep.

[56] Mark Kalsbeek and Christos Margaritis. *SRM 2020*. Apr. 24.

[57] Pavan Kalyan Pandranki and Pavan Kalyan Pandranki ————————————————. *Combined Modes of Heat Transfer in Complex Systems: A Comprehensive Review*. Tech. rep. 5. 2023, pp. 2455–6211. URL: https://www.researchgate.net/publication/372789024.

[58] R Klees and R P Dwight. *Applied Numerical Analysis (AE2220-I)*. Tech. rep. 2020.

[59] Nikolaus Kriegeskorte and Tal Golan. “Neural network models and deep learning”. In: *Current Biology* 29.7 (Apr. 2019), R231–R236. ISSN: 0960-9822. DOI: 10.1016/J.CUB.2019.02.034.

[60] Stefan Kurz et al. “Hybrid modeling: towards the next level of scientific computing in engineering”. In: *Journal of Mathematics in Industry* 12.1 (Dec. 2022). ISSN: 21905983. DOI: 10.1186/s13362-022-00123-0.

[61] Charles L Ladson, Aequilla S Hill, and Wuliam G Johnson. *Pressure Distributions from High Reynolds Number Transonic Tests of an NACA 0012 Airfoil in the Langley 0.3-Meter Transonic Cryogenic Tunnel*. Dec. 1987.

[62] *Lift coefficient - Wikipedia*. URL: https://en.wikipedia.org/wiki/Lift_coefficient.

[63] *List of thermal conductivities - Wikipedia*. URL: https://en.wikipedia.org/wiki/List_of_thermal_conductivities.

[64] *Mach Number*. URL: https://www.grc.nasa.gov/www/k-12/airplane/mach.html.

[65] *Mach number - Wikipedia*. URL: https://en.wikipedia.org/wiki/Mach_number.

[66] *Machine learning, explained | MIT Sloan*. URL: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained.

[67] *MAE 5540-Propulsion Systems Modeling Transient Rocket Operation (Lecture 7.2: Solid Rockets)*. Tech. rep. URL: http://members.aol.com/ricnakk/th_pres.html.

[68] Zhiping Mao, Ameya D. Jagtap, and George Em Karniadakis. "Physics-informed neural networks for high-speed flows". In: *Computer Methods in Applied Mechanics and Engineering* 360 (Mar. 2020). ISSN: 00457825. DOI: 10.1016/j.cma.2019.112789.

[69] C Margaritis and Mark Kalsbeek. *SRM 2020 Casing*. Sept. 2023.

[70] Inc Maxim Integrated Products. *DS18B20, Programmable Resolution 1-Wire Digital Thermometer*. 2019. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf.

[71] *Mean Error: Definition - Statistics How To*. URL: https://www.statisticshowto.com/mean-error/.

[72] *Mean Squared Error (MSE) - Statistics By Jim*. URL: https://statisticsbyjim.com/regression/mean-squared-error-mse/.

[73] Mechanical and Aerospace Engineering. *Project2_solution*.

[74] Siddhartha Mishra and Ben Moseley. *Deep Learning in Scientific Computing PINNs-limitations and extensions Spring Semester 2023*. 2023.

[75] R. Mohanasundaram et al. "Deep Learning and Semi-Supervised and Transfer Learning Algorithms for Medical Imaging". In: *Deep Learning and Parallel Computing Environment for Bioengineering Systems* (Jan. 2019), pp. 139–151. DOI: 10.1016/B978-0-12-816718-2.00015-4.

[76] MR.Cole_Photographer. *3D illustration Rendering of binary code pattern Abstract background.Futuristic Particles for business,Science and technology - Stockfoto*. 2000. URL: https://www.gettyimages.nl/detail/foto/illustration-rendering-of-binary-code-pattern-royalty-free-beeld/1199128740 (visited on 08/28/2024).

[77] *NACA airfoil - Wikipedia*. 2025. URL: https://en.wikipedia.org/wiki/NACA_airfoil.

[78] Richard Nakka. *Solid Rocket Motor Theory - Combustion*. 2001. URL: http://www.nakka-rocketry.net/th_comb.html (visited on 09/30/2024).

[79] Richard Nakka. *Solid Rocket Motor Theory - Propellant Grain*. 2001. URL: http://www.nakka-rocketry.net/th_grain.html (visited on 09/30/2024).

[80] Richard Nakka. *Sorbitol Chemistry in Rocket Propellants*. 2001. URL: https://www.nakka-rocketry.net/sorbchem.html (visited on 09/30/2024).

[81] Richard Nakka. *Sorbitol-based Propellant (aka "R-candy")*. 2001. URL: http://www.nakka-rocketry.net/sorb.html#Introduction (visited on 09/30/2024).

[82] NASA. *nasa_systems_engineering_handbook_0*. 2016.

[83] NASA. *Superseding NASA-STD-7009A w/Change 1 STANDARD FOR MODELS AND SIMULATIONS NOT MEASUREMENT SENSITIVE*. May 2024. URL: https://standards.nasa.gov..

[84] NASA Glenn Research Center. *CEARUN: Chemical Equilibrium with Applications*. https://cearun.grc.nasa.gov. 2024. (Visited on 10/24/2024).

[85] NASA/MSFC. *Brilliant Sub-scale Solid Rocket Motor Test*. 2012. URL: https://www.nasa.gov/image-article/brilliant-sub-scale-solid-rocket-motor-test/ (visited on 08/28/2024).

[86] *Navier-Stokes Equations*. URL: https://www.grc.nasa.gov/www/k-12/airplane/nseqs.html.

[87] *Navier–Stokes equations - Wikipedia*. URL: https://en.wikipedia.org/wiki/Navier%E2%80%93Stokes_equations.

[88] *Newton's law of cooling - Wikipedia*. 2025. URL: https://en.wikipedia.org/wiki/Newton%27s_law_of_cooling.

[89] NI. *NI TDMS File Format - What is a TDMS File?* 2024. URL: https://www.ni.com/en/support/documentation/supplemental/06/the-ni-tdms-file-format.html?srsltid=AfmBOoqcKAK_4wkofmMLr19xyWGbrJMs-no1twZiR83Lp4nDb3pEA70G (visited on 09/12/2024).

[90] *Normal distribution - Wikipedia*. URL: https://en.wikipedia.org/wiki/Normal_distribution.

[91] M C Olde. *Potassium Nitrate Sorbitol Propellant Experimental Investigation of Solid Propellant Characteristics*. Tech. rep. URL: http://repository.tudelft.nl/..

[92] *OpenFOAM*. 2025. URL: https://www.openfoam.com/.

[93] Piero Paialunga. *Noise Cancellation with Python and Fourier Transform*. 2021. URL: https://towardsdatascience.com/noise-cancellation-with-python-and-fourier-transform-97303314aa71 (visited on 09/23/2024).

[94] *Paired difference test - Wikipedia*. URL: https://en.wikipedia.org/wiki/Paired_difference_test.

[95] *Partial derivative - Wikipedia*. URL: https://en.wikipedia.org/wiki/Partial_derivative.

[96] *Physics-informed Neural Networks: a simple tutorial with PyTorch | by Theo Wolf | Medium*. URL: https://medium.com/@theo.wolf/physics-informed-neural-networks-a-simple-tutorial-with-pytorch-f28a890b874a.

[97] *Pressure coefficient - Wikipedia*. URL: https://en.wikipedia.org/wiki/Pressure_coefficient.

[98] *PyTorch*. 2025. URL: https://pytorch.org/.

[99] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (Feb. 2019), pp. 686–707. ISSN: 10902716. DOI: 10.1016/j.jcp.2018.10.045.

[100] Alessandro Rovetta. "Raiders of the Lost Correlation: A Guide on Using Pearson and Spearman Coefficients to Detect Hidden Correlations in Medical Sciences". In: *Cureus* (Nov. 2020). DOI: 10.7759/cureus.11794.

[101] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. Dublin, Sept. 2017. URL: http://arxiv.org/abs/1609.04747.

[102] Ideen Sadrehaghighi. *CFD Open Series Patch Mesh Generation in CFD A N N A P O L I S , MD Cyliner Head-(Polyhedral cells) Mixer (SAMM cells) Typical Turbo-Machine Mesh (Hexahedral cells) Wing-Body-Pylon-Nacelle (Tetrahedral cells)*. Tech. rep.

[103] Mark Schwabacher. "Machine learning for rocket propulsion health monitoring". In: *SAE Technical Papers*. SAE International, 2005. DOI: 10.4271/2005-01-3370.

[104] SciPy Community. *SciPy Documentation: scipy.integrate.solve_ivp*. 2024. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html (visited on 10/24/2024).

[105] SciPy Community. *scipy.signal.savgol_filter — SciPy v1.10.0 Manual*. 2023. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.savgol_filter.html (visited on 10/08/2024).

[106] *Shapiro–Wilk test - Wikipedia*. URL: https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test.

[107] Vinod Sharma. "A Study on Data Scaling Methods for Machine Learning". In: *International Journal for Global Academic & Scientific Research* 1.1 (Feb. 2022). DOI: 10.55938/ijgasr.v1i1.4.

[108] Farhad Mortezapour Shiri et al. *A Comprehensive Overview and Comparative Analysis on Deep Learning Models*. Tech. rep.

[109] Gyorgyyj Simon. *Health Informatics Artificial Intelligence and Machine Learning in Health Care and Medical Sciences Best Practices and Pitfalls*. English. 2024. DOI: 10.1007/978-3-031-39355-6. URL: https://link.springer.com/book/10.1007/978-3-031-39355-6.

[110] *Solid-propellant rocket - Wikipedia*. URL: https://en.wikipedia.org/wiki/Solid-propellant_rocket.

[111] *Statistical hypothesis test - Wikipedia*. URL: https://en.wikipedia.org/wiki/Statistical_hypothesis_test.

[112] *Statistical Significance - StatPearls - NCBI Bookshelf*. URL: https://www.ncbi.nlm.nih.gov/books/NBK459346/.

[113] Nathaniel P Stebbins Dahl et al. *Assignment for AE4S01(P), 2022/2023 Thermal Rocket Propulsion Practical*. Tech. rep.

[114] *Stoichiometry - Wikipedia*. URL: https://en.wikipedia.org/wiki/Stoichiometry.

[115] Xue Qin Sun et al. "Regression transients modeling of solid rocket motor burning surfaces with physics-guided neural network". In: *Machine Learning: Science and Technology* 5.1 (Mar. 2024). ISSN: 26322153. DOI: 10.1088/2632-2153/ad2973.

[116] *Supervised Machine Learning: Classification and Regression | by Nimra Shahzadi | Medium*. URL: https://medium.com/@nimrashahzadisa064/supervised-machine-learning-classification-and-regression-c145129225f8.

[117] Aerocoon Systems. *500 and 1000 kG Load Cell*. 2025. URL: https://aeroconsystems.com/product/load-cells/500-and-1000-kg-load-cell.

[118] S. Taamallah et al. *Fuel flexibility, stability and emissions in premixed hydrogen-rich gas turbine combustion: Technology, fundamentals, and numerical simulations*. Sept. 2015. DOI: 10.1016/j.apenergy.2015.04.044.

[119] *Table of specific heat capacities - Wikipedia*. URL: https://en.wikipedia.org/wiki/Table_of_specific_heat_capacities.

[120] Juan Terven et al. "Loss Functions and Metrics in Deep Learning". In: (July 2023). URL: http://arxiv.org/abs/2307.02694.

[121] *The Essential Guide to Neural Network Architectures*. URL: https://www.v7labs.com/blog/neural-network-architectures-guide.

[122] TU Delft. *Ch. 10: nozzle flow*. Tech. rep.

[123] *Turbulence Modeling Resource*. URL: https://turbmodels.larc.nasa.gov/index.html.

[124] *Understanding Autograd in PyTorch: The Core of Automatic Differentiation | by Sahin Ahmed, Data Scientist | Dec, 2024 | Medium*. URL: https://medium.com/@sahin.samia/understanding-autograd-in-pytorch-the-core-of-automatic-differentiation-ca9d98da980d.

[125] Can Unlusoy et al. "Advancing Airfoil Design: A Physics-Inspired Neural Network Model". In: ASME International, June 2024. DOI: 10.1115/gt2024-122682.

[126] Wouter van der Wal. *Challenge the future Simulation, verification*. Tech. rep.

[127] Sifan Wang et al. *An Expert's Guide to Training Physics-informed Neural Networks*. Pennsylvania, Aug. 2023. DOI: 10.48550/arXiv.2308.08468. URL: https://www.researchgate.net/publication/373160693.

[128] Waweru Wangui. *Adversarial Attacks on Neural Networks*. 2023. URL: https://medium.com/@wanguiwawerub/adversarial-attacks-on-neural-networks-240a47c76f4c (visited on 10/01/2024).

[129] Binbin Wei, Yongwei Gao, and Shuling Hu. "Experimental study on transition of dynamic airfoil in pitching oscillation". In: *Advances in Aerodynamics* 5.1 (Dec. 2023). ISSN: 25246992. DOI: 10.1186/s42774-023-00141-5.

[130] Ran Wei et al. "Towards an extensible model-based digital twin framework for space launch vehicles". In: *Journal of Industrial Information Integration* 41 (Sept. 2024). ISSN: 2452414X. DOI: 10.1016/j.jii.2024.100641.

[131] *What Are Navier-Stokes Equations? | SimWiki | SimScale*. URL: https://www.simscale.com/docs/simwiki/numerics-background/what-are-the-navier-stokes-equations/.

[132] *What Is a Neuron? Diagrams, Types, Function, and More*. URL: https://www.healthline.com/health/neurons.

[133] *What Is Artificial Intelligence (AI)? | Google Cloud*. URL: https://cloud.google.com/learn/what-is-artificial-intelligence.

[134] *What is bias-variance decomposition and when is it used?* URL: https://analyticsindiamag.com/ai-mysteries/what-is-bias-variance-decomposition-and-when-is-it-used/.

[135] *What is Reynolds Number (Re)? (Complete Guide) | SimScale*. URL: https://www.simscale.com/docs/simwiki/numerics-background/what-is-the-reynolds-number/.

[136] *Wilcoxon signed-rank test - Wikipedia*. URL: https://en.wikipedia.org/wiki/Wilcoxon_signed-rank_test.

[137] B T C Zandbergen. *ae4-S01 Thermal Rocket Propulsion (version 2.09)*. Tech. rep. 2022.

[138] Wei Zhang et al. *Data-driven deep learning approach for thrust prediction of solid rocket motors*. Feb. 2024. DOI: 10.1016/j.measurement.2023.114051.

A

Experimental data figures of Cooling Down of Objects Experiments

In this appendix, the visualised data of the experiments performed on the cooling down of various objects are presented. Both the filtered and raw data are presented. Each section provides an individual dataset and is named according to the file name as presented in Table 6.5

A.1. CDOB_01.csv

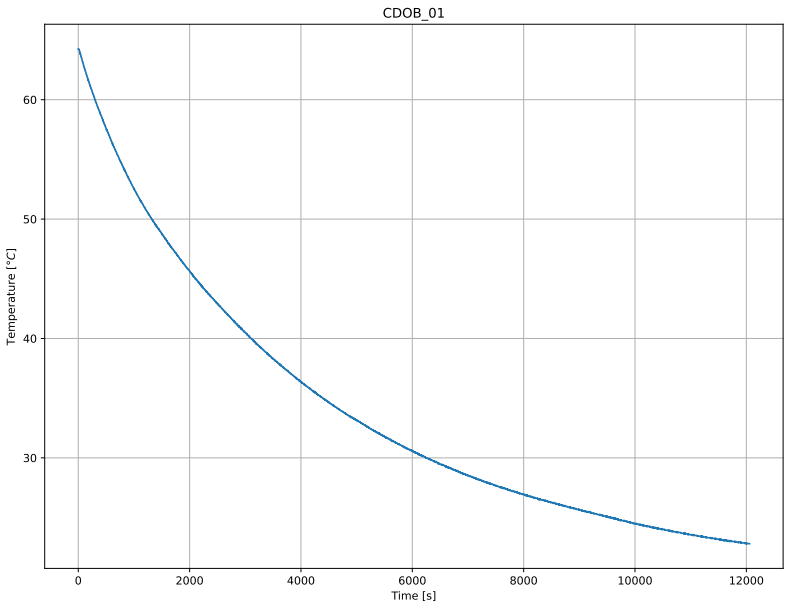


Figure A.1: Raw Object Temperature Data for CDOB-01

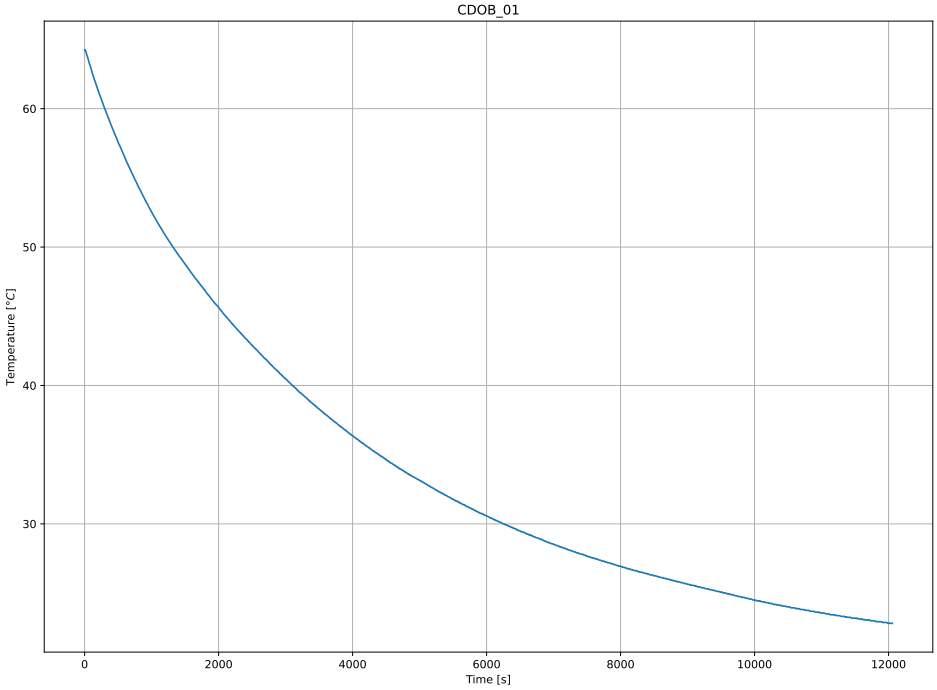


Figure A.2: Filtered Object Temperature Data for CDOB-01

A.2. CDOB_02.csv

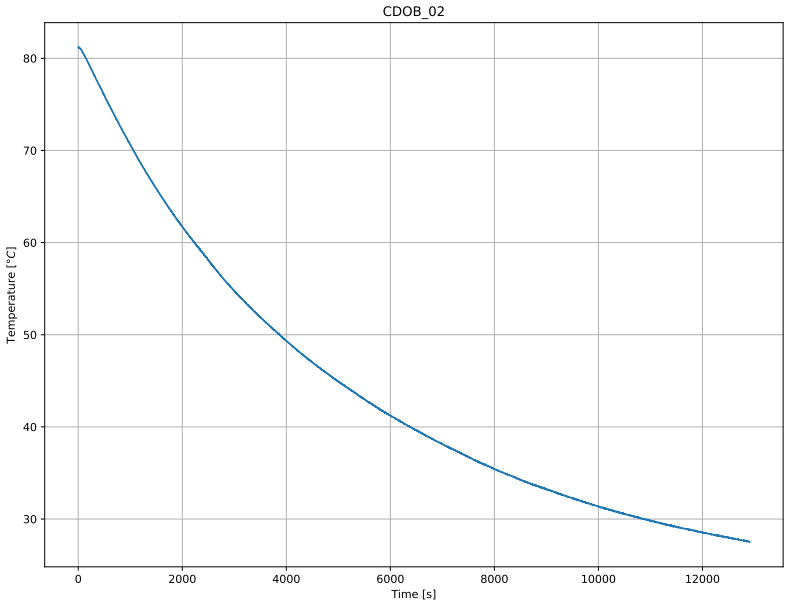


Figure A.3: Raw Object Temperature Data for CDOB-02

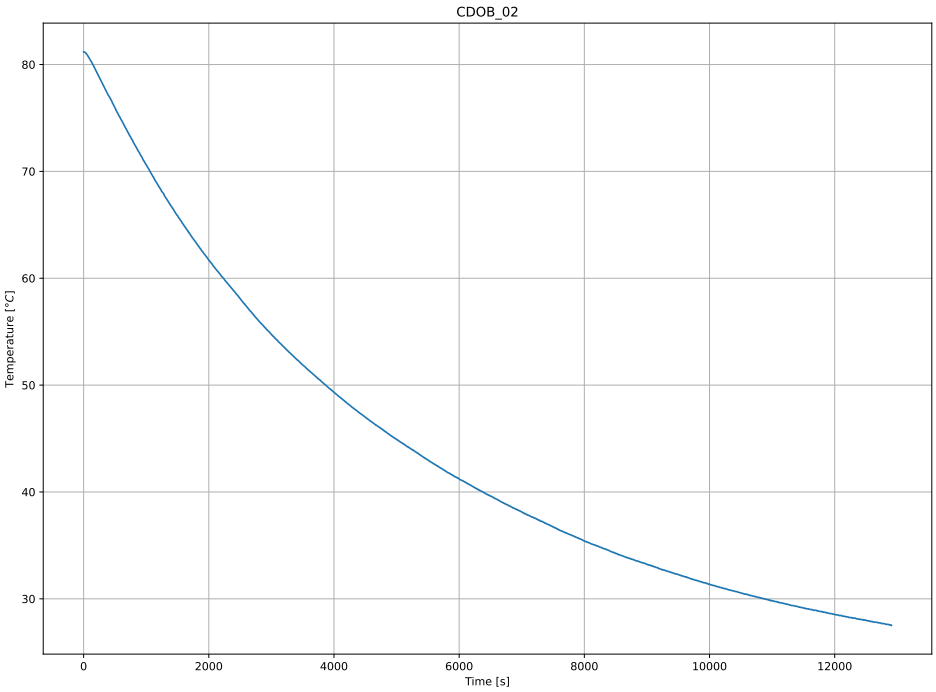


Figure A.4: Filtered Object Temperature Data for CDOB-02

A.3. CDOB_03.csv

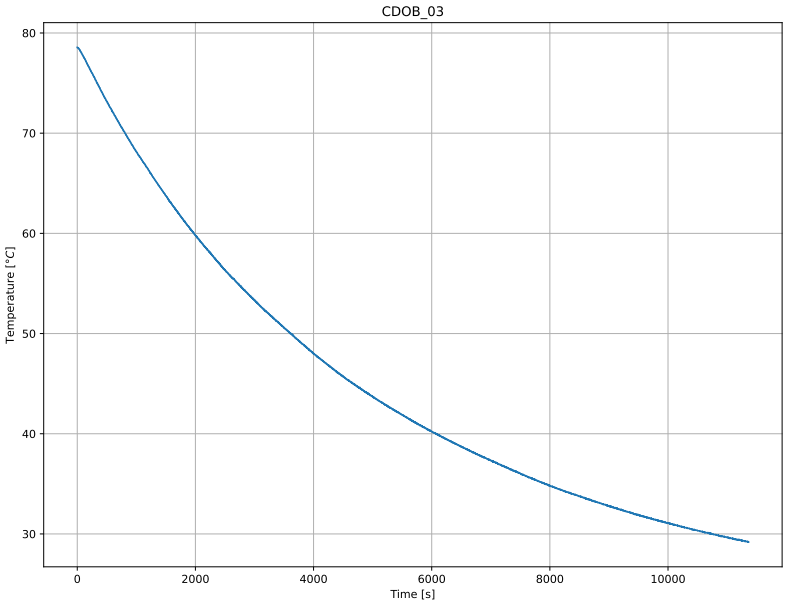


Figure A.5: Raw Object Temperature Data for CDOB-03

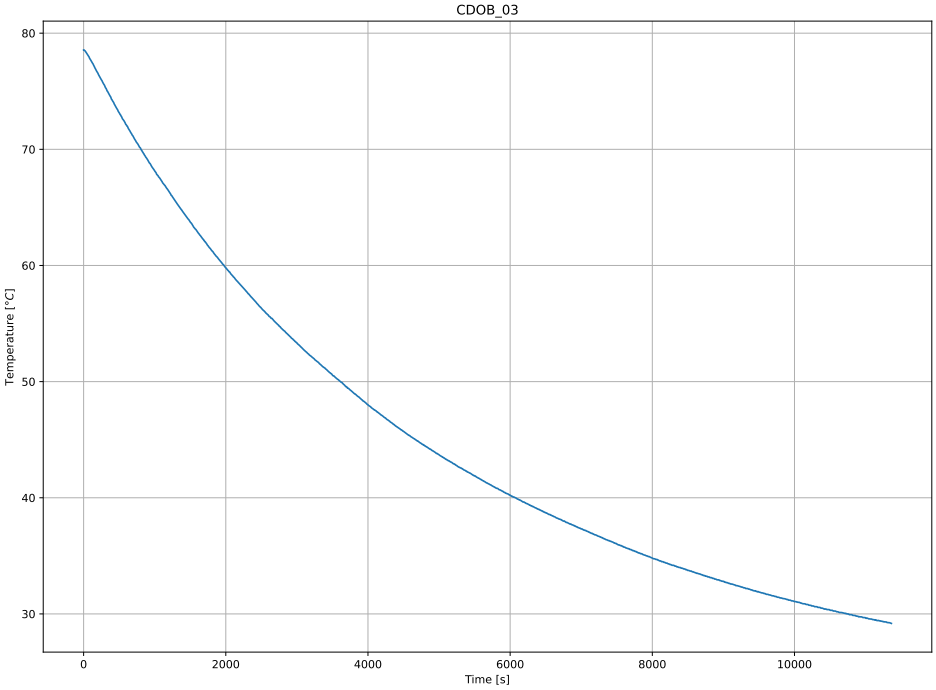


Figure A.6: Filtered Object Temperature Data for CDOB-03

A.4. CDOB_04.csv

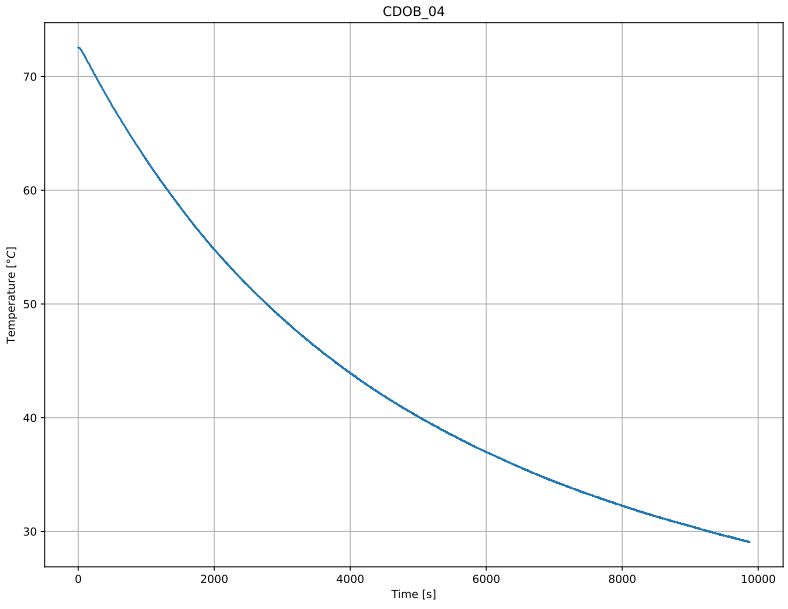


Figure A.7: Raw Object Temperature Data for CDOB-04

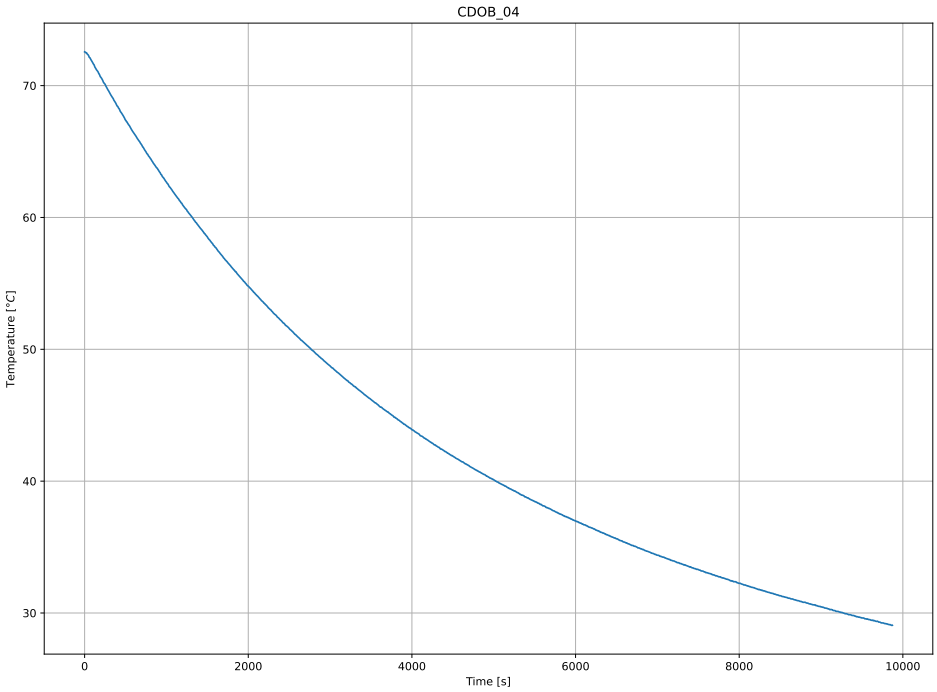


Figure A.8: Filtered Object Temperature Data for CDOB-04

A.5. CDOB_05.csv

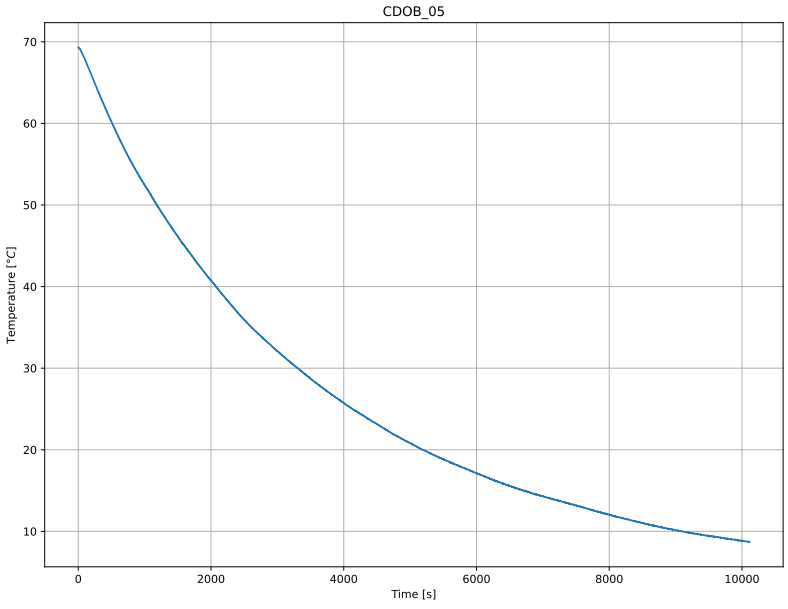


Figure A.9: Raw Object Temperature Data for CDOB-05

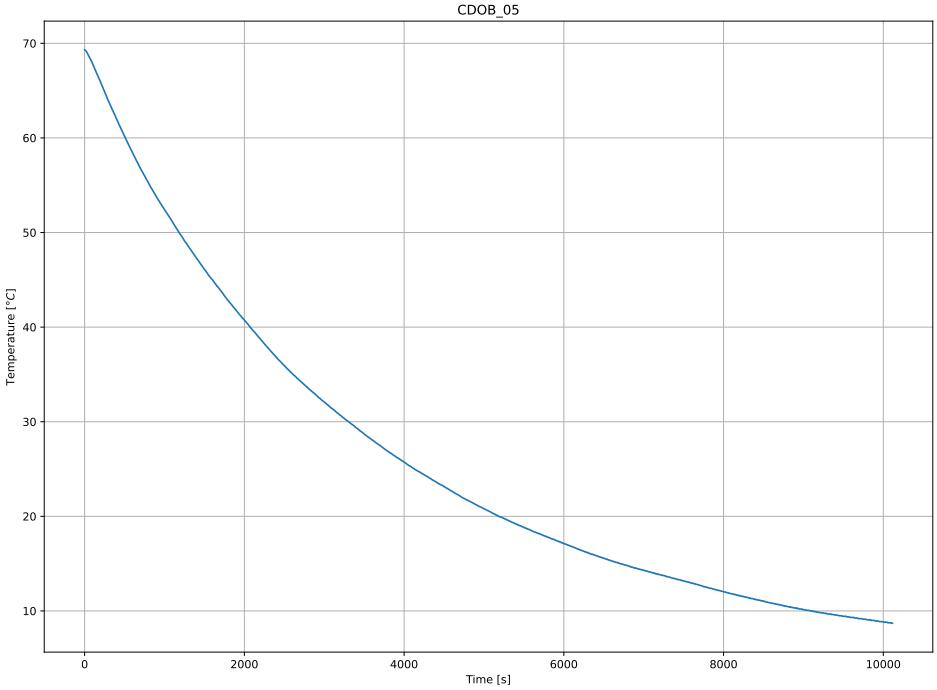


Figure A.10: Filtered Object Temperature Data for CDOB-05

A.6. CDOB_06.csv

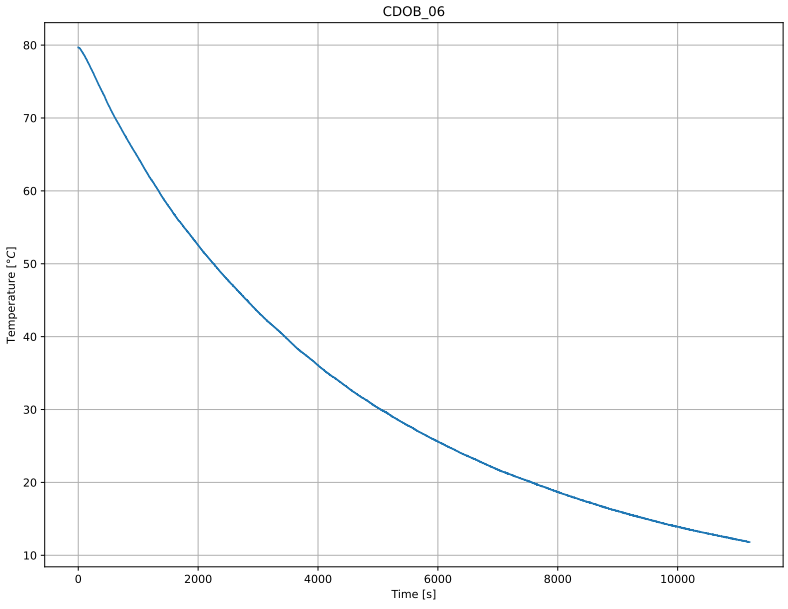


Figure A.11: Raw Object Temperature Data for CDOB-06

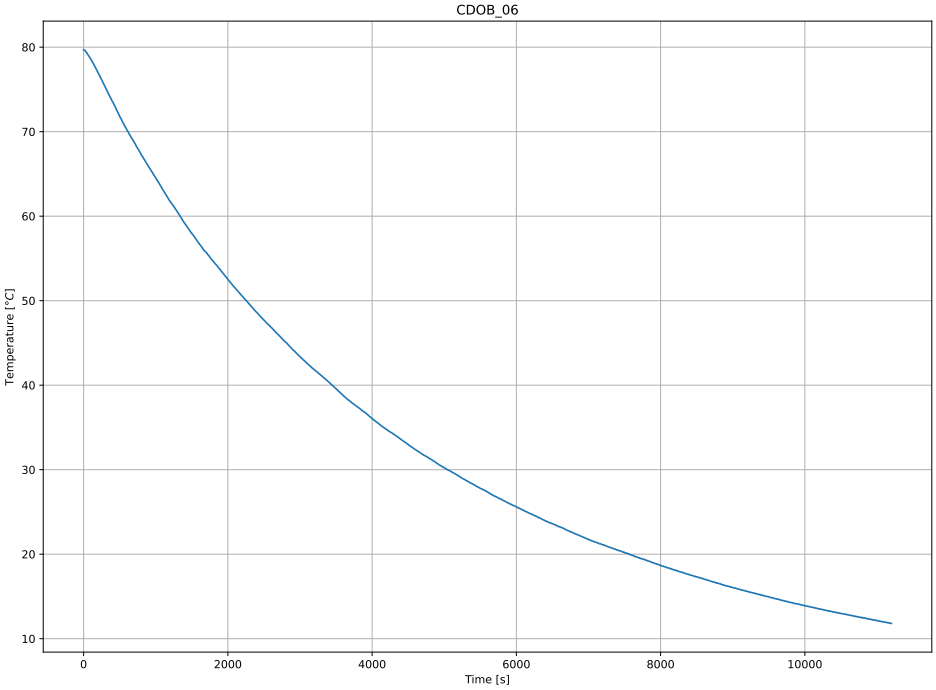


Figure A.12: Filtered Object Temperature Data for CDOB-06

A.7. CDOB_07.csv

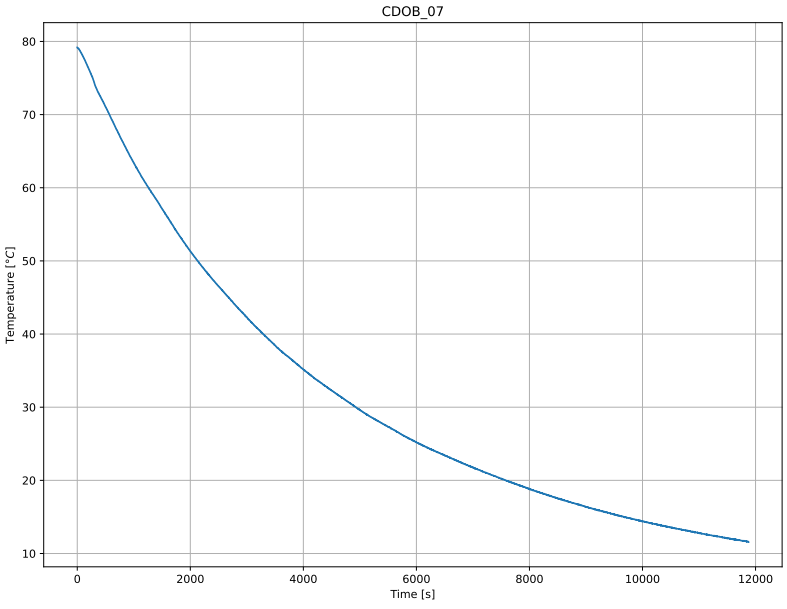


Figure A.13: Raw Object Temperature Data for CDOB-07

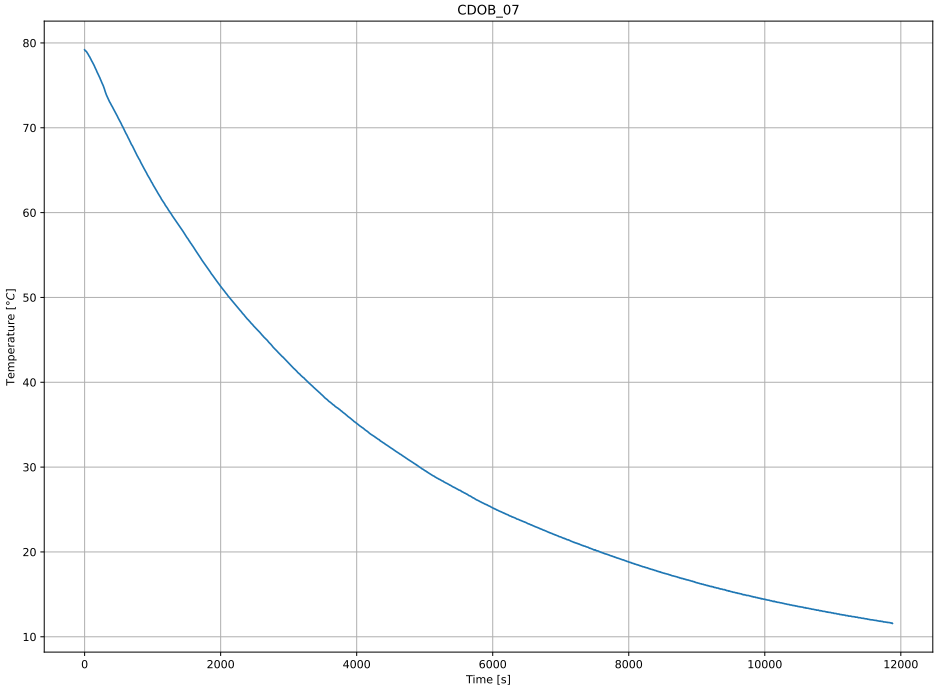


Figure A.14: Filtered Object Temperature Data for CDOB-07

A.8. CDOB_08.csv

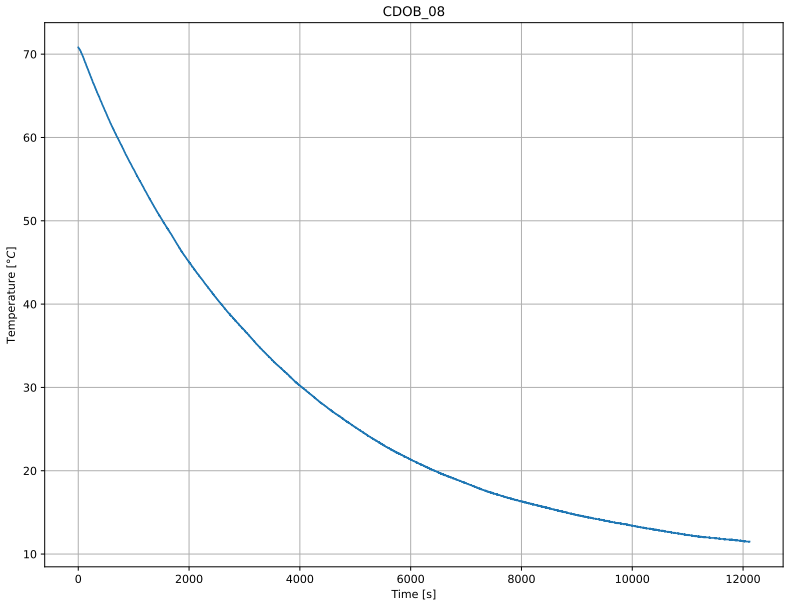


Figure A.15: Raw Object Temperature Data for CDOB-08

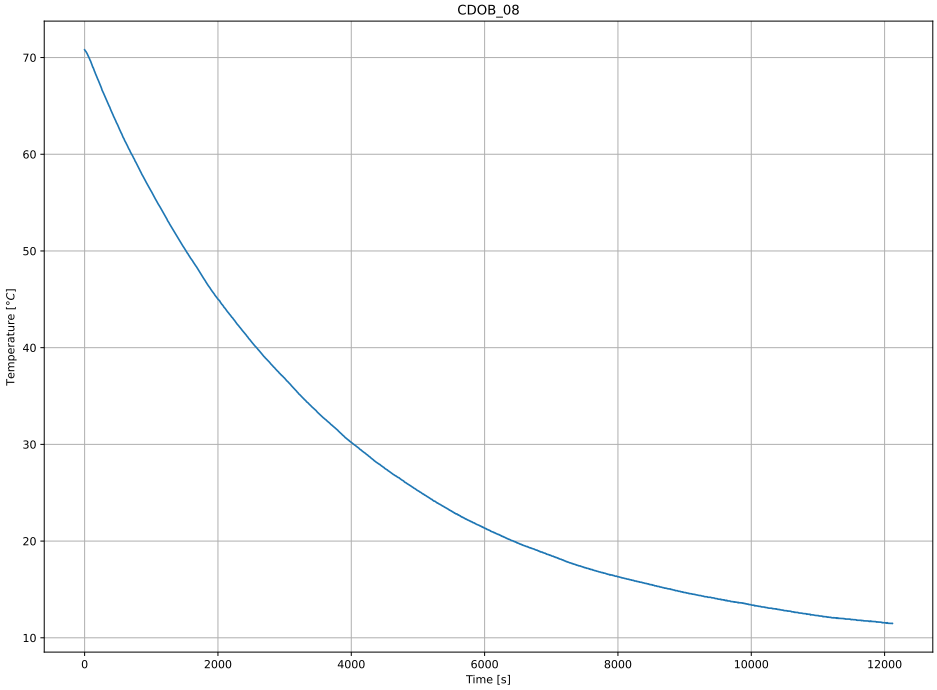


Figure A.16: Filtered Object Temperature Data for CDOB-08

B

Datasheet DHT 11

In this appendix, the datasheet of the DHT 11 temperature sensor is presented. The datasheet can be found in the following link [13]: DHT 11 Datasheet

C

Datasheet DS18B20

In this appendix, the datasheet of the DS18B20 temperature sensor is presented. The datasheet can be found in the following link [70]: DS18B20 Datasheet

D

Experimental data figures of the pressure coefficient distributions

In this appendix, the visualized data of the pressure coefficient distribution recorded from the NACA 0012 airfoil are presented. Each section provides an individual dataset and is named according to the file name as presented in Table 12.1

D.1. Cp_alpha_0169_Re6.txt

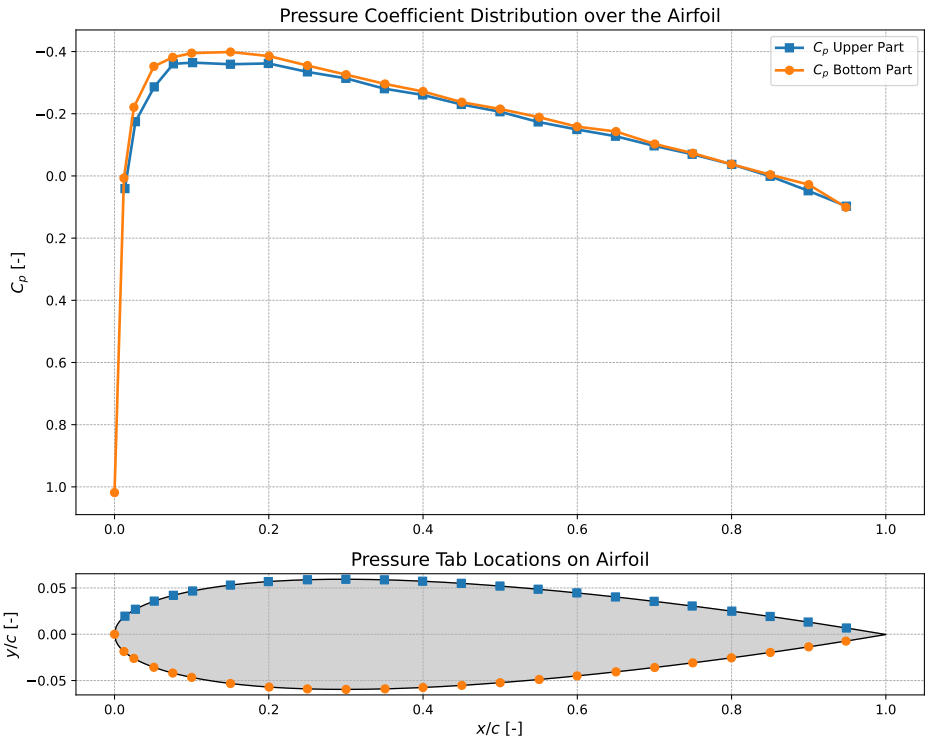


Figure D.1: Pressure Coefficient Distribution around a NACA 0012 airfoil at $\alpha = 0.0169$ [°], $Re = 6 \cdot 10^6$ and $M = 0.3$ [−]. The bottom part of the figure shows the location of the pressure tabs on the airfoil. Data obtained from [61]

D.2. Cp_alpha_10_0254_Re6.txt

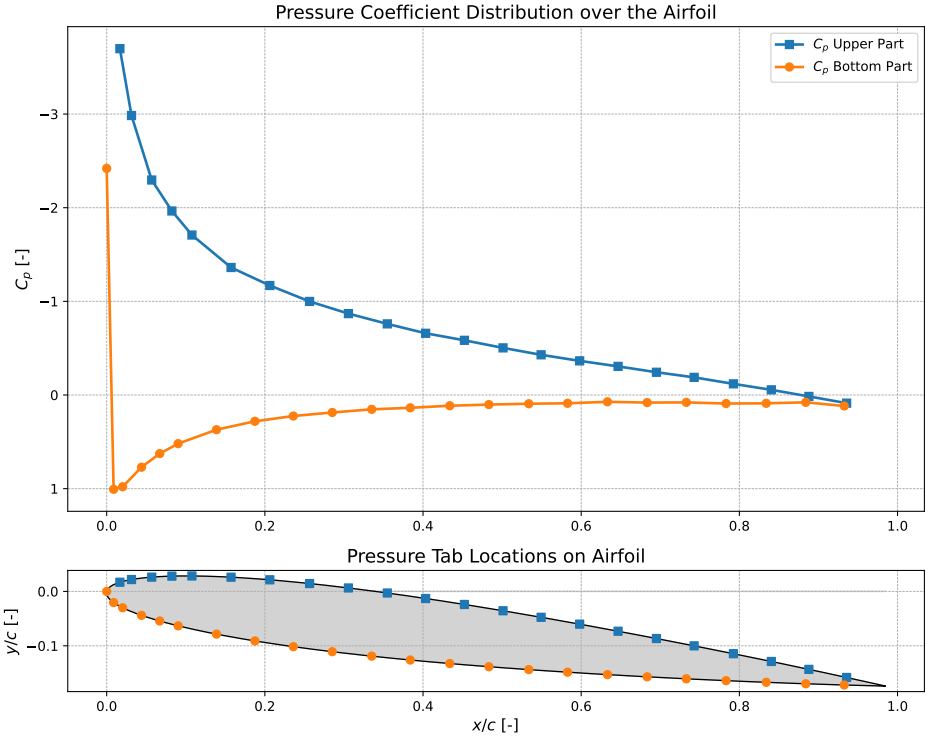


Figure D.2: Pressure Coefficient Distribution around a NACA 0012 airfoil at $\alpha = 10.0254\,[°]$, $Re = 6 \cdot 10^6$ and $M = 0.3\,[-]$. The bottom part of the figure shows the location of the pressure tabs on the airfoil. Data obtained from [61]

D.3. Cp_alpha_15_0299_Re6.txt

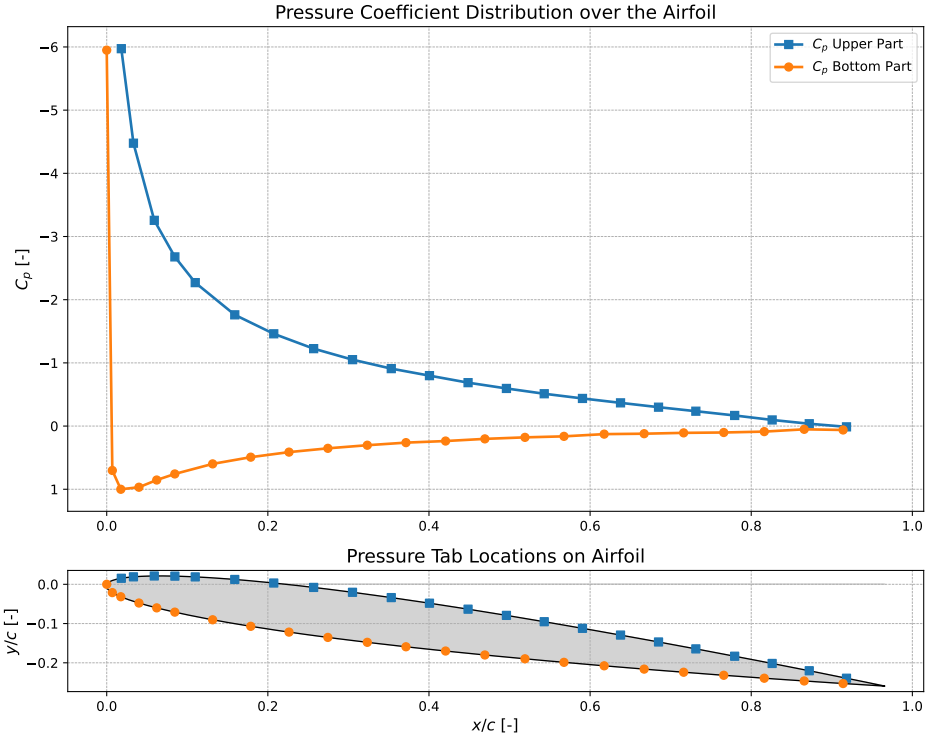


Figure D.3: Pressure Coefficient Distribution around a NACA 0012 airfoil at $\alpha = 15.0299$ [°], $Re = 6 \cdot 10^6$ and $M = 0.3$ [−]. The bottom part of the figure shows the location of the pressure tabs on the airfoil. Data obtained from [61]

E

DARE BEM Technical Drawings

In this appendix, the technical drawings of the DARE BEM are presented. The overall geometry of the DARE BEM can be seen in Figure E.1. The geometry of the various nozzle inserts can be seen in Figure E.2. The geometry of the various grain configurations can be seen in Figure E.3.

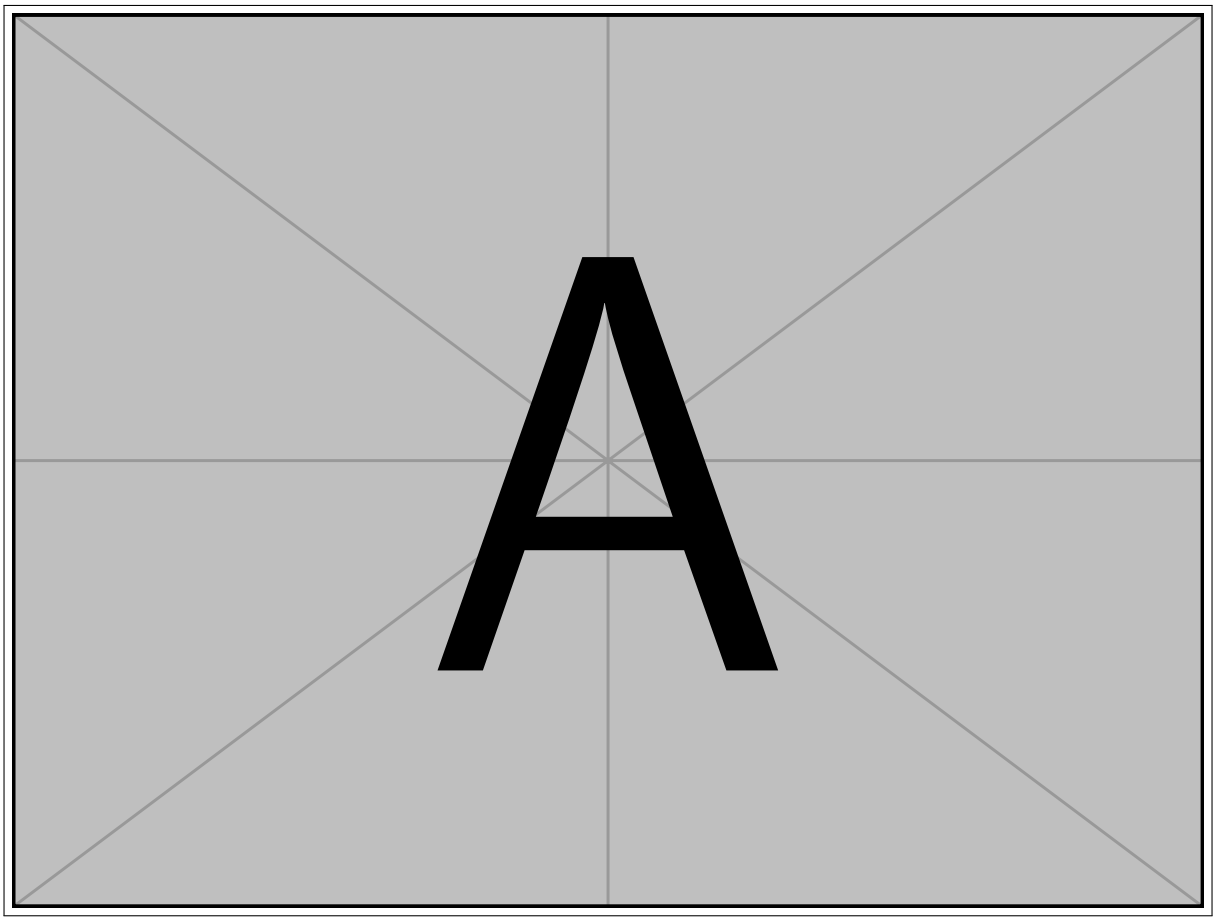


Figure E.1: DARE BEM overall geometry technical drawing [113]

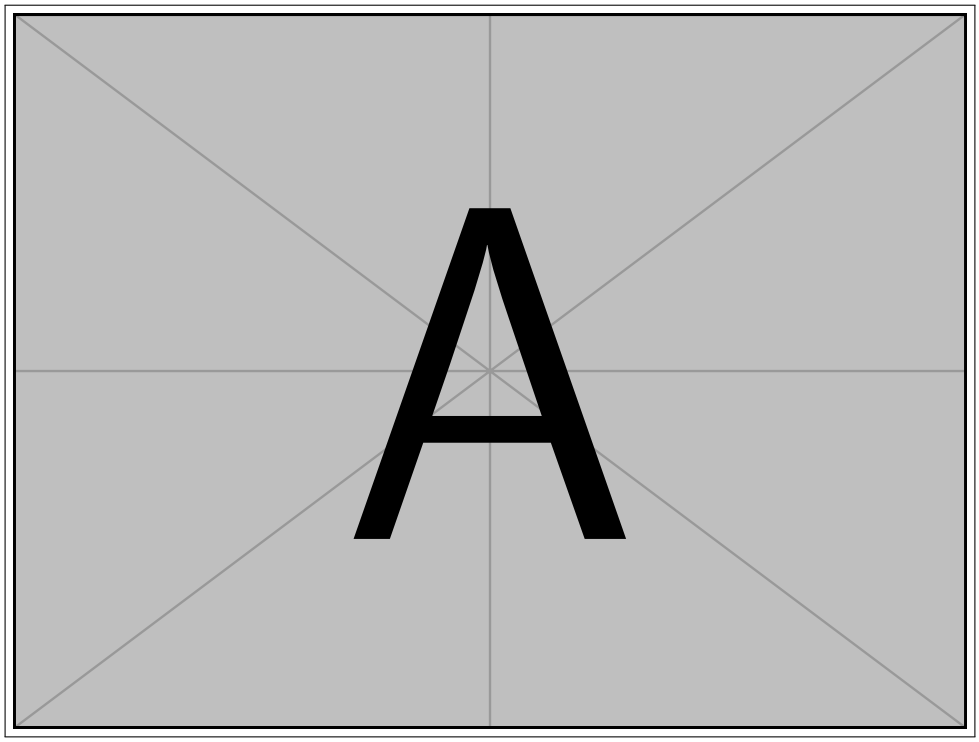


Figure E.2: DARE BEM nozzles geometry technical drawing [113]

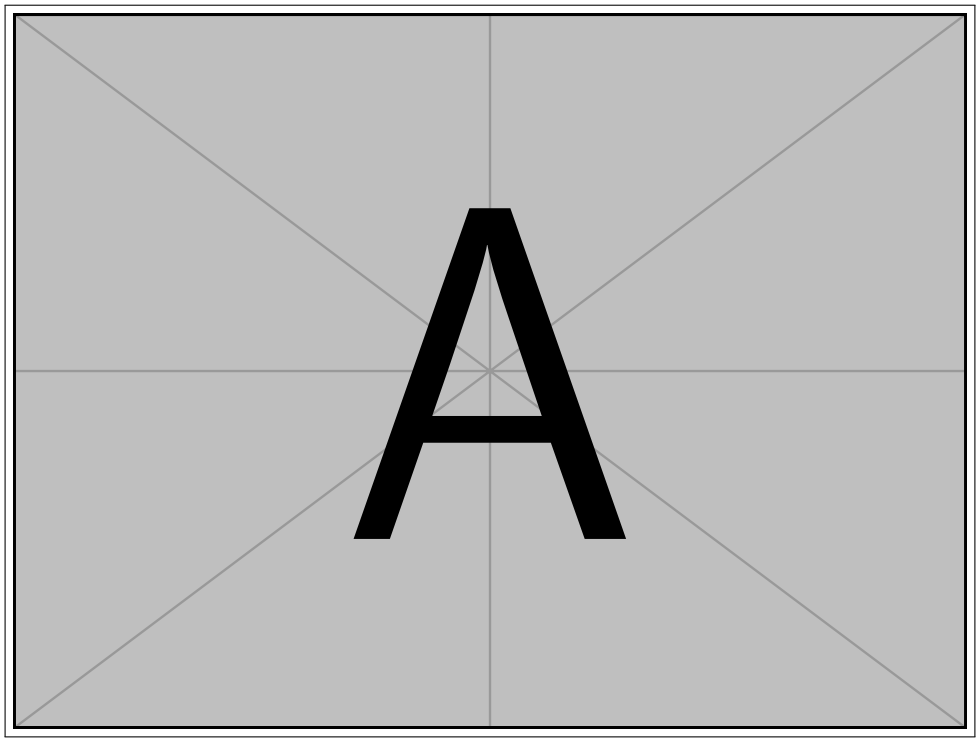


Figure E.3: DARE BEM propellant grain geometry technical drawing [113]

Experimental data figures of the DARE BEM

In this appendix, the visualised data of the experiments performed making use the DARE BEM are presented. Both, the filtered and raw data are presented. Each section provides an individual dataset and is named according to the file name as presented in Table 17.6

F.1. Case2Config1_211222_104543.tdms

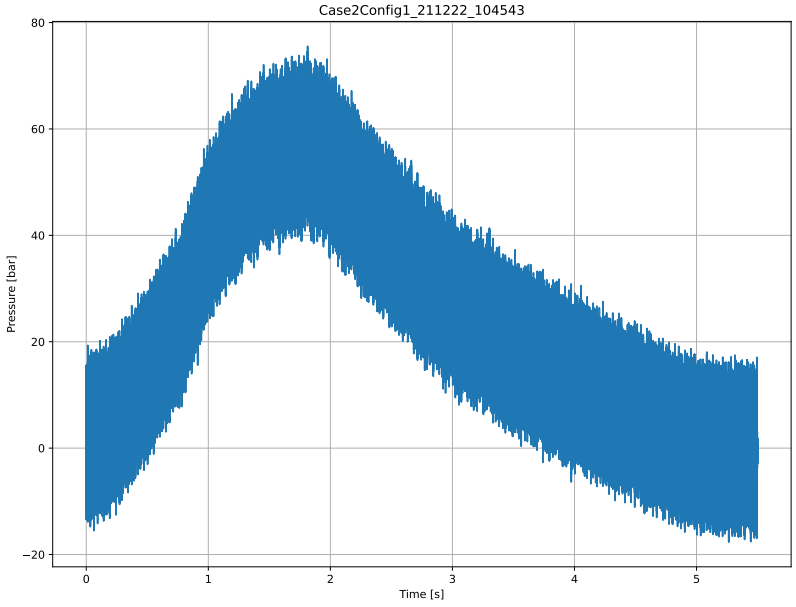


Figure F.1: Raw Pressure Data for Case2Config1

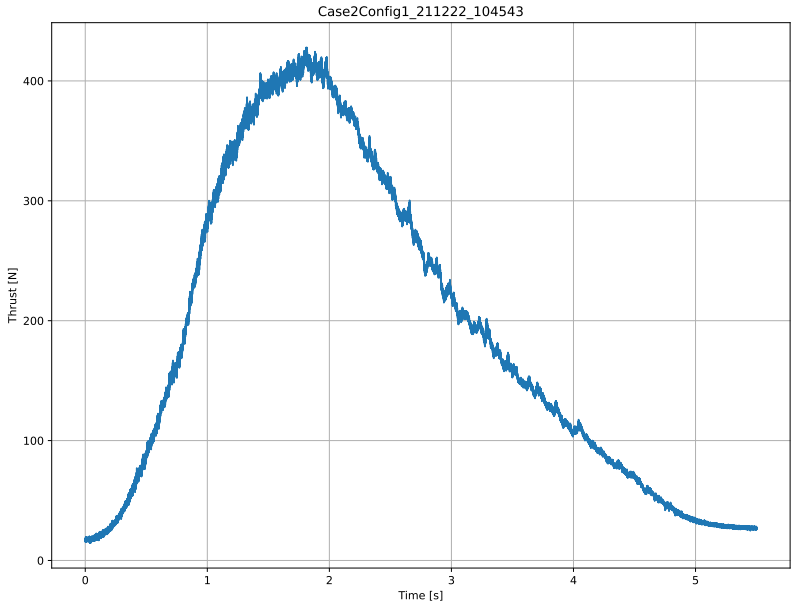


Figure F.2: Raw Thrust Data for Case2Config1

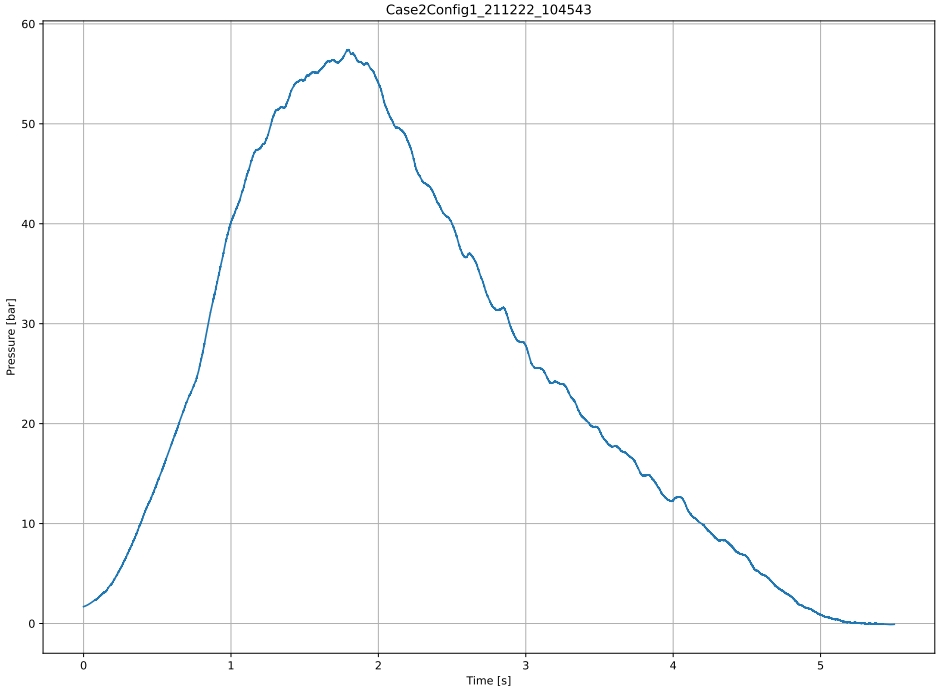


Figure F.3: Filtered Pressure Data for Case2Config1

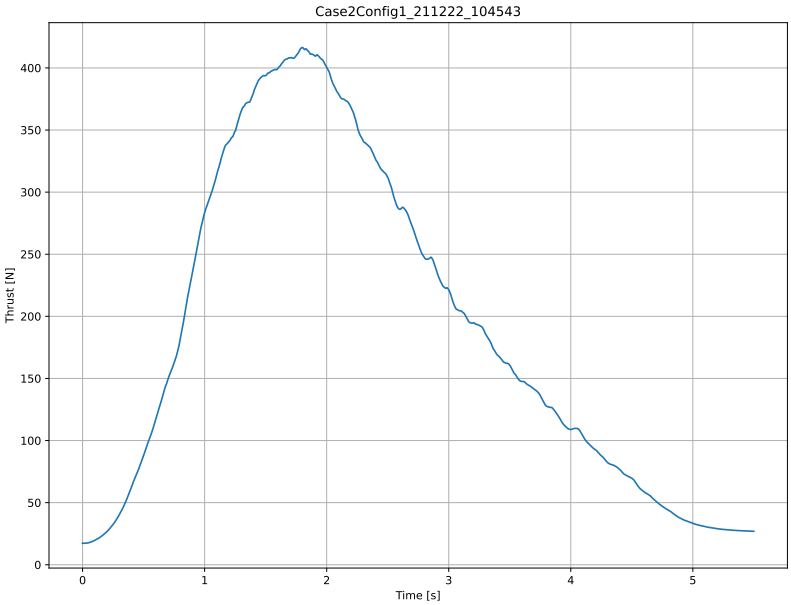


Figure F.4: Filtered Thrust Data for Case2Config1

F.2. Case2Config2_211222_131305.tdms

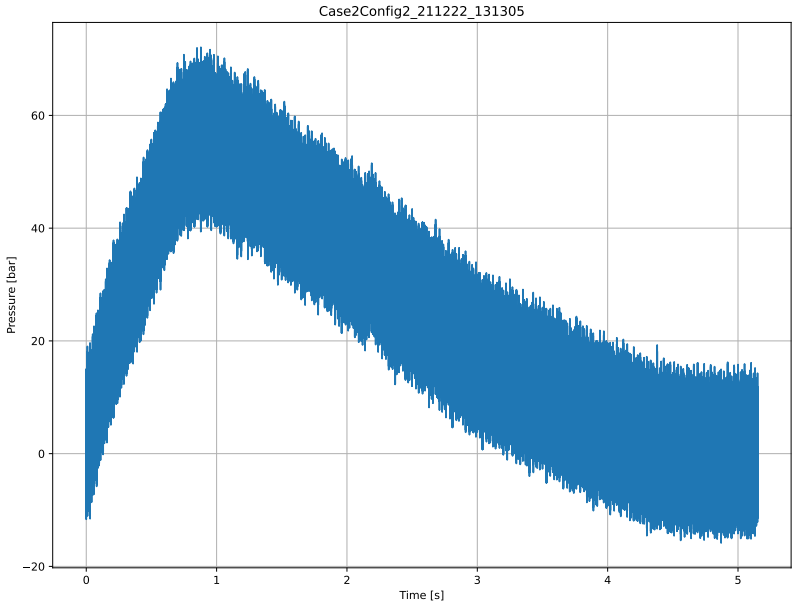


Figure F.5: Raw Pressure Data for Case2Config2

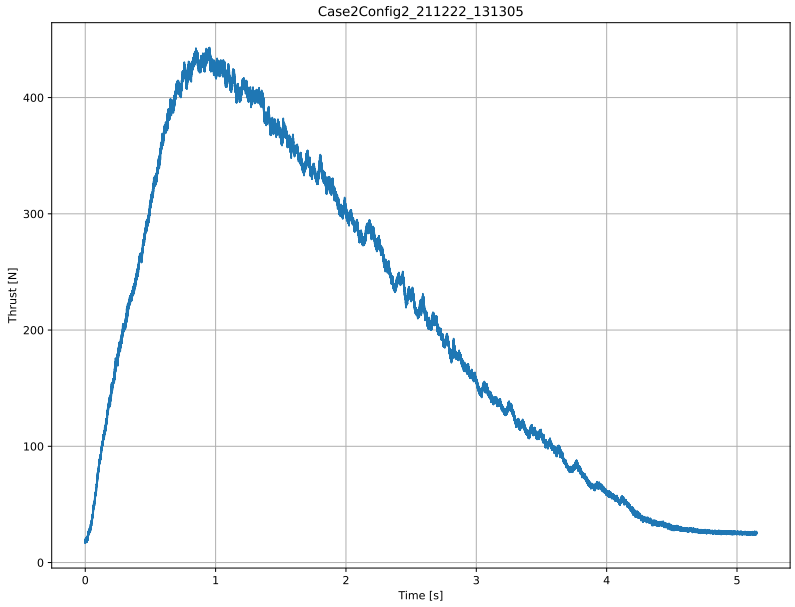


Figure F.6: Raw Thrust Data for Case2Config2

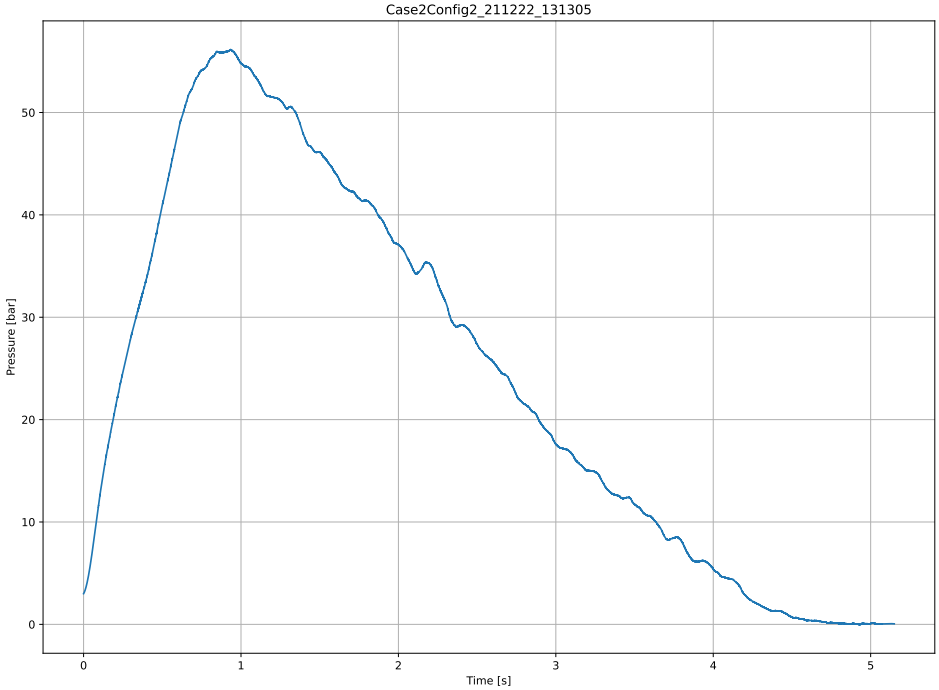


Figure F.7: Filtered Pressure Data for Case2Config2

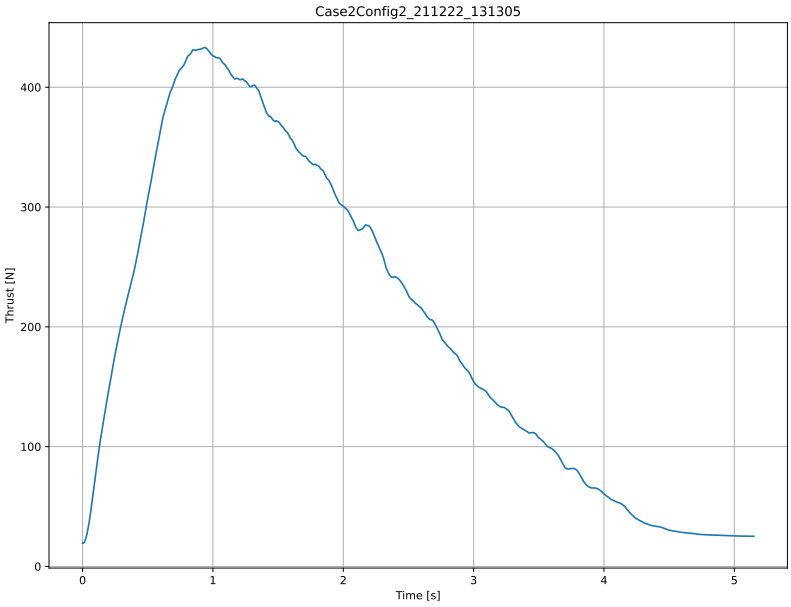


Figure F.8: Filtered Thrust Data for Case2Config2

F.3. Case3Config1_211222_123902.tdms

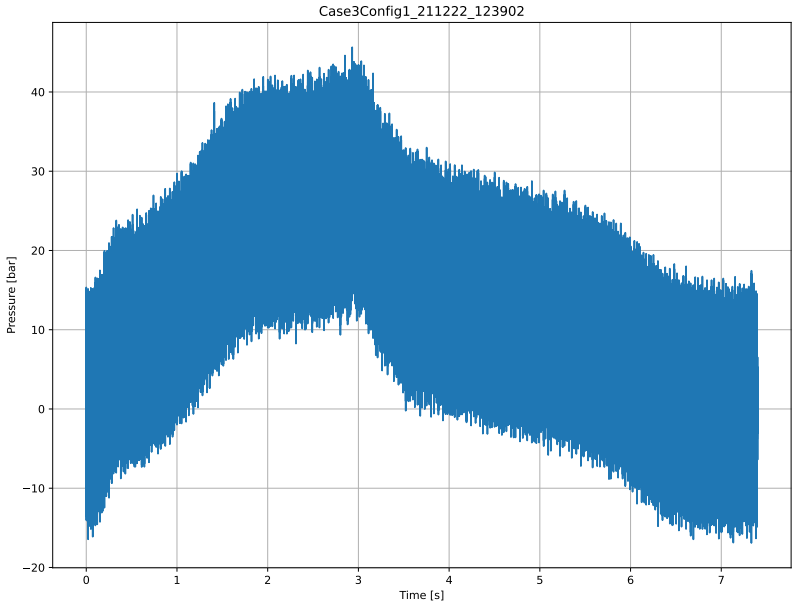


Figure F.9: Raw Pressure Data for Case3Config1

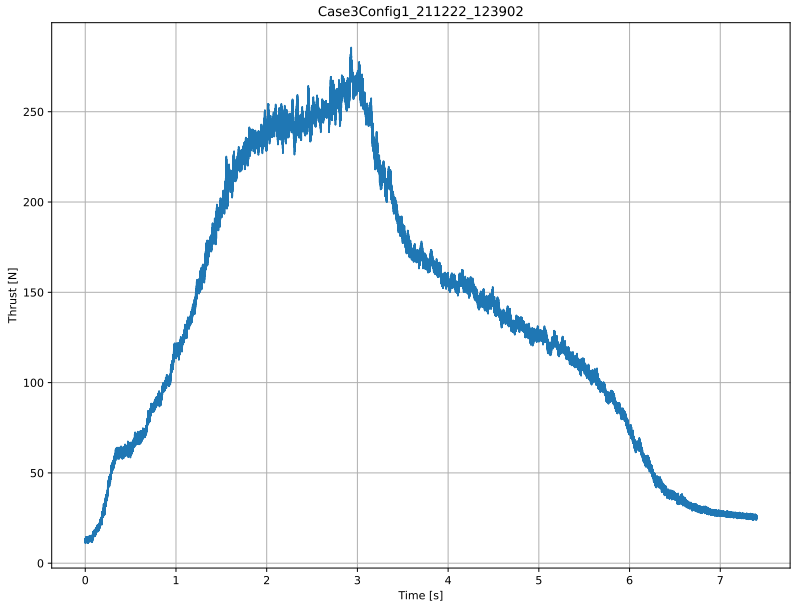


Figure F.10: Raw Thrust Data for Case3Config1

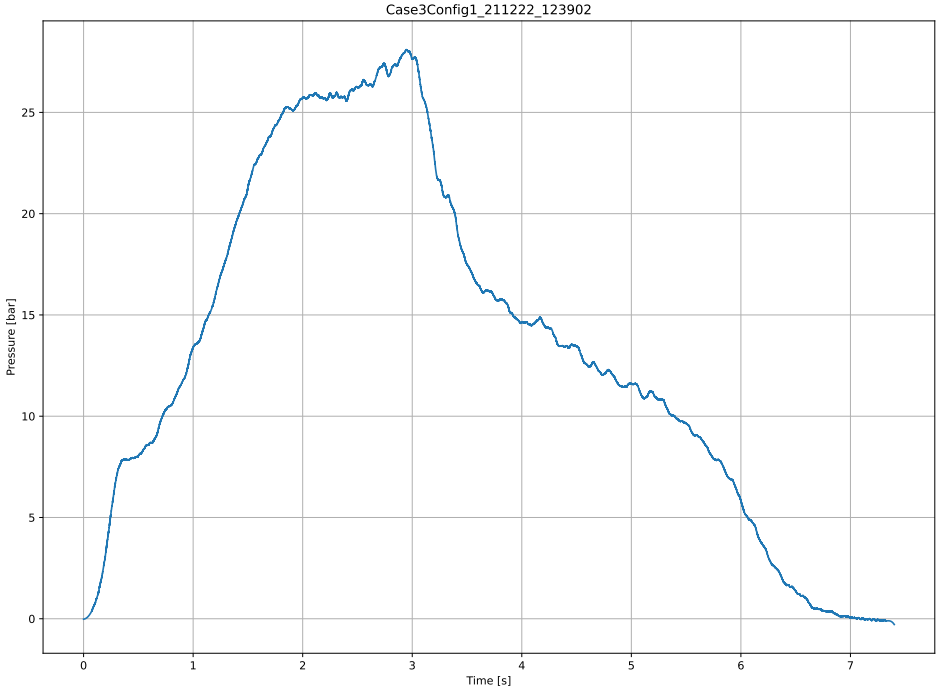


Figure F.11: Filtered Pressure Data for Case3Config1

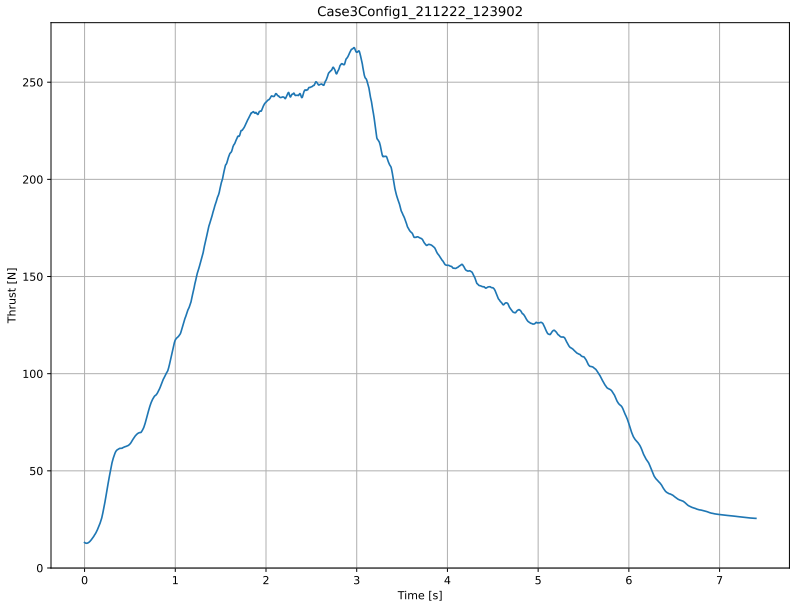


Figure F.12: Filtered Thrust Data for Case3Config1

F.4. Case3Config2_211221_124725.tdms

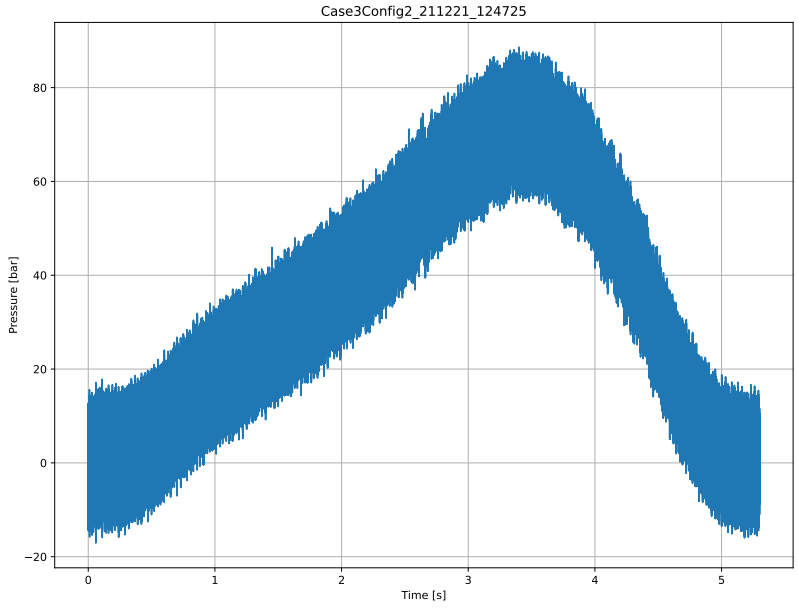


Figure F.13: Raw Pressure Data for Case3Config2

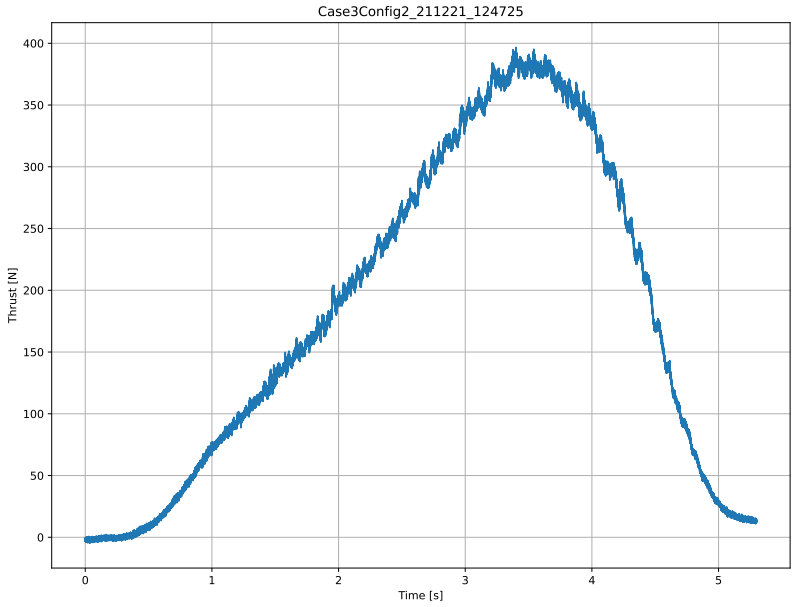


Figure F.14: Raw Thrust Data for Case3Config2

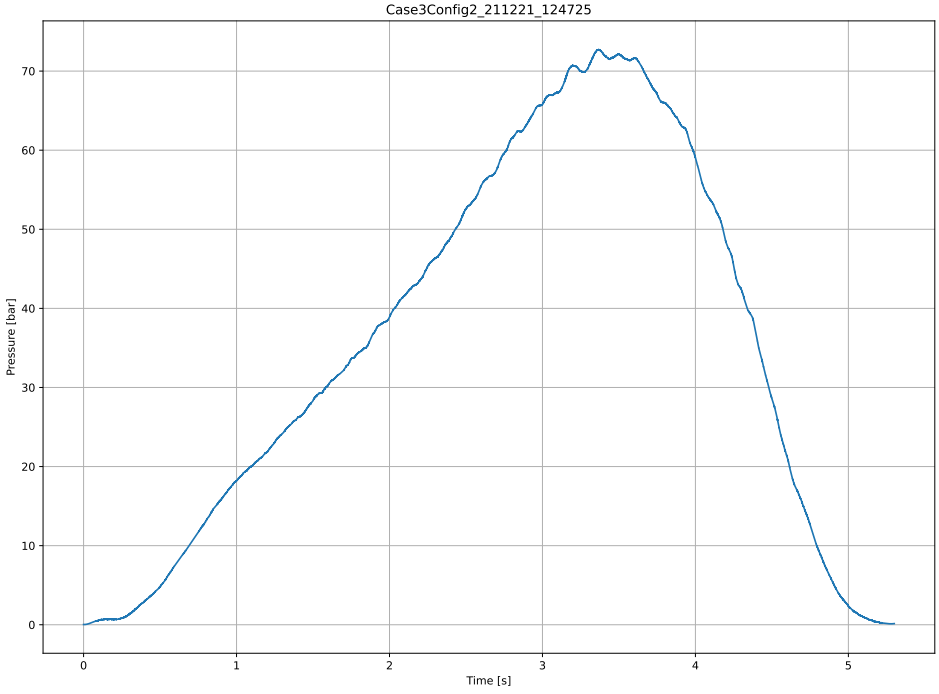


Figure F.15: Filtered Pressure Data for Case3Config2

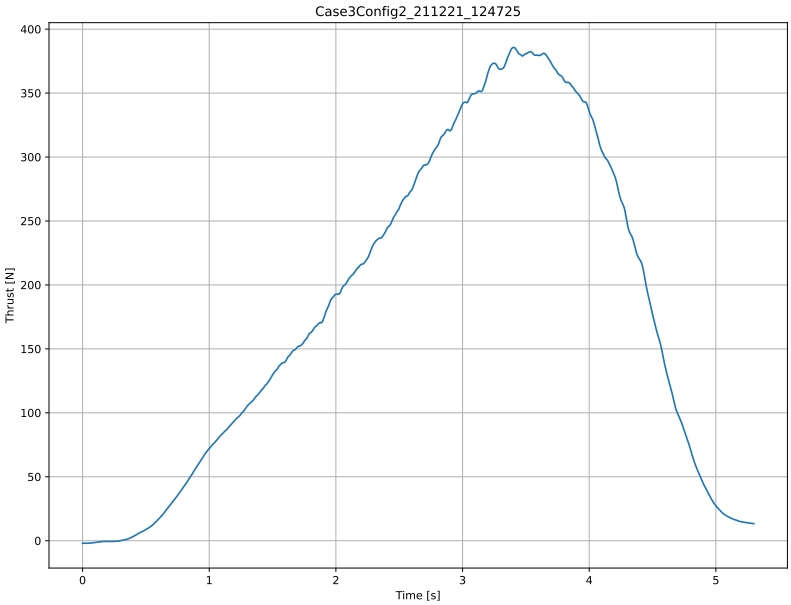


Figure F.16: Filtered Thrust Data for Case3Config2

F.5. ReferenceMotor_211221_1141351.tdms

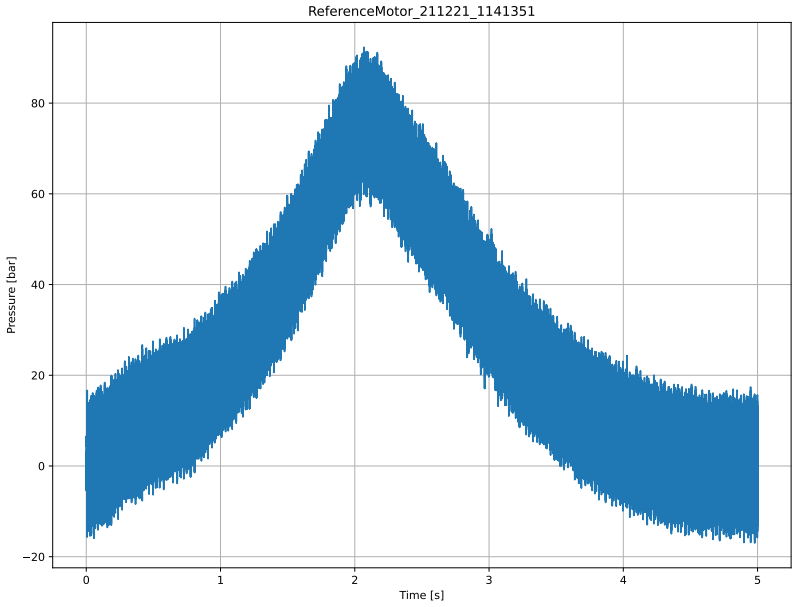


Figure F.17: Raw Pressure Data for ReferenceMotor

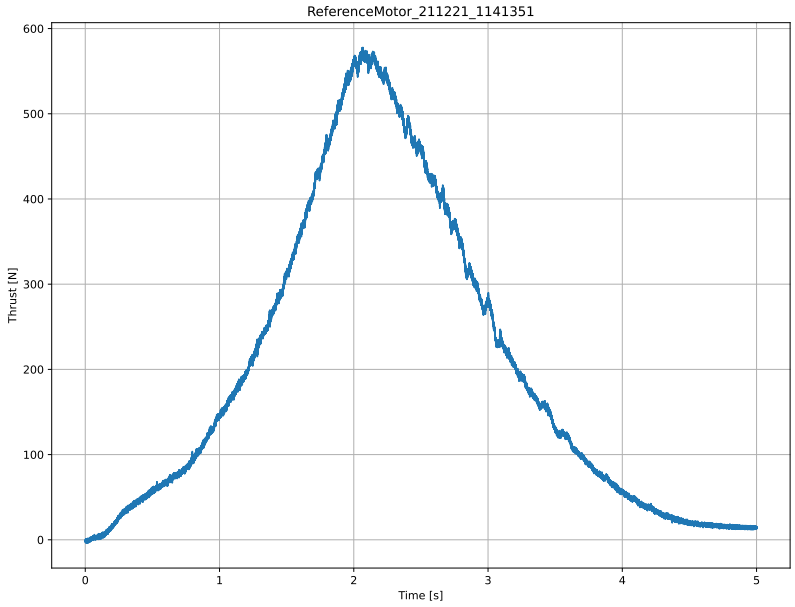


Figure F.18: Raw Thrust Data for ReferenceMotor

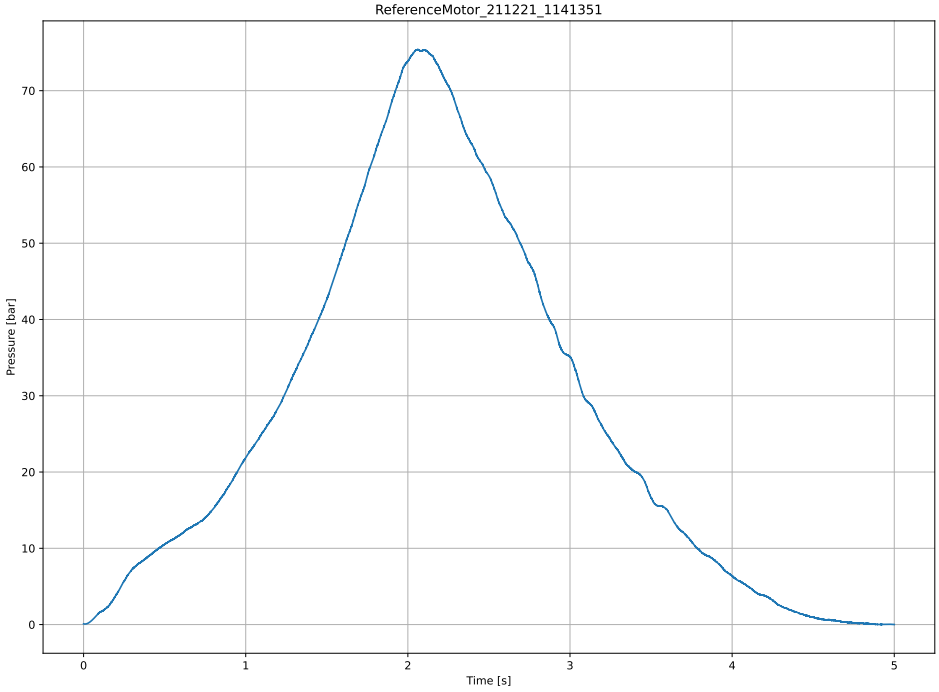


Figure F.19: Filtered Pressure Data for ReferenceMotor

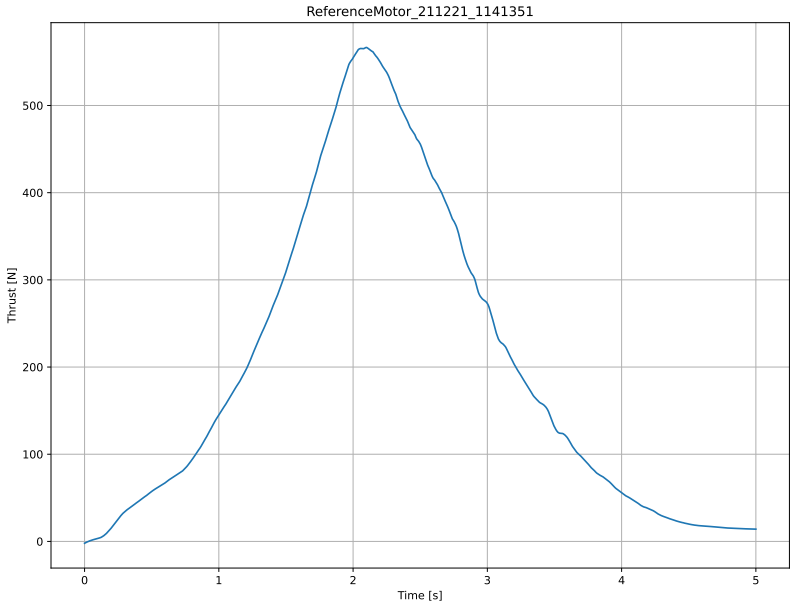


Figure F.20: Filtered Thrust Data for ReferenceMotor

F.6. ReferenceMotor2_211222_092347.tdms

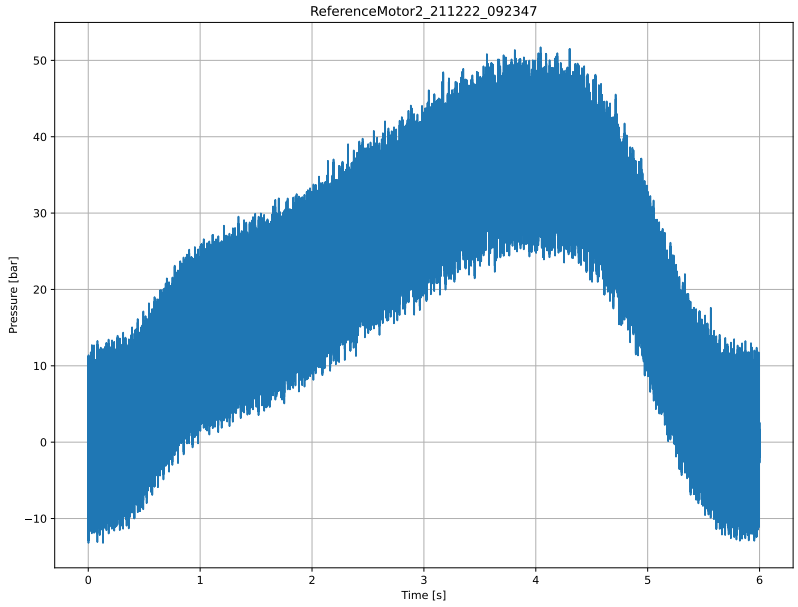


Figure F.21: Raw Pressure Data for ReferenceMotor2

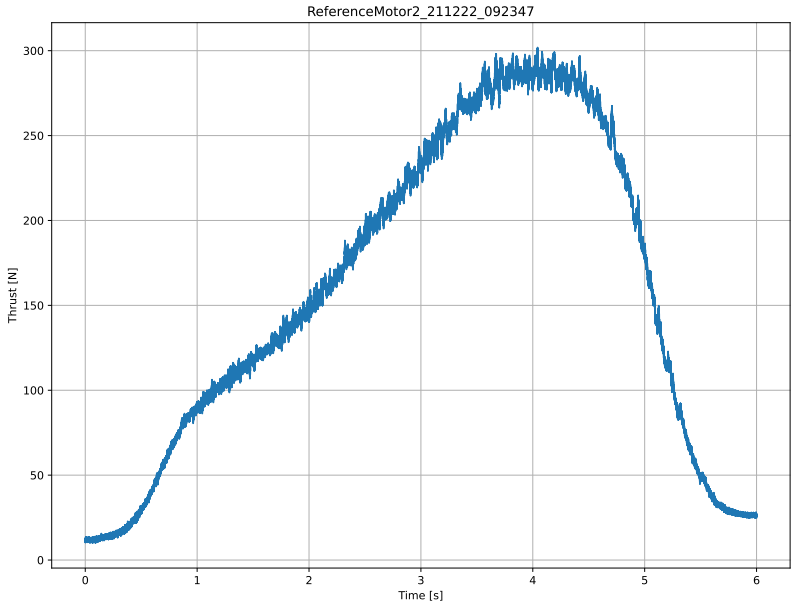


Figure F.22: Raw Thrust Data for ReferenceMotor2

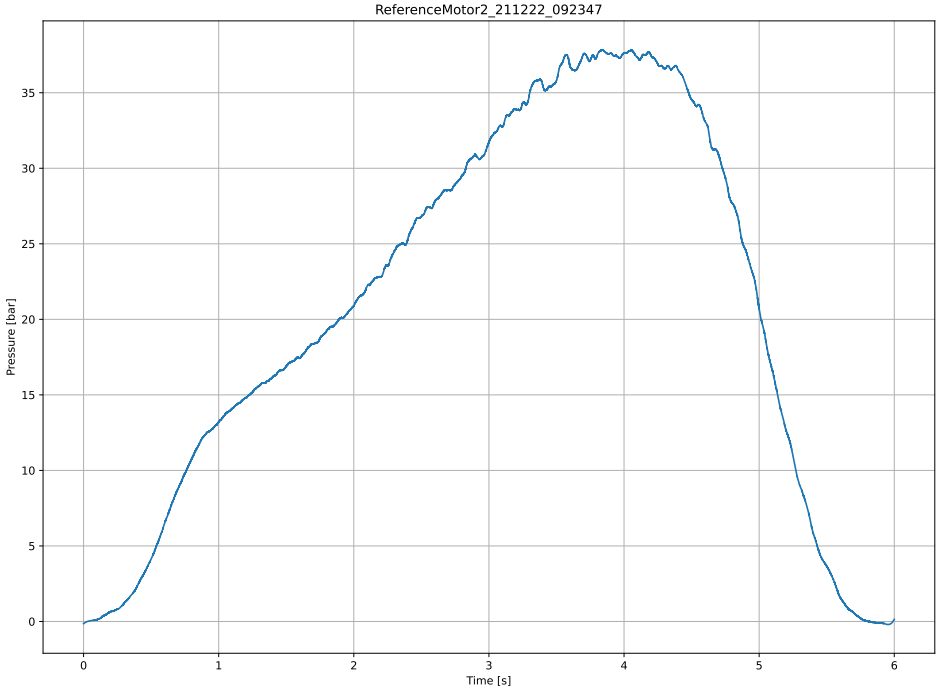


Figure F.23: Filtered Pressure Data for ReferenceMotor2

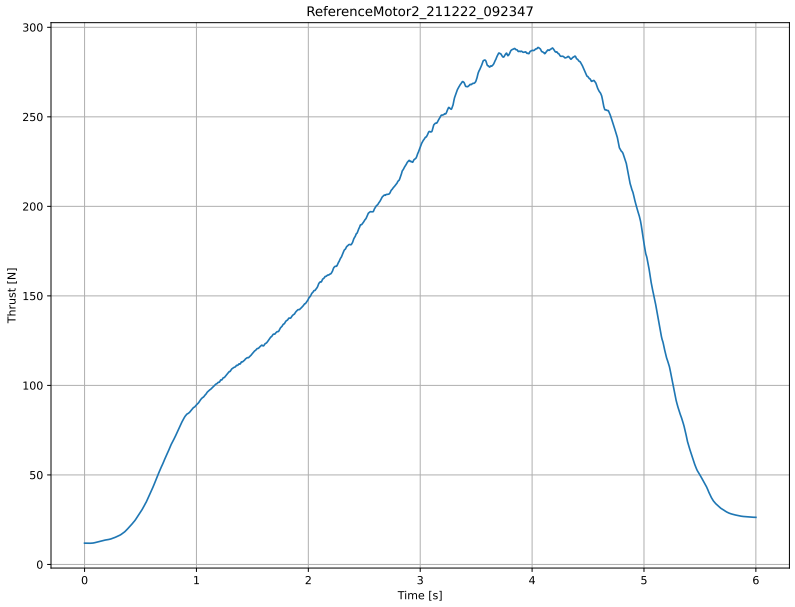


Figure F.24: Filtered Thrust Data for ReferenceMotor2

F.7. 20230327_103125_Ref27Morning.tdms

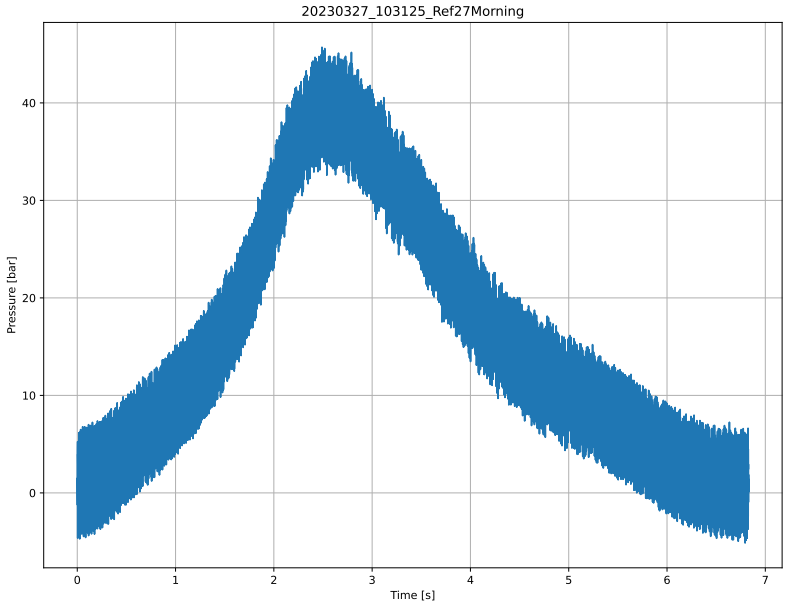


Figure F.25: Raw Pressure Data for 20230327_103125_Ref27Morning

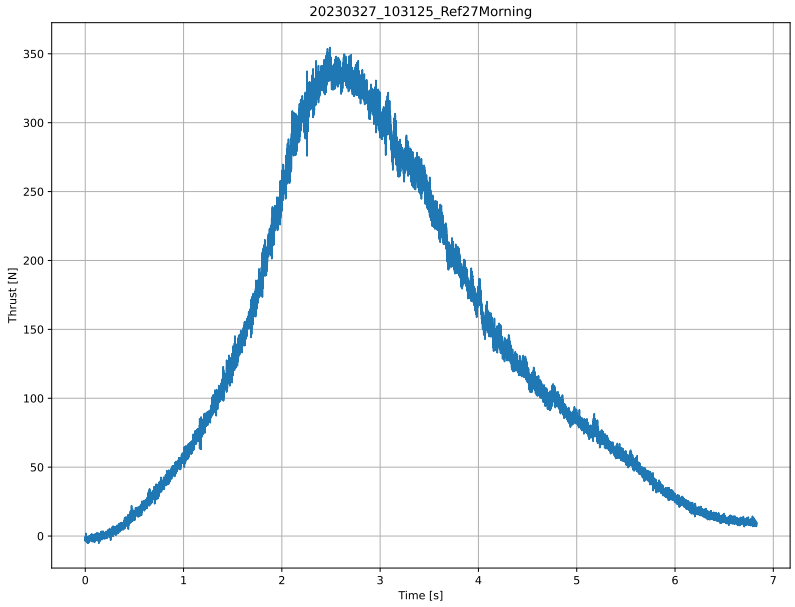


Figure F.26: Raw Thrust Data for 20230327_103125_Ref27Morning

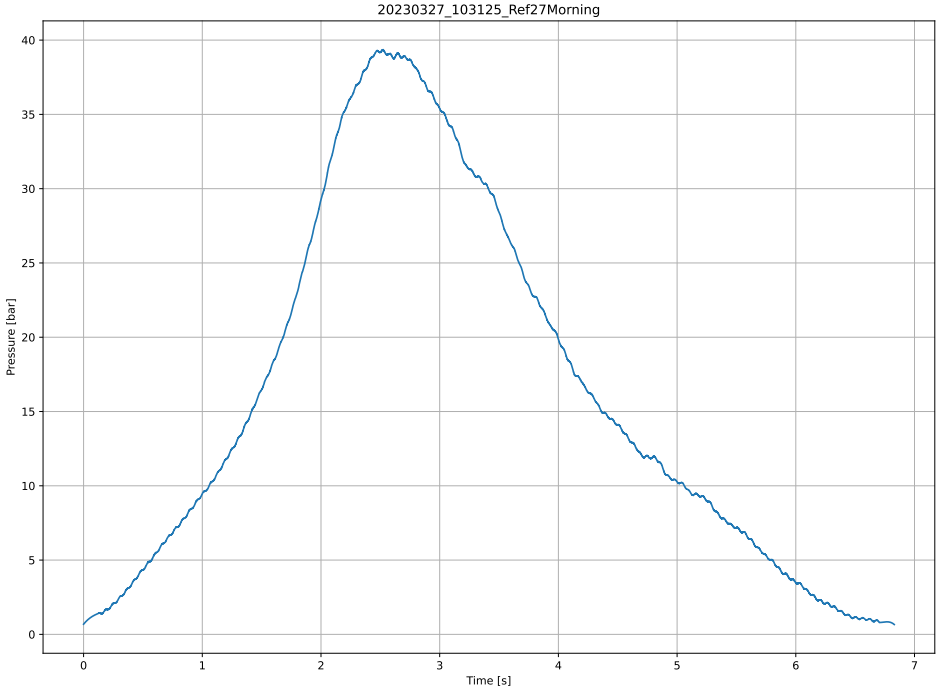


Figure F.27: Filtered Pressure Data for 20230327_103125_Ref27Morning

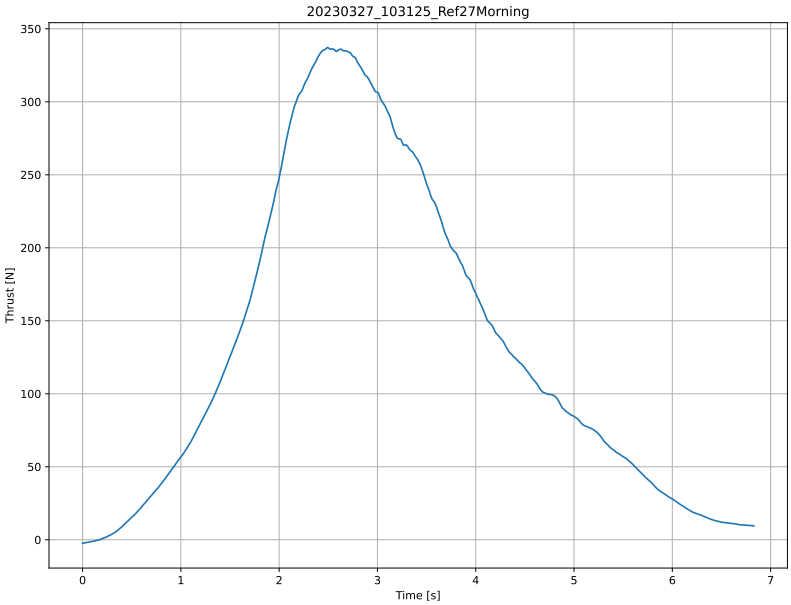


Figure F.28: Filtered Thrust Data for 20230327_103125_Ref27Morning

F.8. 20230327_110844_1per27Morning.tdms

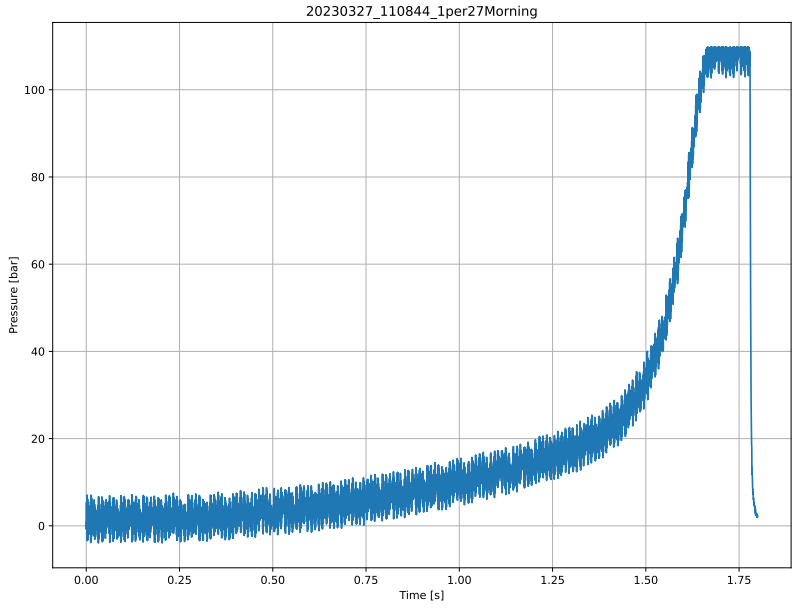


Figure F.29: Raw Pressure Data for 20230327_110844_1per27Morning

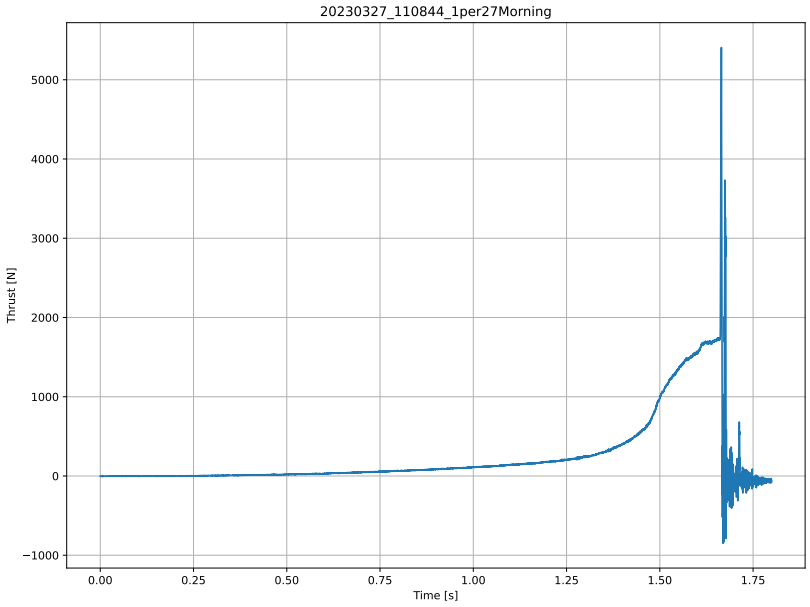


Figure F.30: Raw Thrust Data for 20230327_110844_1per27Morning

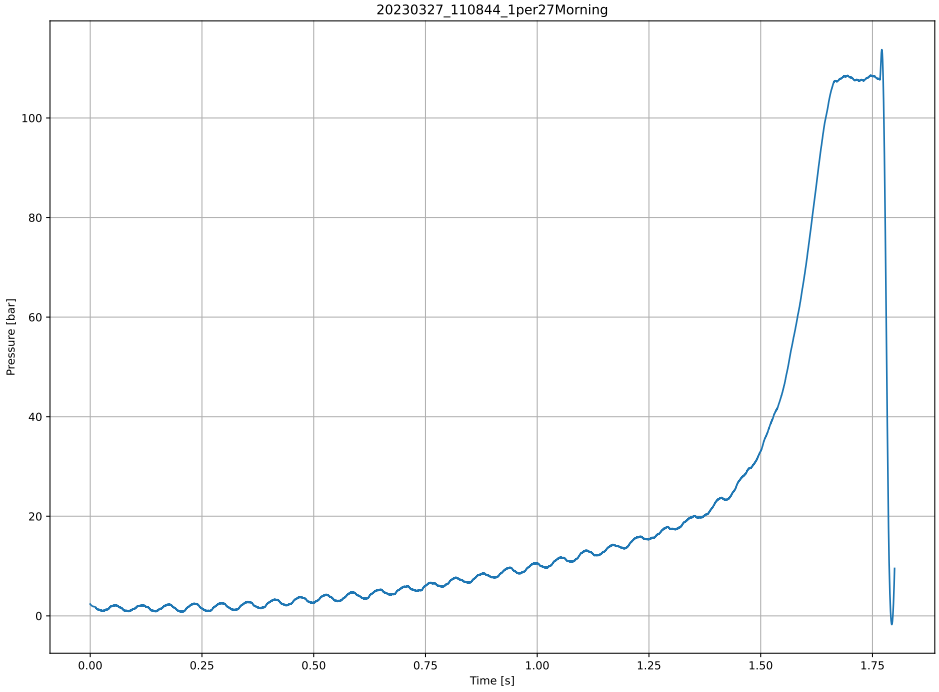


Figure F.31: Filtered Pressure Data for 20230327_110844_1per27Morning

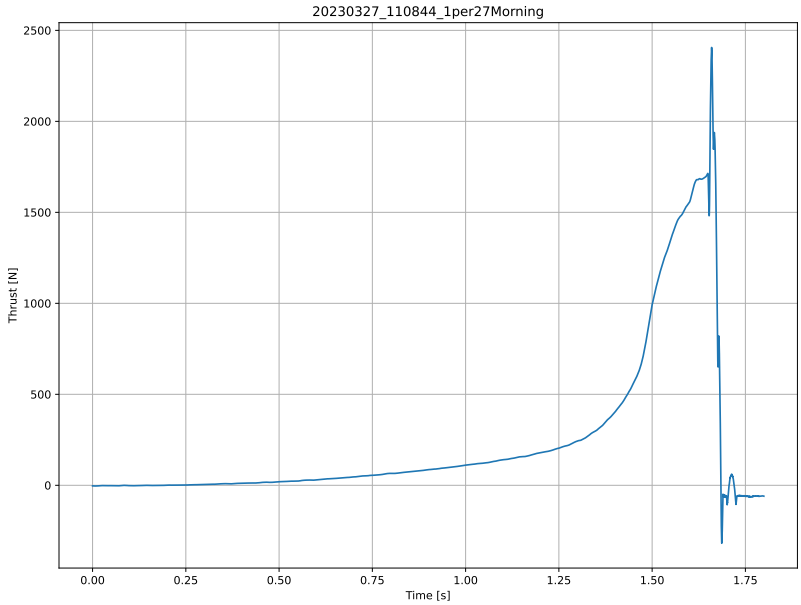


Figure F.32: Filtered Thrust Data for 20230327_110844_1per27Morning

G

DARE SRM 2020 Technical Drawings

In this appendix, the technical drawings of the DARE SRM 2020 and DARE SRM 2020 EE6 are presented. The overall geometry of the DARE SRM 2020 can be seen in Figure E.1, and the overall geometry of the DARE SRM 2020 EE6 can be seen in Figure G.2. The geometry of the nozzle insert can be seen in Figure G.3.

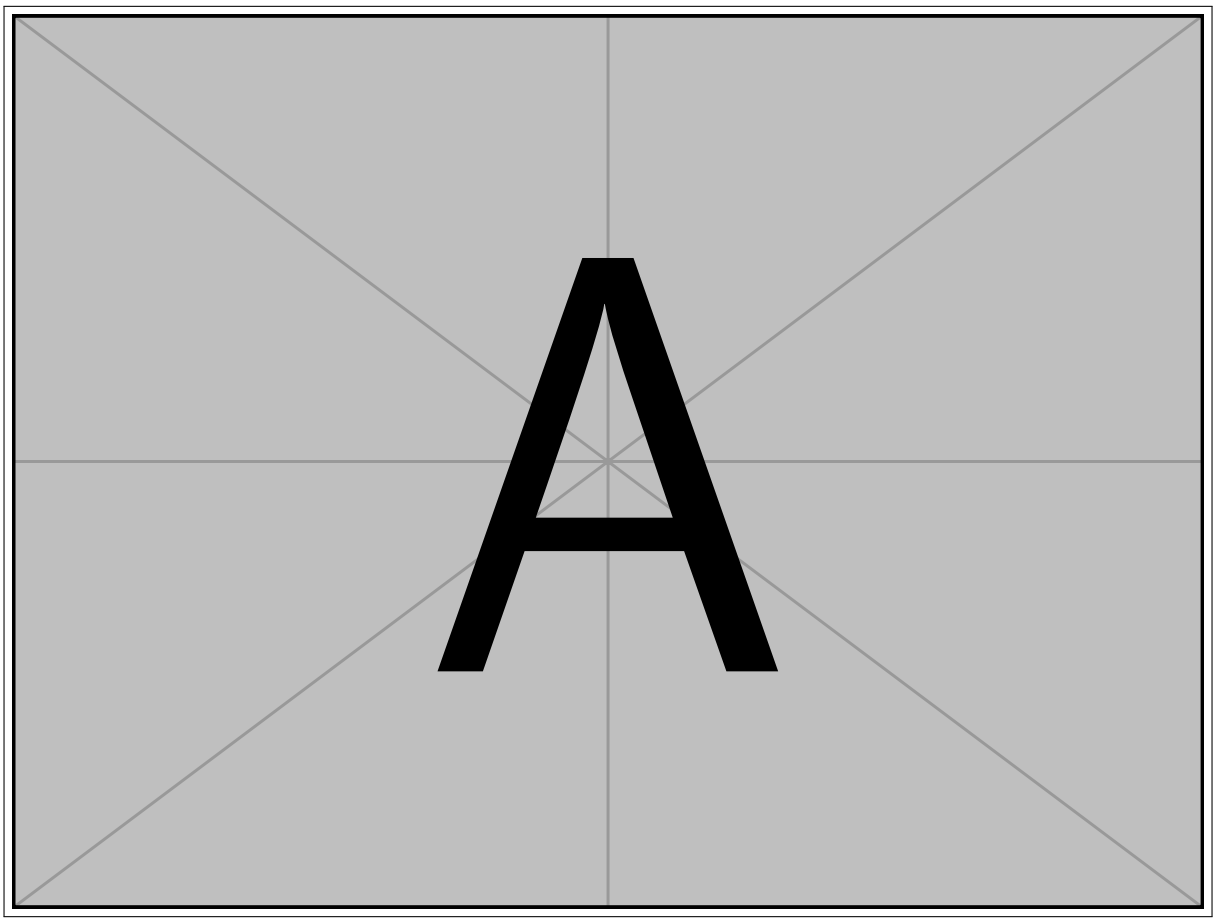


Figure G.1: DARE SRM 2020 overall geometry technical drawing [56]

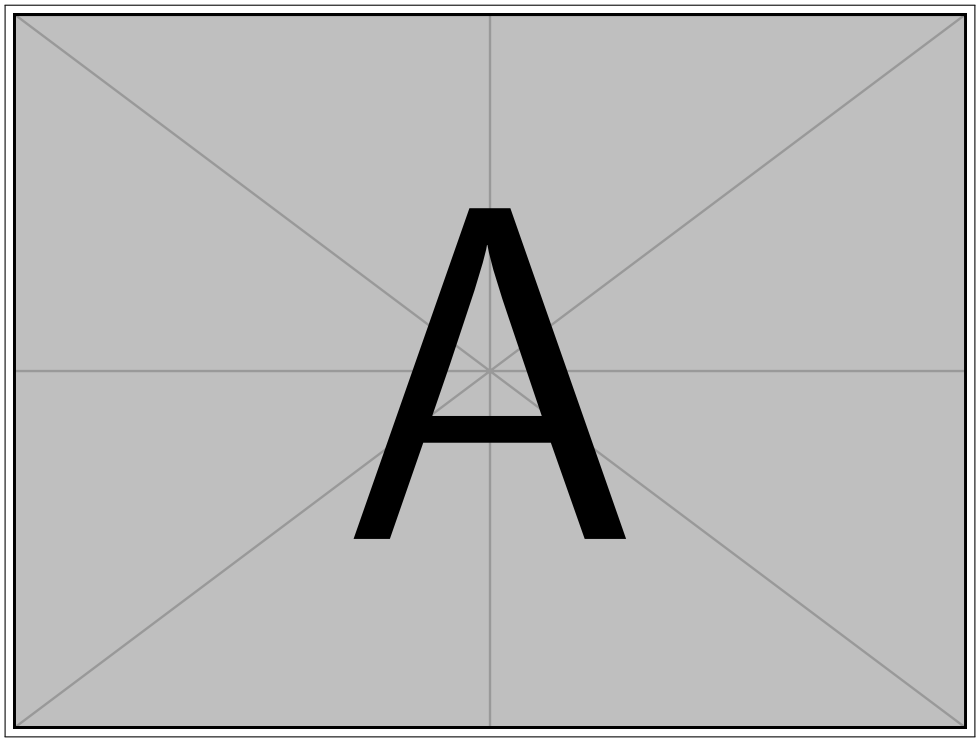


Figure G.2: DARE SRM 2020 EE6 overall geometry technical drawing [69]

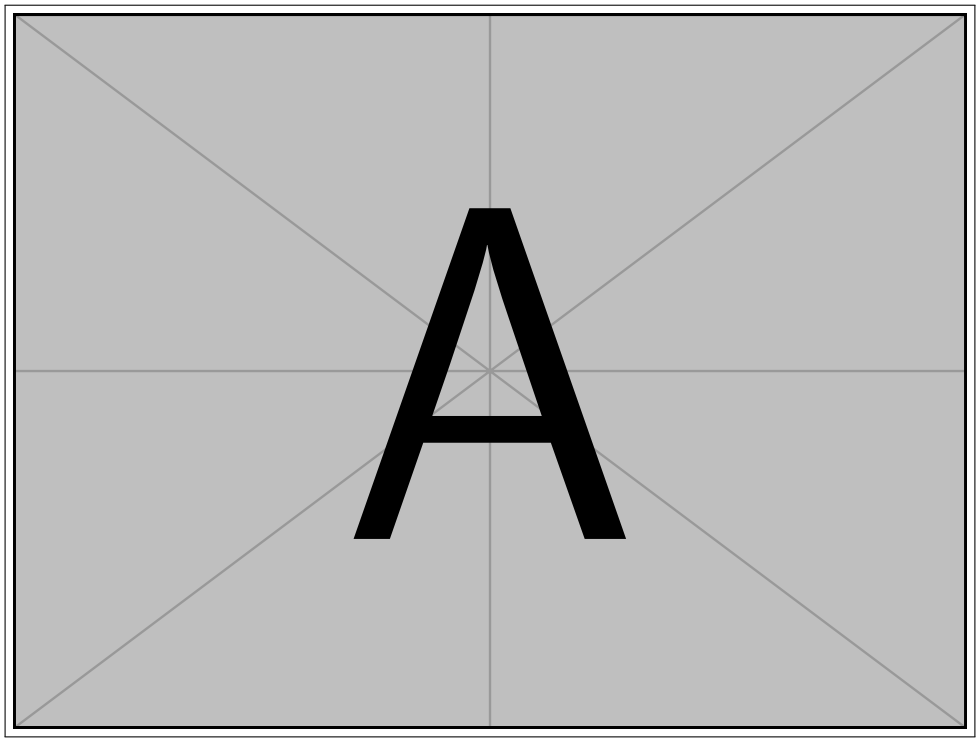


Figure G.3: DARE SRM 2020 (EE6) nozzle geometry technical drawing [48]

H

Experimental data figures of the DARE SRM 2020 (EE6)

In this appendix, the visualised data of the experiments performed using the DARE BEM are presented. Both, the filtered and raw data are presented. Each section provides an individual dataset and is named according to the file name as presented in Table 17.7

H.1. Engine 1 Test 1

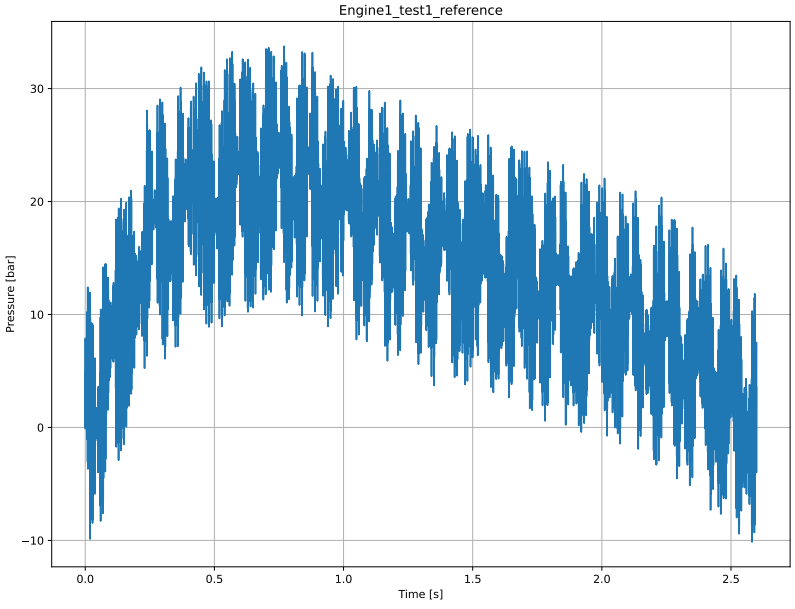


Figure H.1: Raw Pressure Data for Engine 1 Test 1

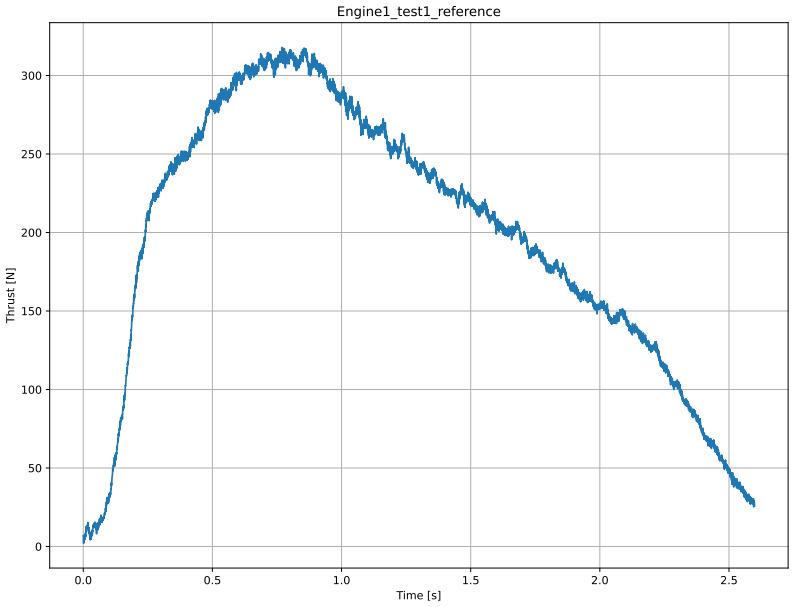


Figure H.2: Raw Thrust Data for Engine 1 Test 1

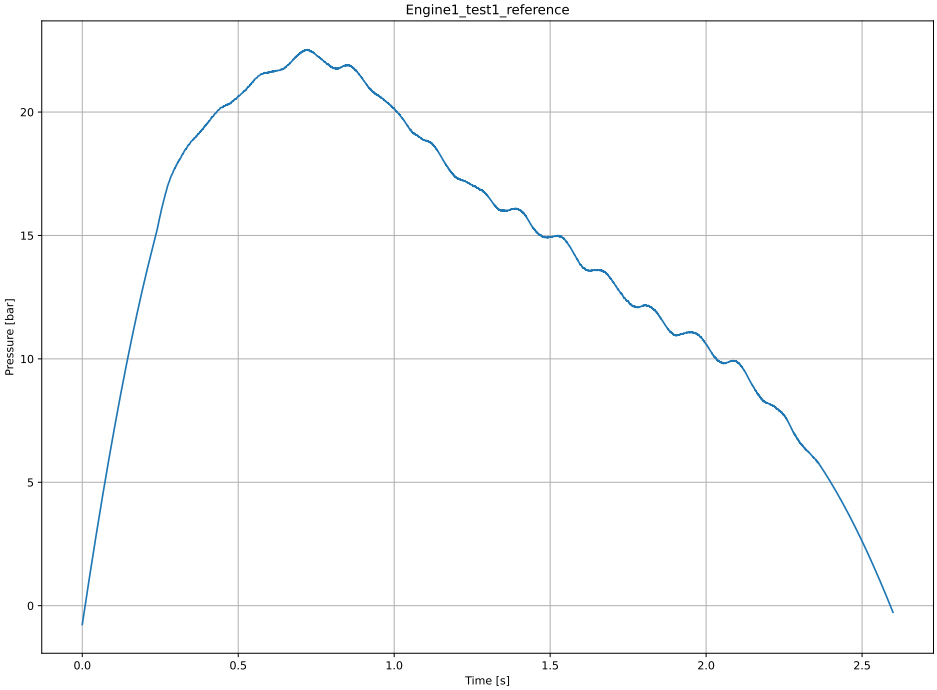


Figure H.3: Filtered Pressure Data for Engine 1 Test 1

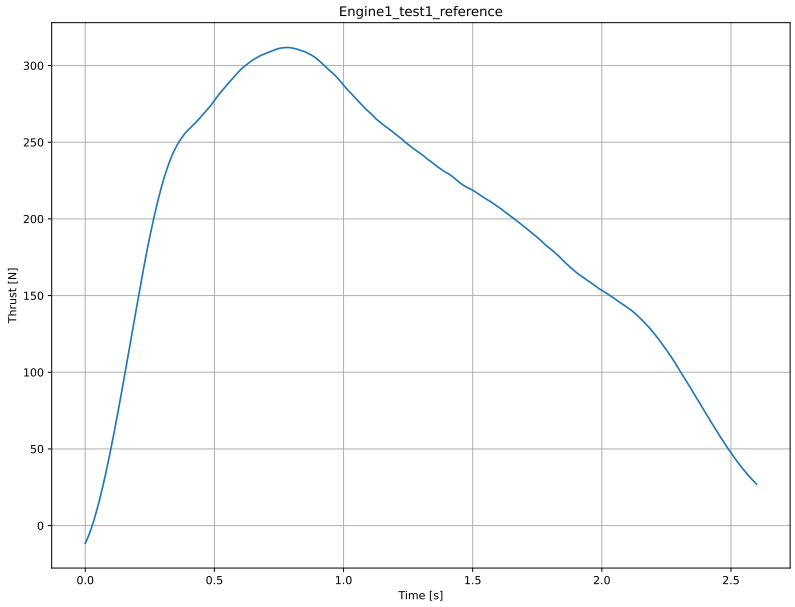


Figure H.4: Filtered Thrust Data for Engine 1 Test 1

H.2. Engine 1 Test 2

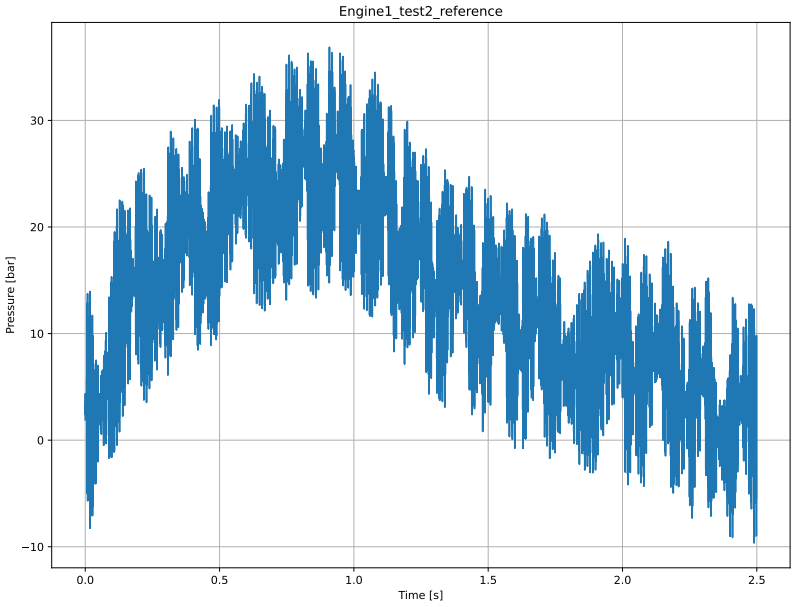


Figure H.5: Raw Pressure Data for Engine 1 Test 2

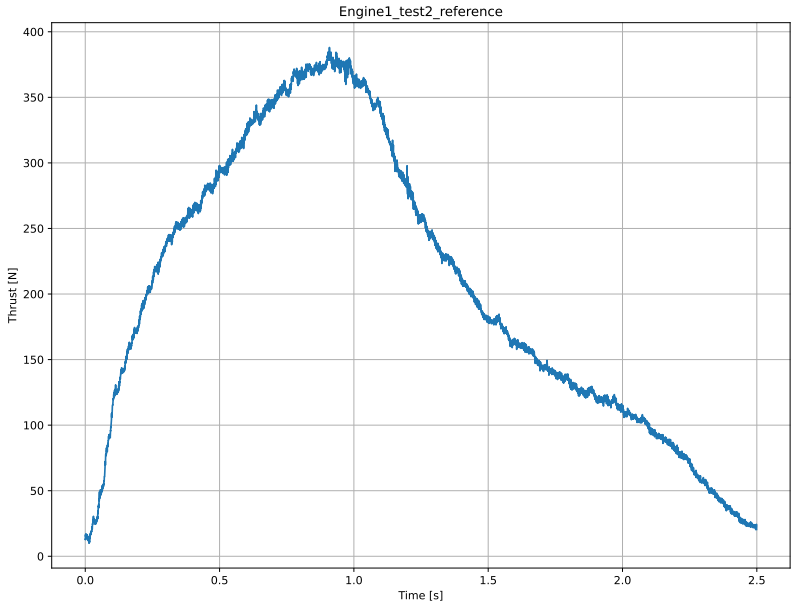


Figure H.6: Raw Thrust Data for Engine 1 Test 2

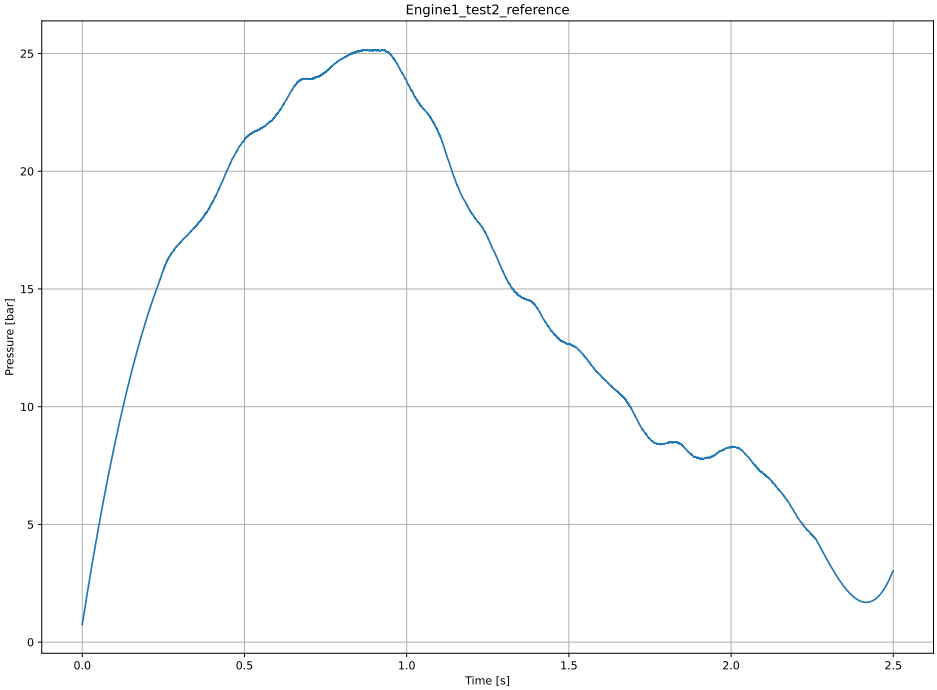


Figure H.7: Filtered Pressure Data for Engine 1 Test 2

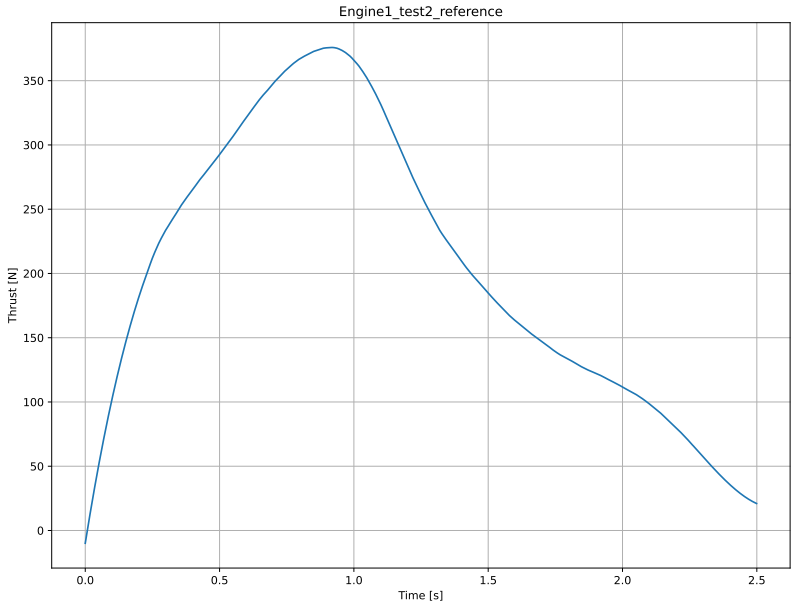


Figure H.8: Filtered Thrust Data for Engine 1 Test 2

H.3. Engine 2 Test 3

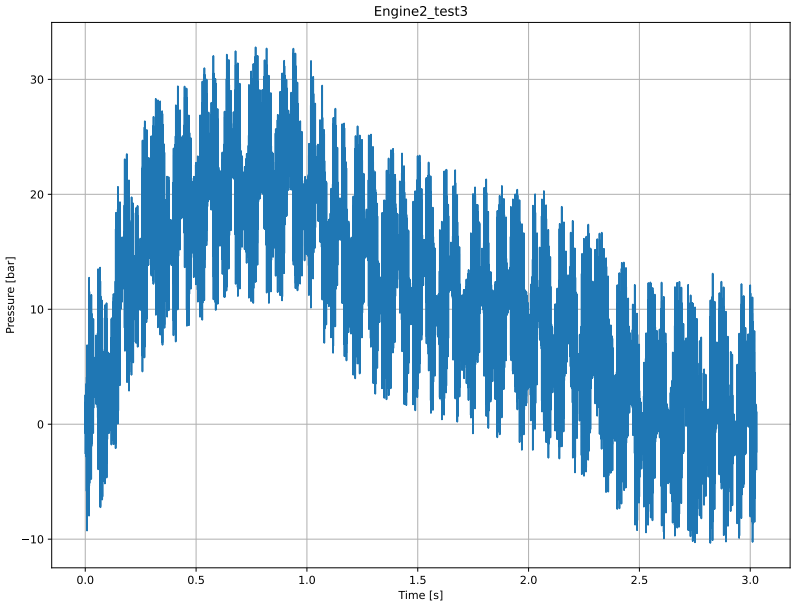


Figure H.9: Raw Pressure Data for Engine 2 Test 3

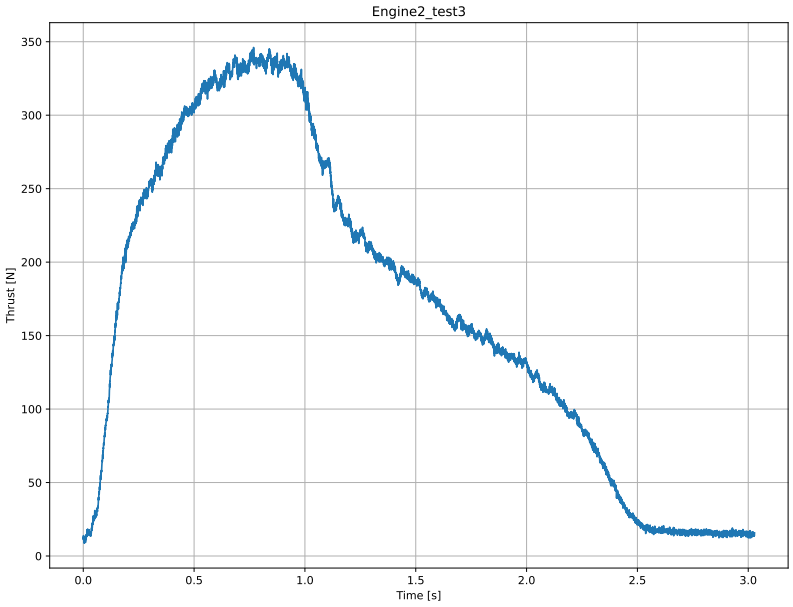


Figure H.10: Raw Thrust Data for Engine 2 Test 3

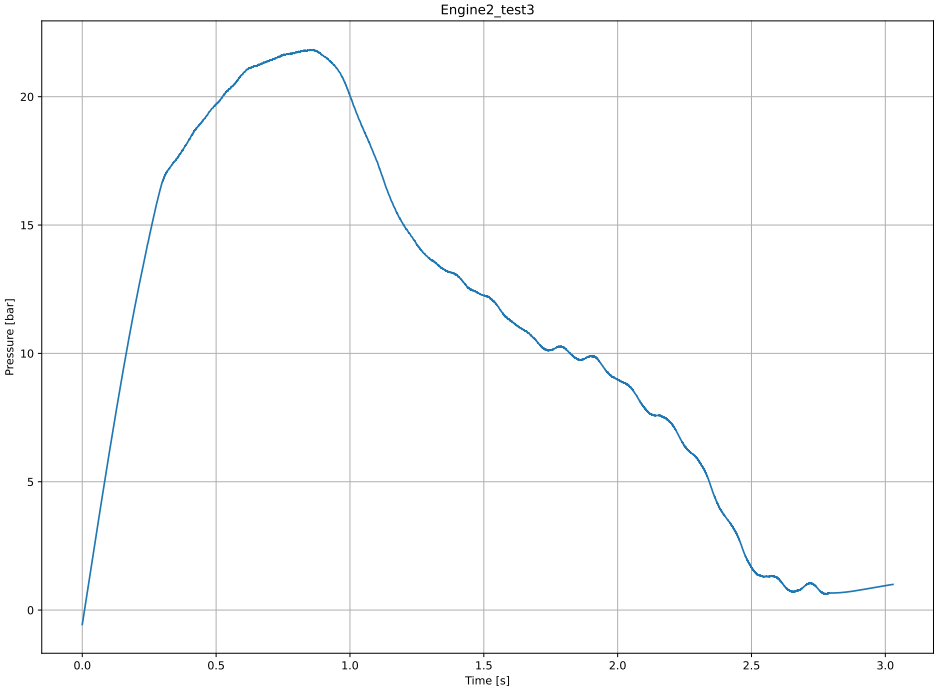


Figure H.11: Filtered Pressure Data for Engine 2 Test 3

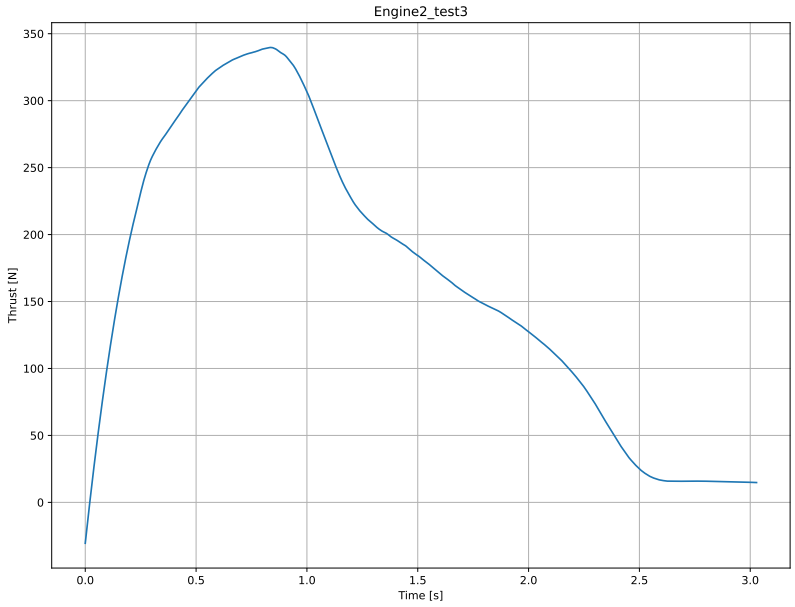


Figure H.12: Filtered Thrust Data for Engine 2 Test 3

H.4. Engine 2 Test 4

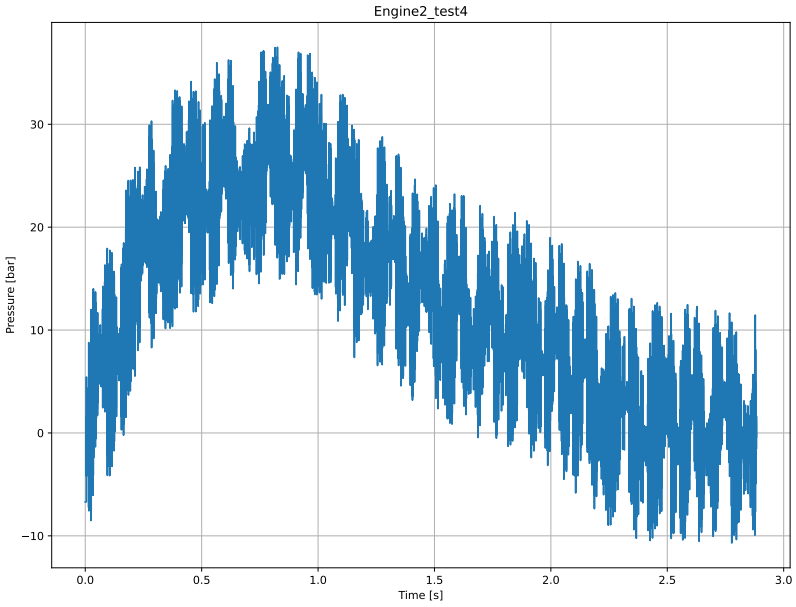


Figure H.13: Raw Pressure Data for Engine 2 Test 4

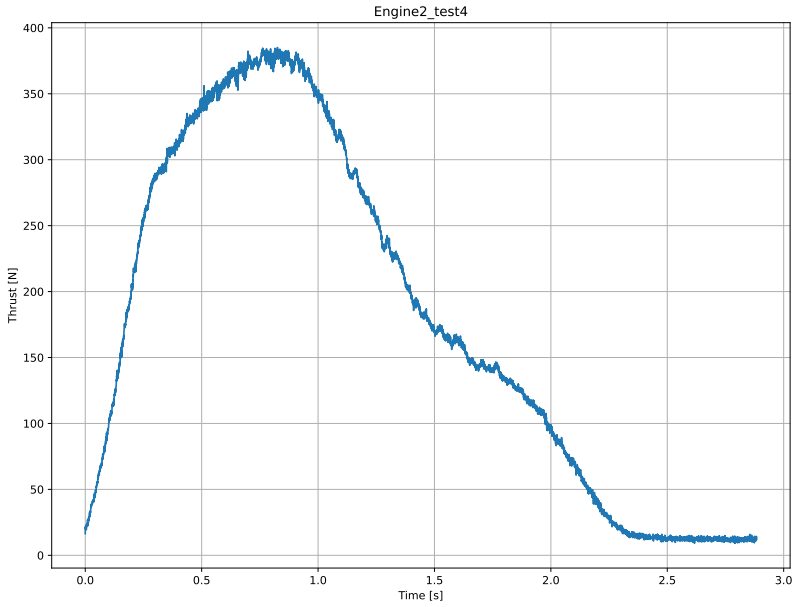


Figure H.14: Raw Thrust Data for Engine 2 Test 4

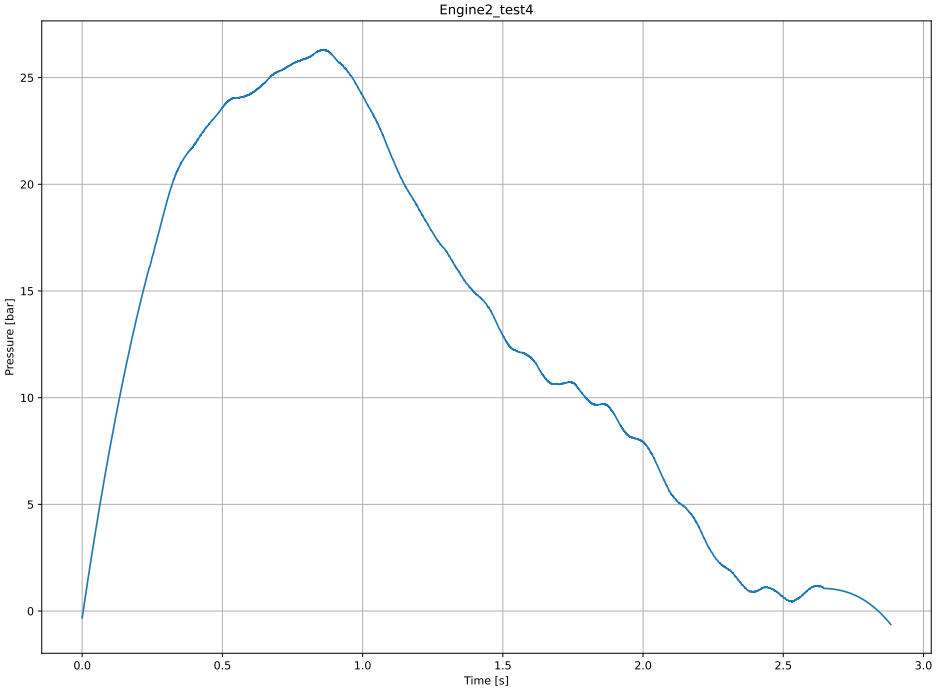


Figure H.15: Filtered Pressure Data for Engine 2 Test 4

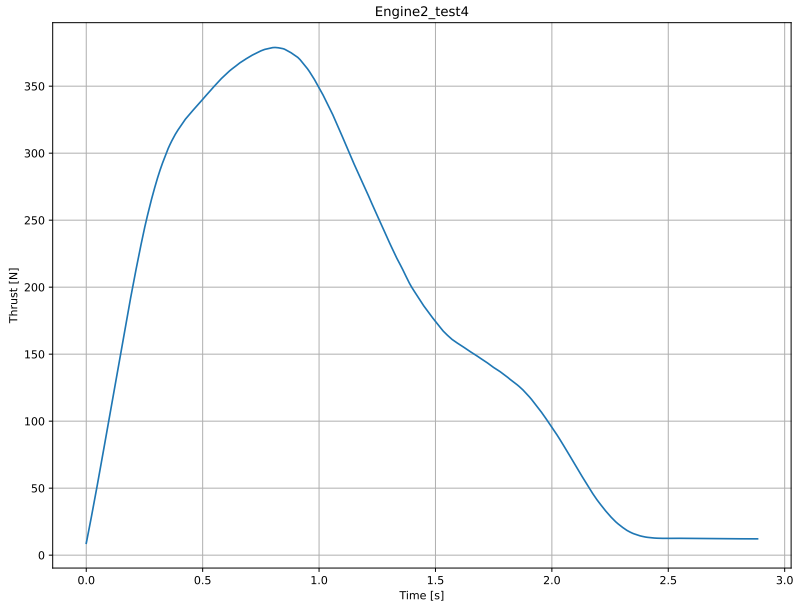


Figure H.16: Filtered Thrust Data for Engine 2 Test 4

H.5. Engine 3 Test 5

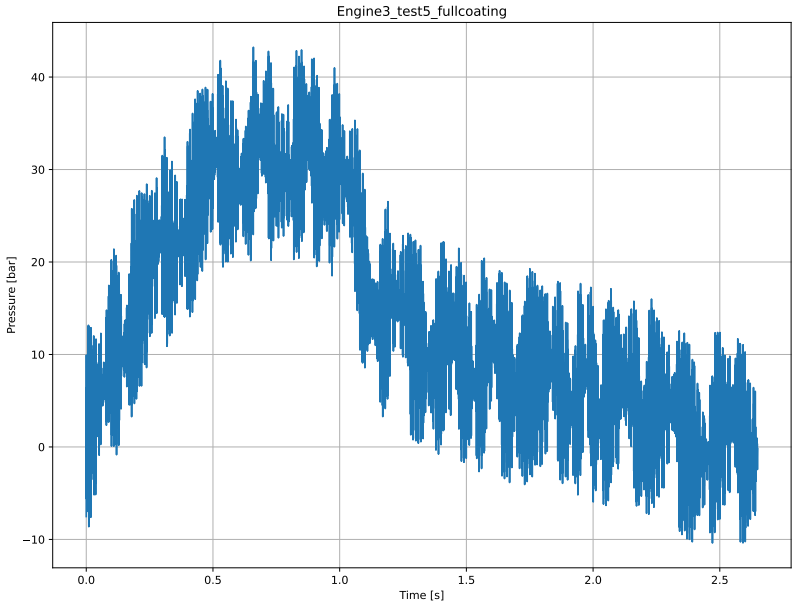


Figure H.17: Raw Pressure Data for Engine 3 Test 5

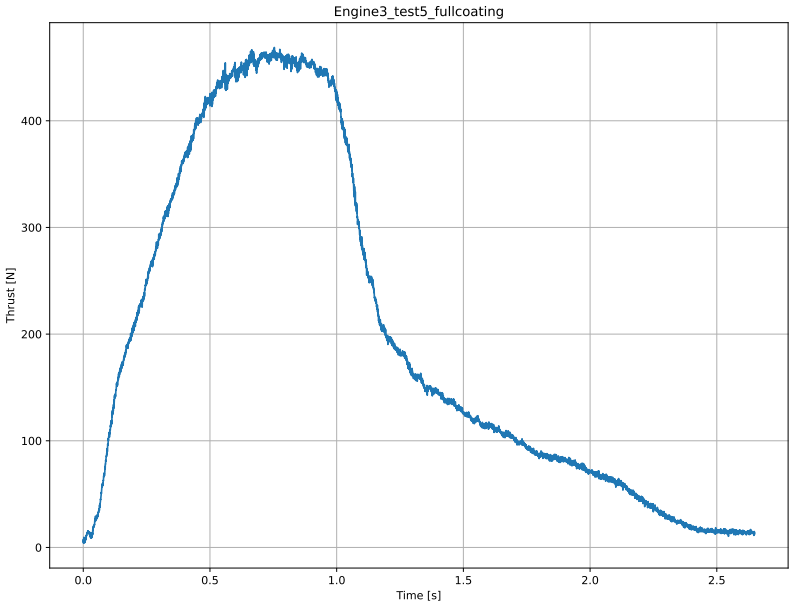


Figure H.18: Raw Thrust Data for Engine 3 Test 5

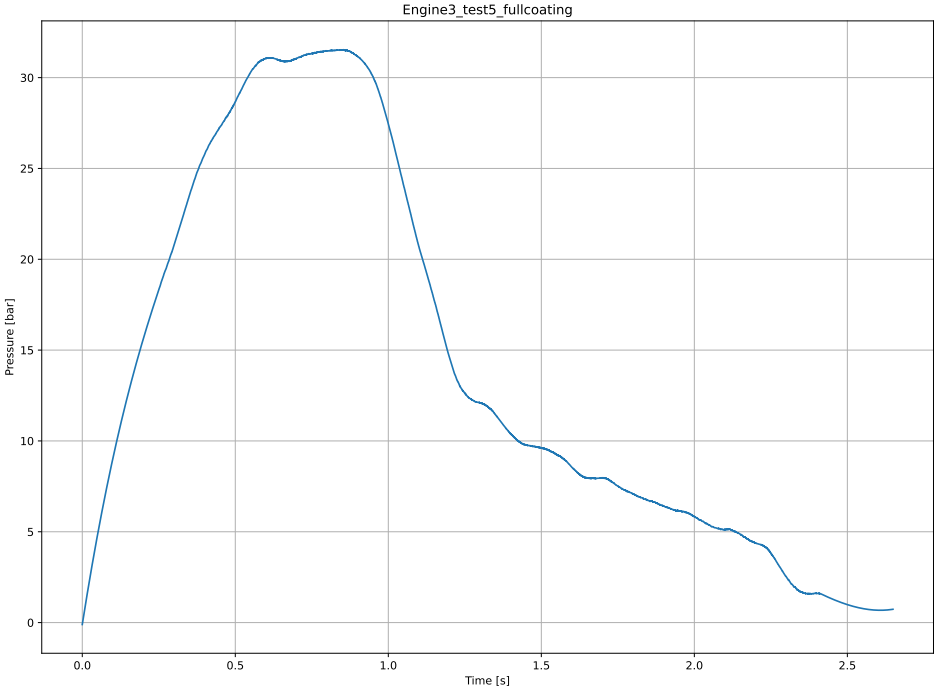


Figure H.19: Filtered Pressure Data for Engine 3 Test 5

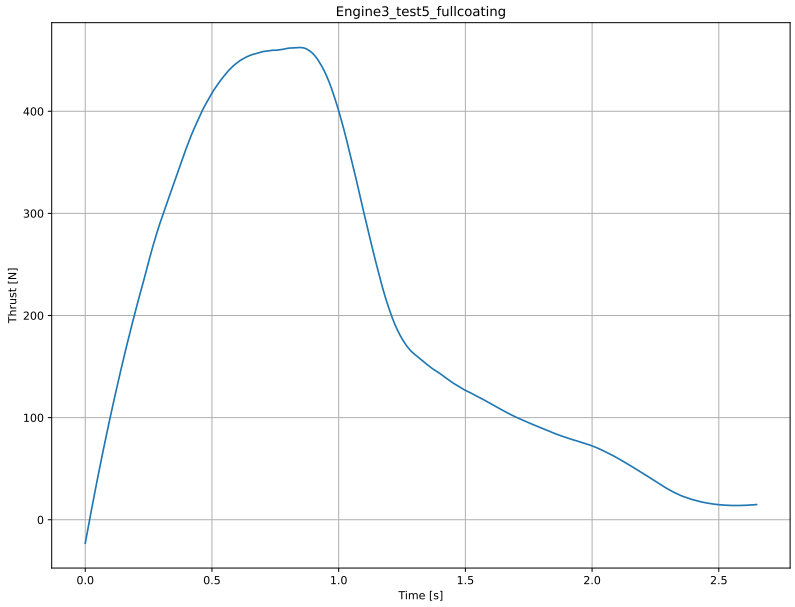


Figure H.20: Filtered Thrust Data for Engine 3 Test 5

H.6. Engine 3 Test 6

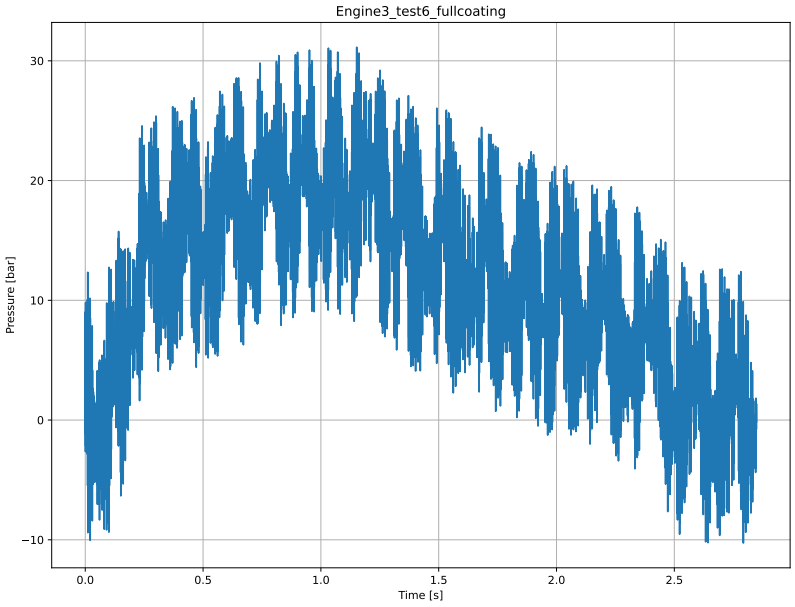


Figure H.21: Raw Pressure Data for Engine 3 Test 6

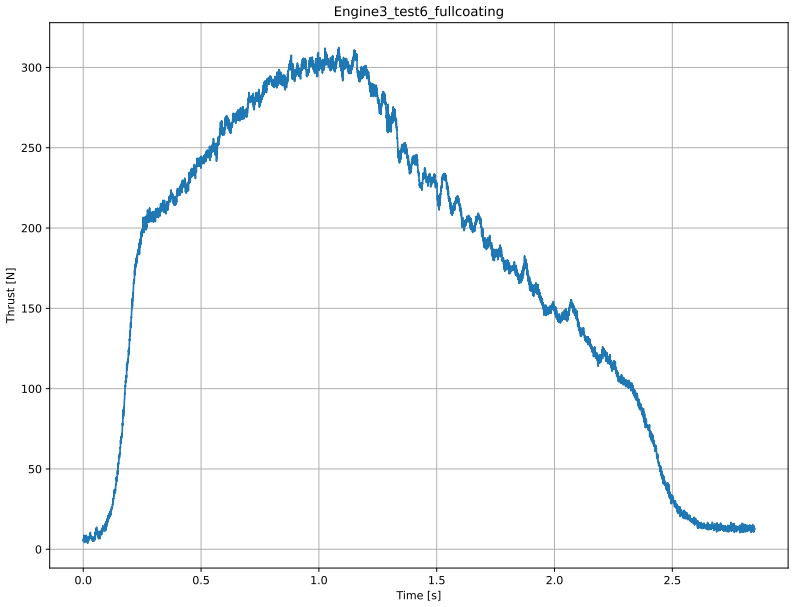


Figure H.22: Raw Thrust Data for Engine 3 Test 6

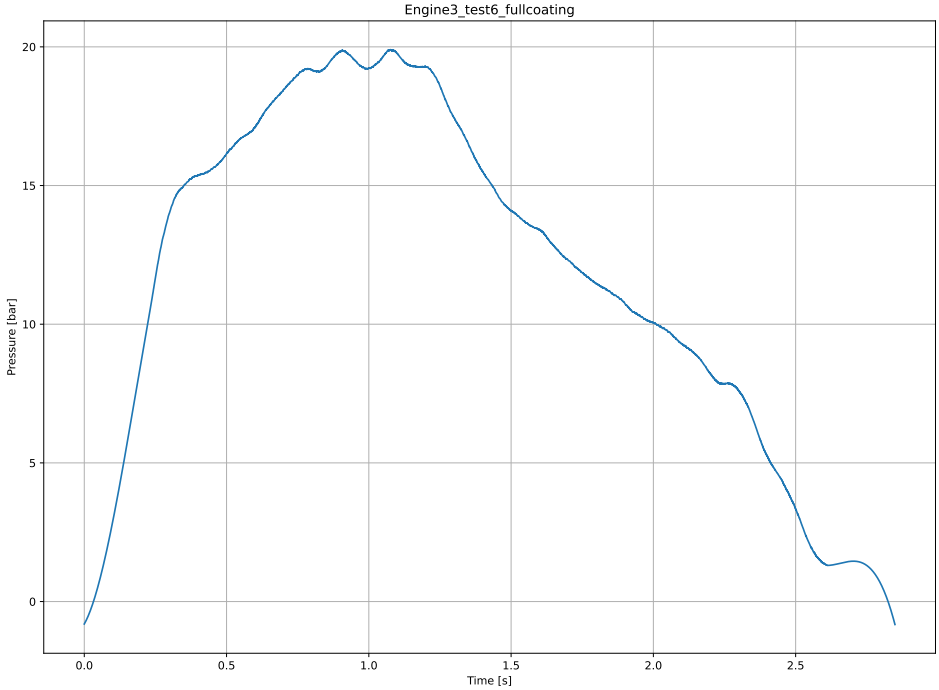


Figure H.23: Filtered Pressure Data for Engine 3 Test 6

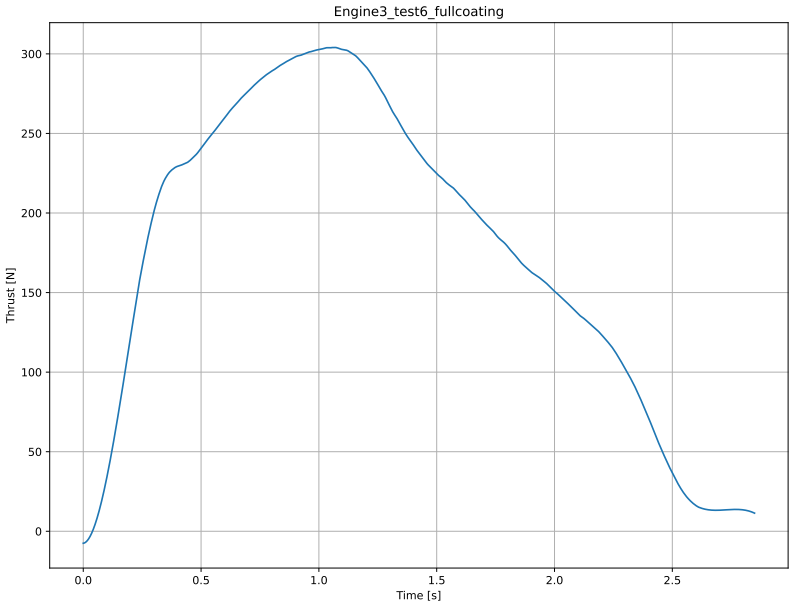


Figure H.24: Filtered Thrust Data for Engine 3 Test 6

H.7. Engine 3 Test 7

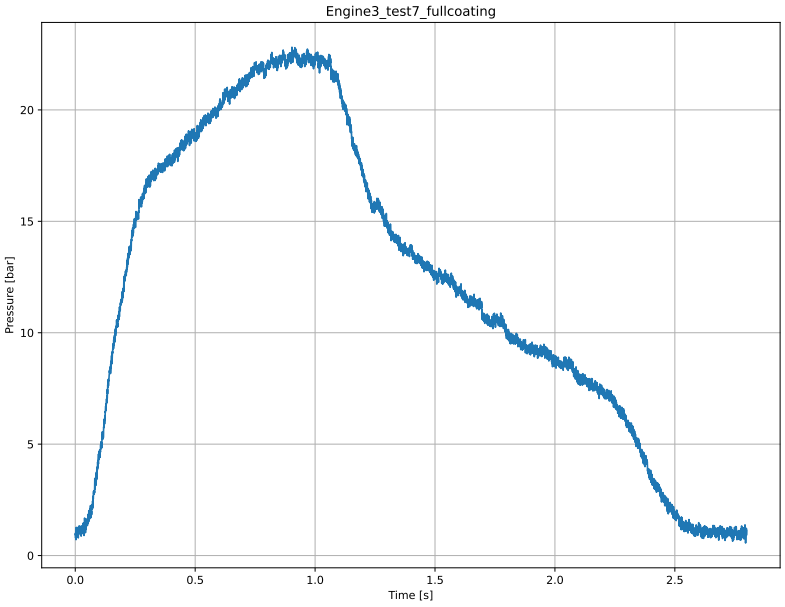


Figure H.25: Raw Pressure Data for Engine 3 Test 7

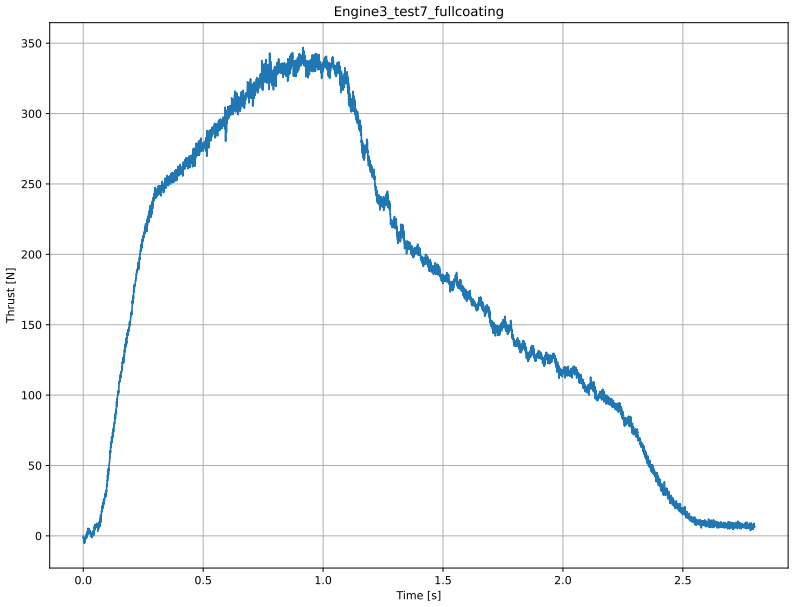


Figure H.26: Raw Thrust Data for Engine 3 Test 7

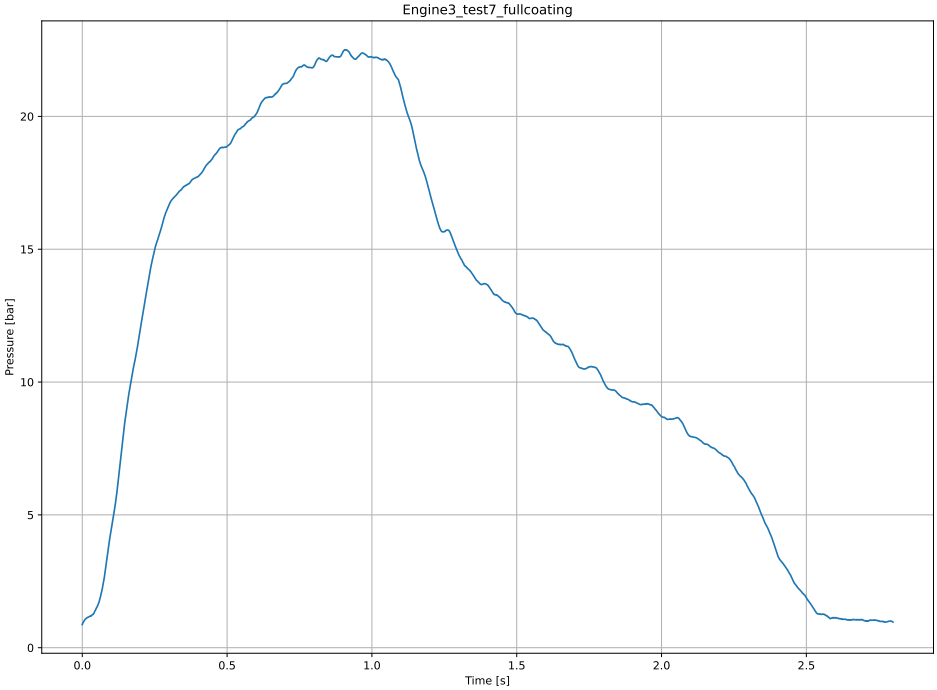


Figure H.27: Filtered Pressure Data for Engine 3 Test 7

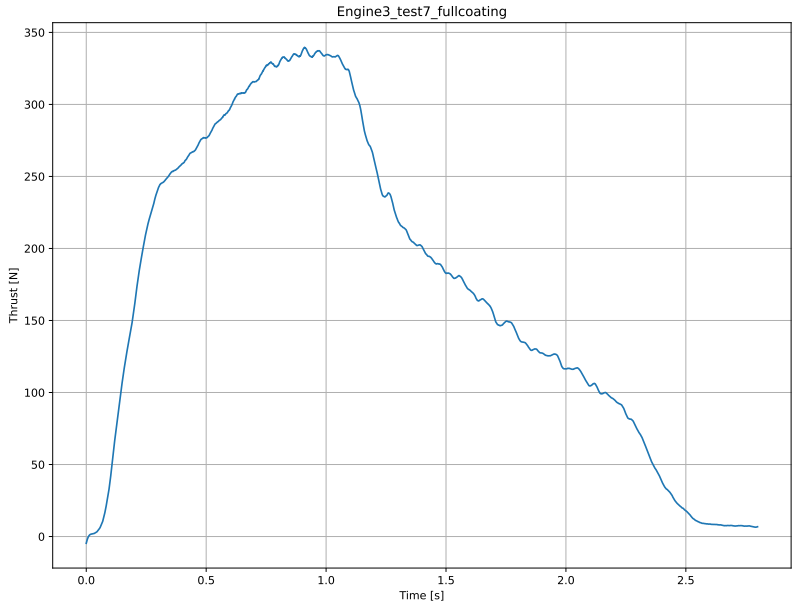


Figure H.28: Filtered Thrust Data for Engine 3 Test 7

H.8. Engine 3 Test 8

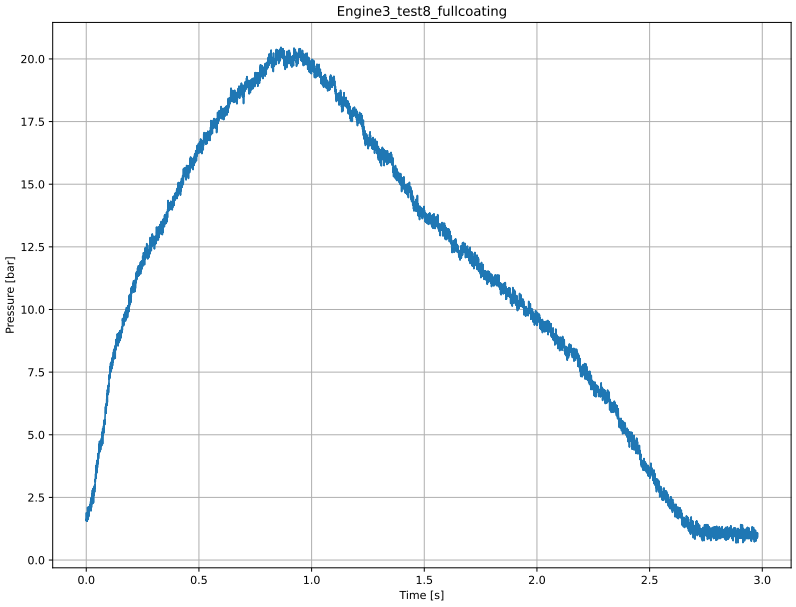


Figure H.29: Raw Pressure Data for Engine 3 Test 8

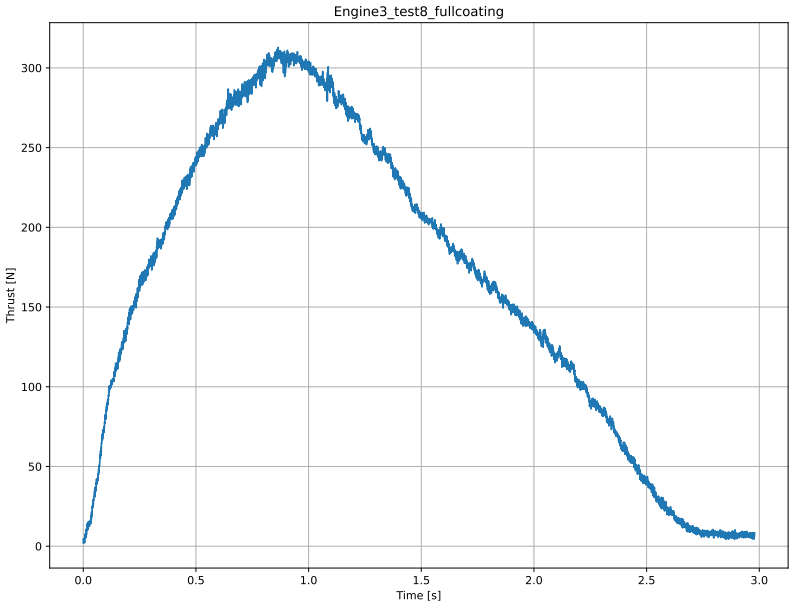


Figure H.30: Raw Thrust Data for Engine 3 Test 8

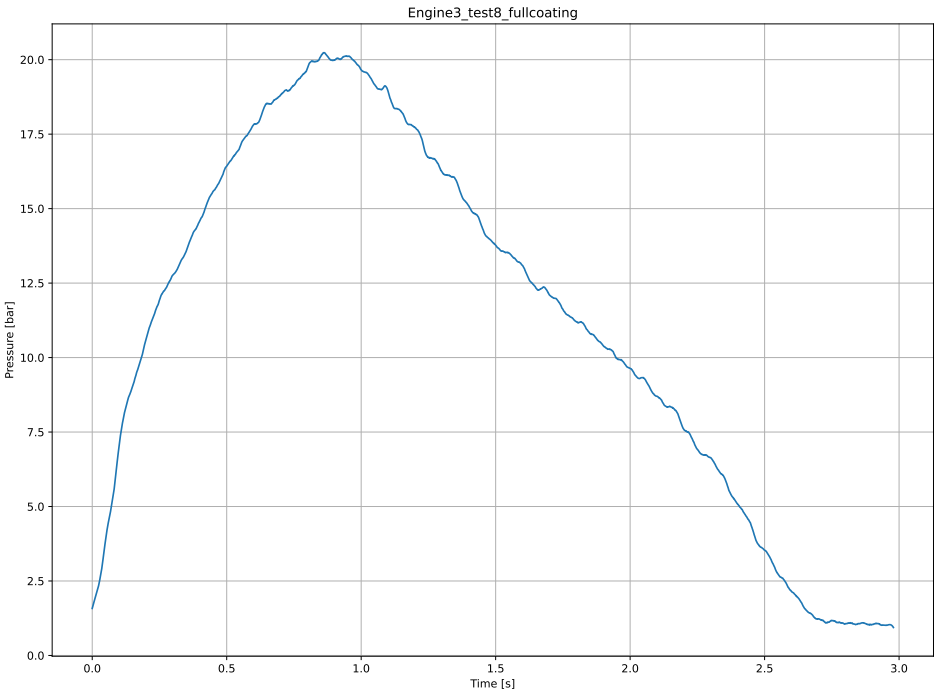


Figure H.31: Filtered Pressure Data for Engine 3 Test 8

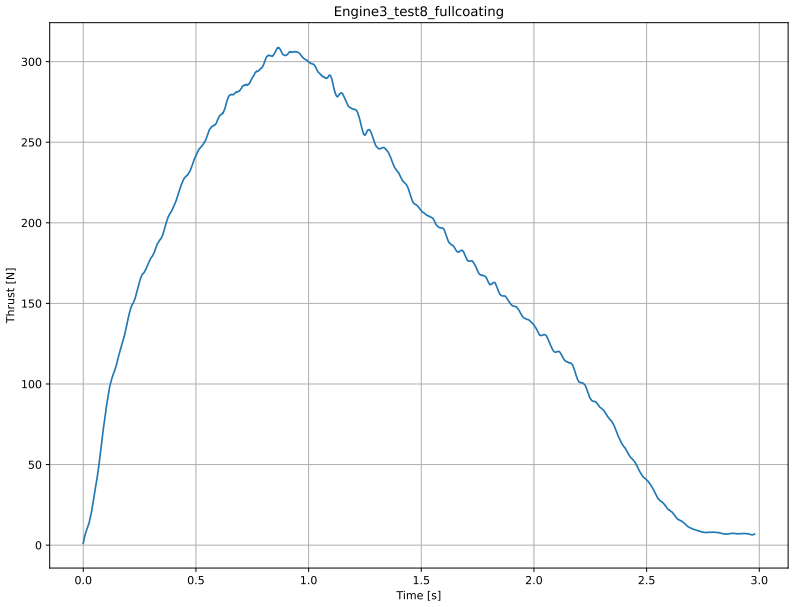


Figure H.32: Filtered Thrust Data for Engine 3 Test 8

H.9. Engine 4 Test 9

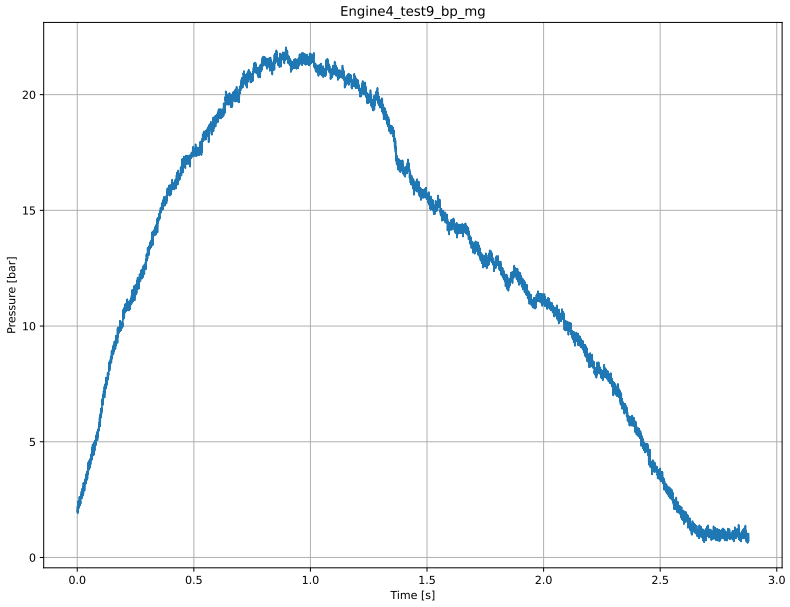


Figure H.33: Raw Pressure Data for Engine 4 Test 9

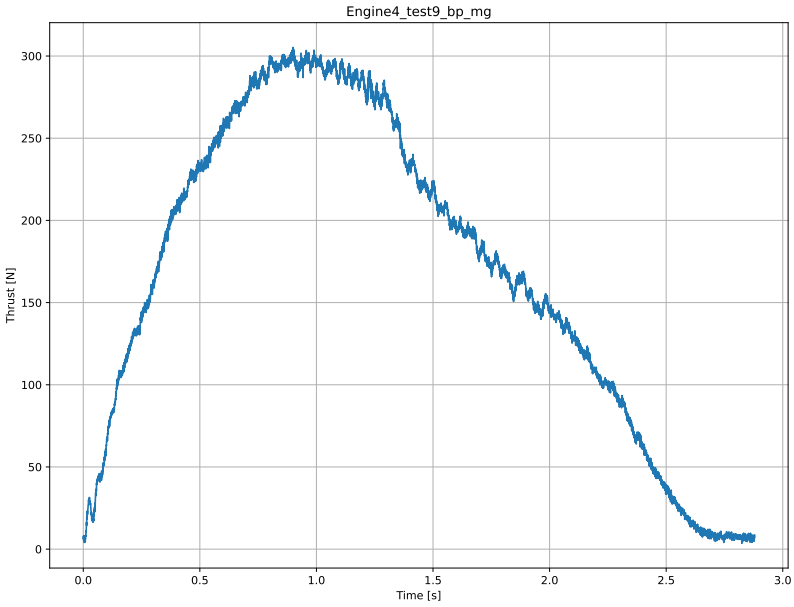


Figure H.34: Raw Thrust Data for Engine 4 Test 9

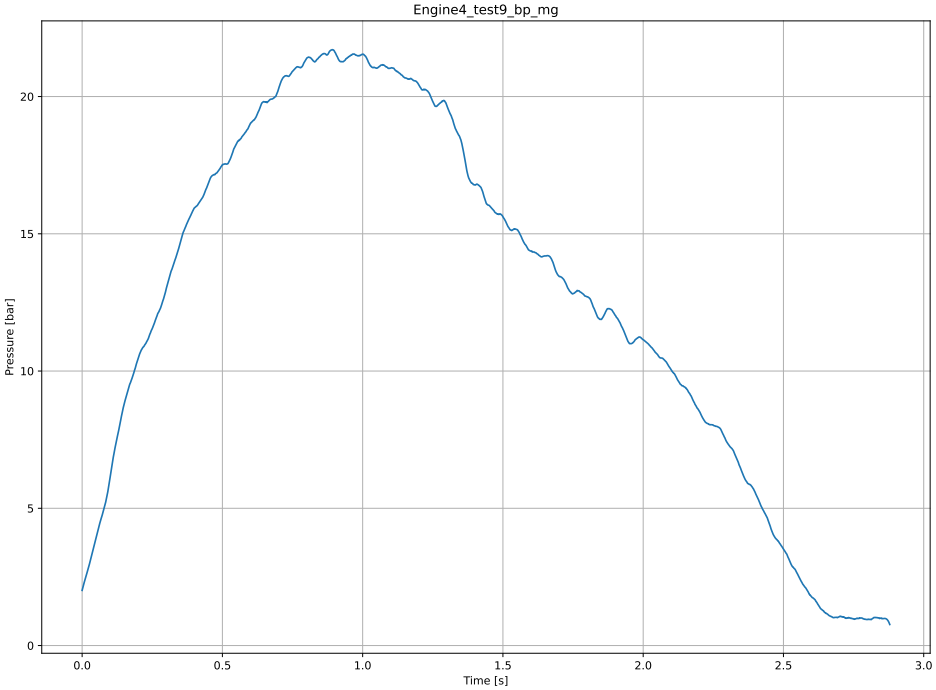


Figure H.35: Filtered Pressure Data for Engine 4 Test 9

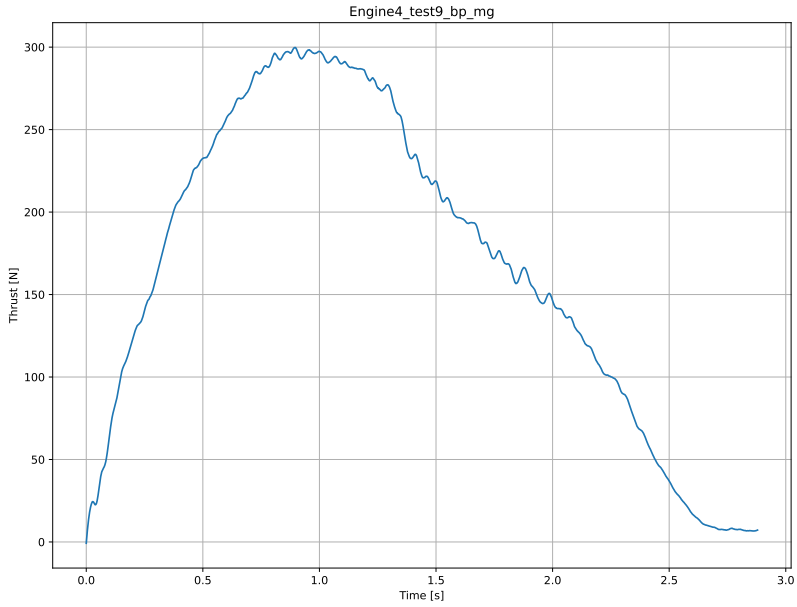


Figure H.36: Filtered Thrust Data for Engine 4 Test 9

H.10. Engine 4 Test 10

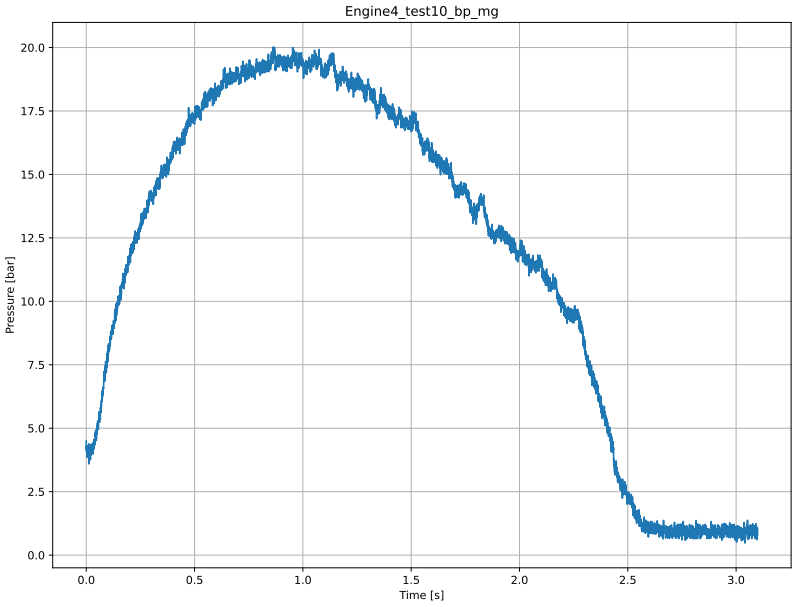


Figure H.37: Raw Pressure Data for Engine 4 Test 10

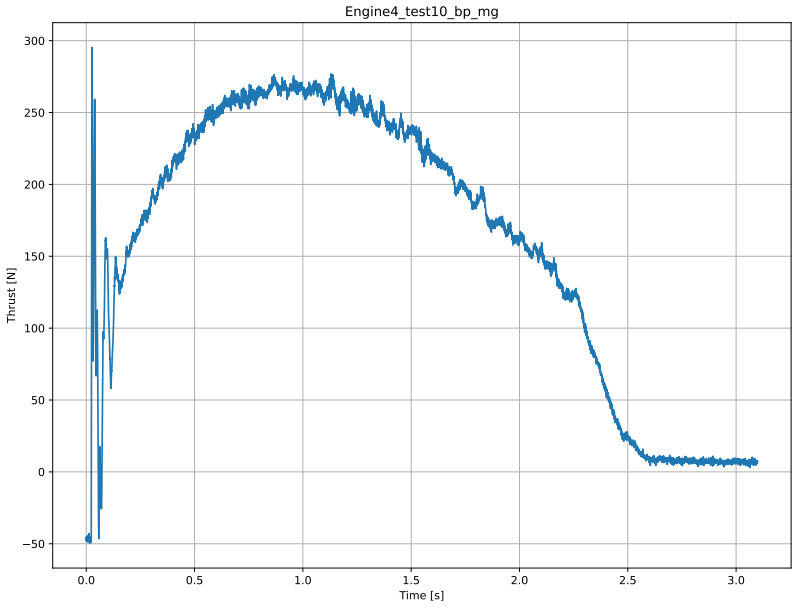


Figure H.38: Raw Thrust Data for Engine 4 Test 10

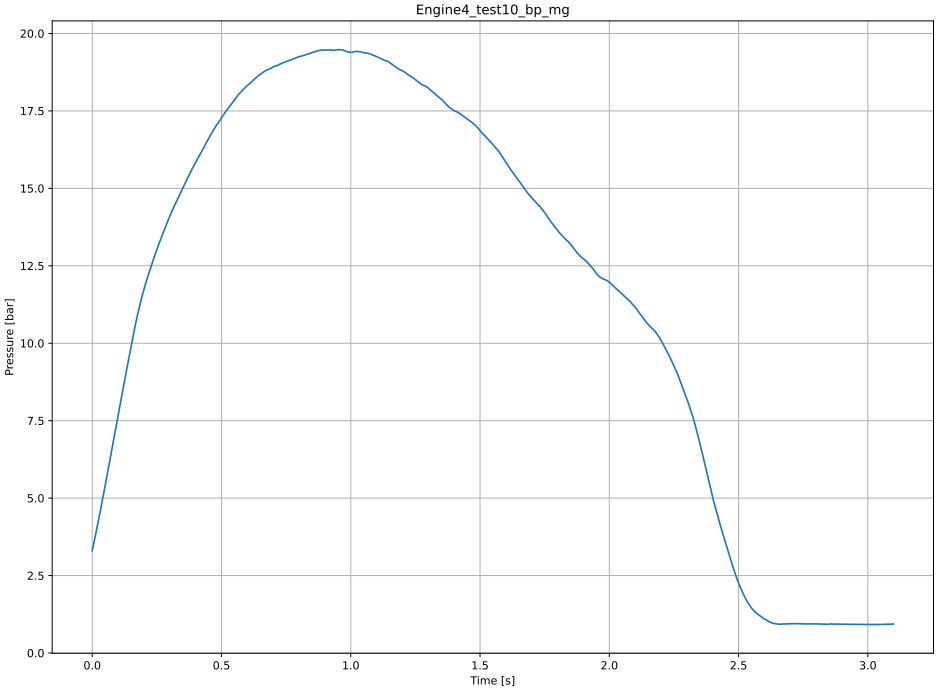


Figure H.39: Filtered Pressure Data for Engine 4 Test 10

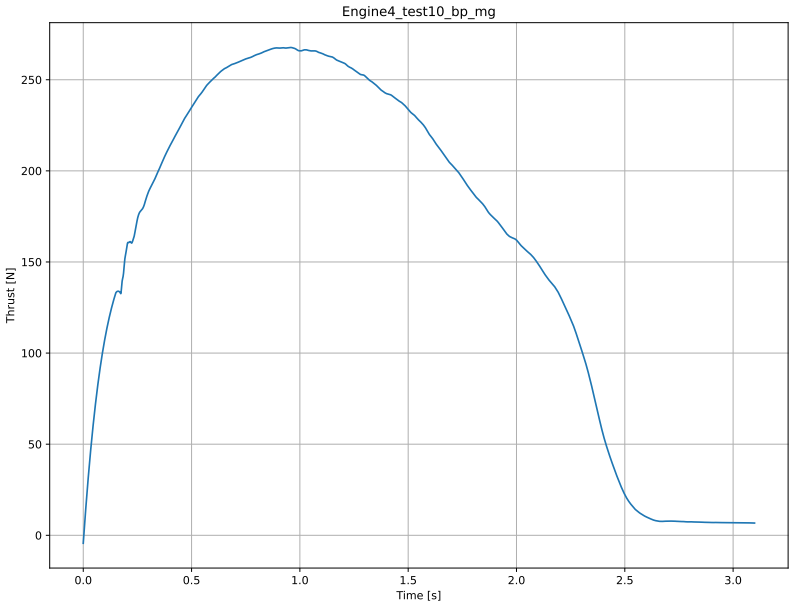


Figure H.40: Filtered Thrust Data for Engine 4 Test 10

H.11. Engine 5 Test 11

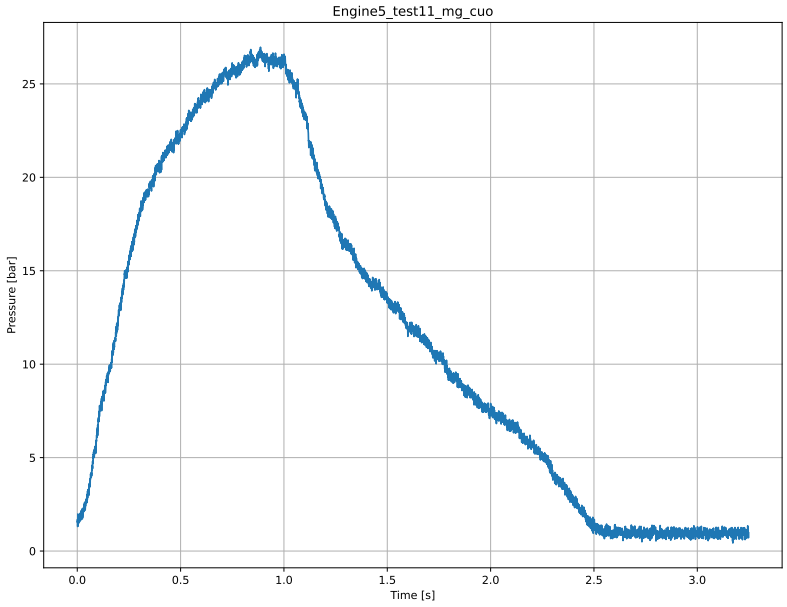


Figure H.41: Raw Pressure Data for Engine 5 Test 11

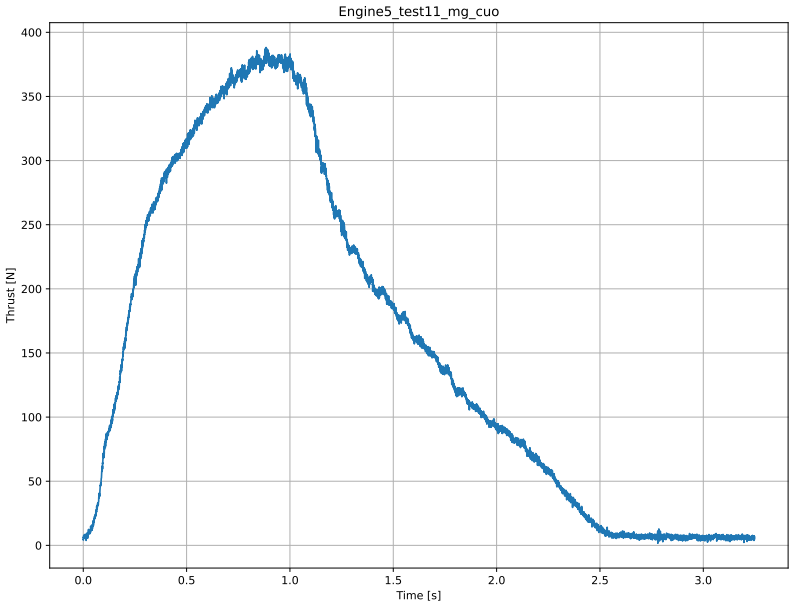


Figure H.42: Raw Thrust Data for Engine 5 Test 11

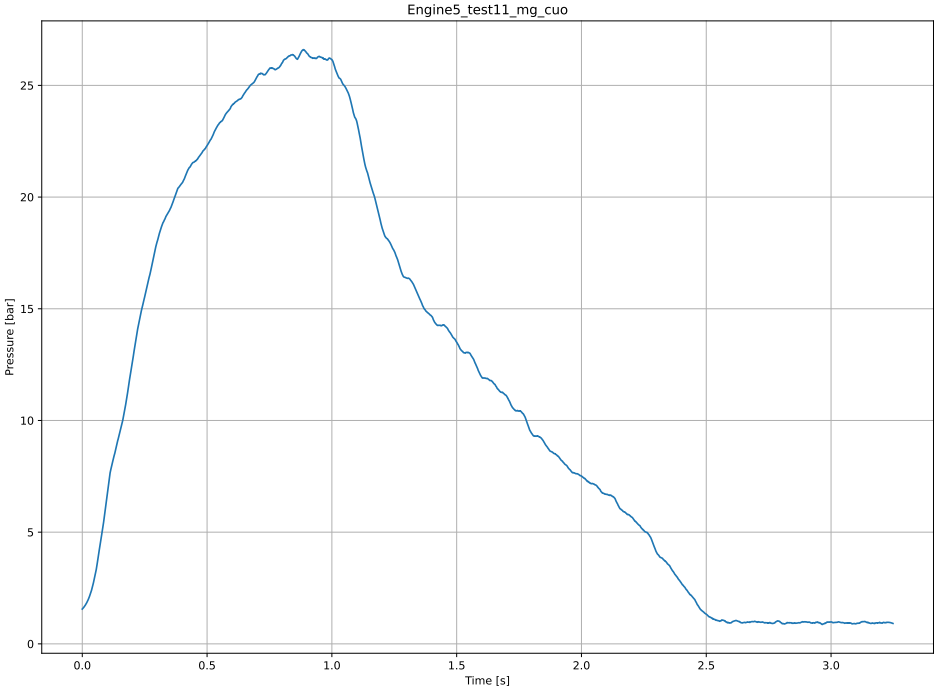


Figure H.43: Filtered Pressure Data for Engine 5 Test 11

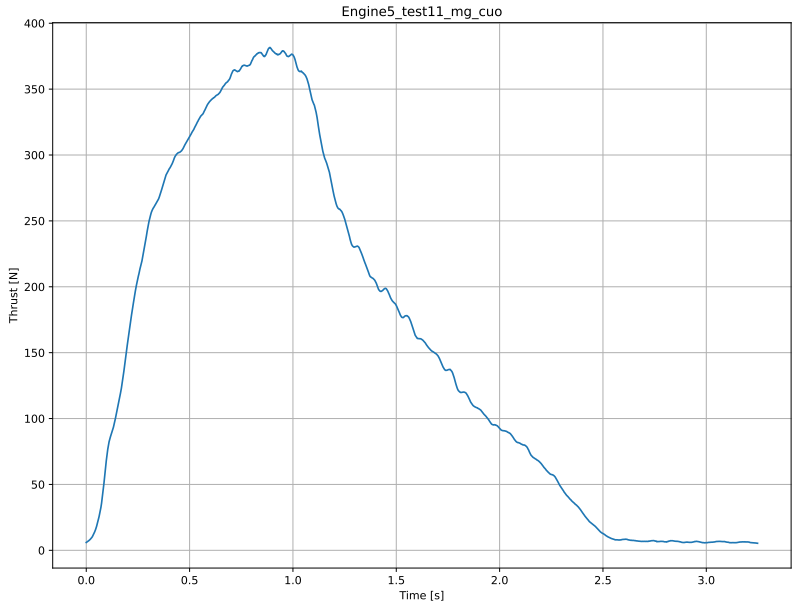


Figure H.44: Filtered Thrust Data for Engine 5 Test 11

H.12. Engine 5 Test 12

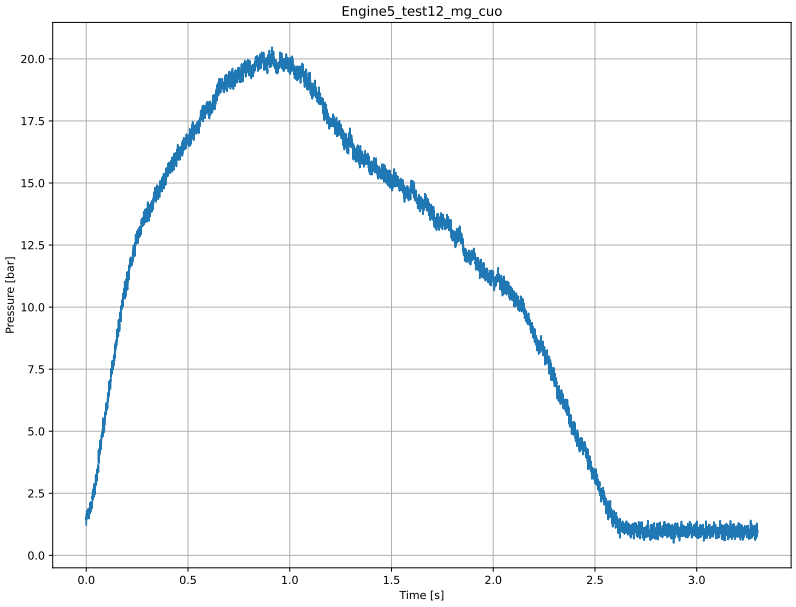


Figure H.45: Raw Pressure Data for Engine 5 Test 12

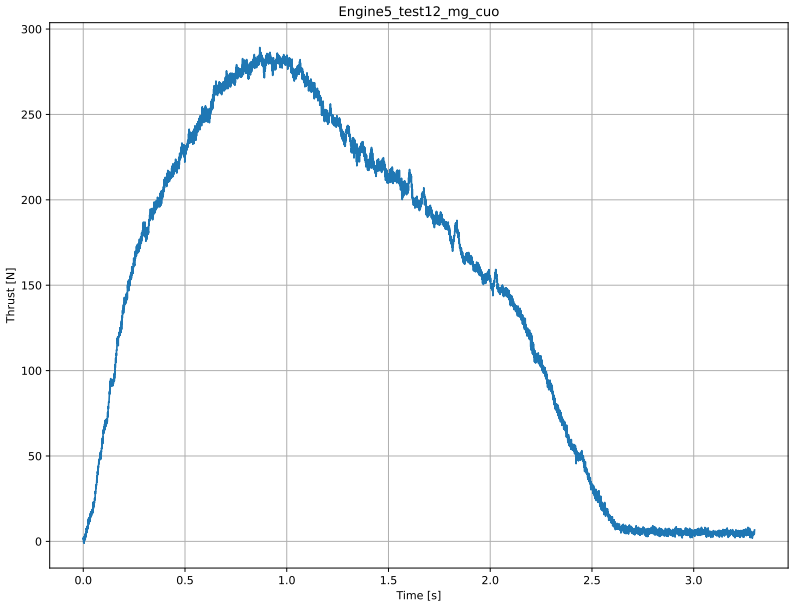


Figure H.46: Raw Thrust Data for Engine 5 Test 12

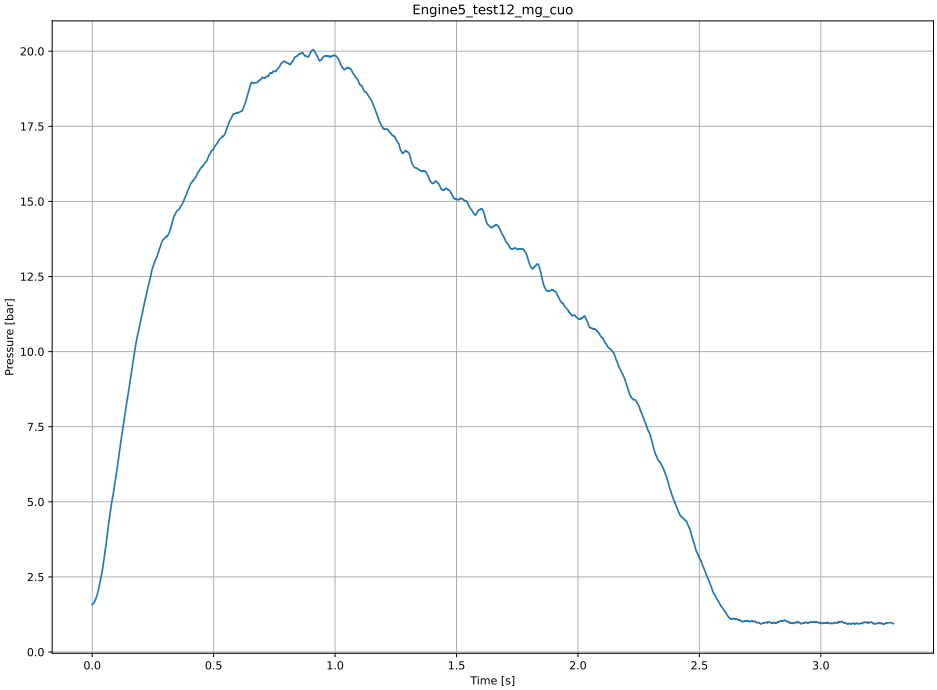


Figure H.47: Filtered Pressure Data for Engine 5 Test 12

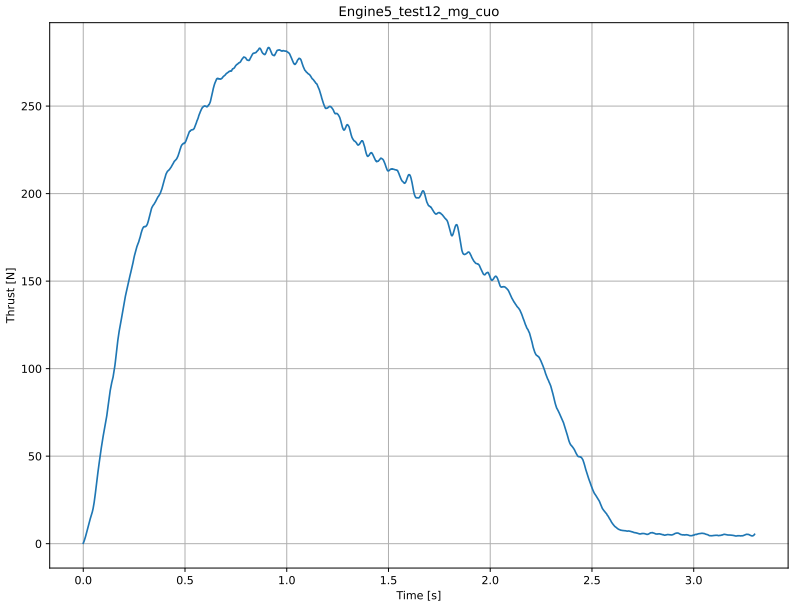


Figure H.48: Filtered Thrust Data for Engine 5 Test 12

I

Datasheet PT5402

In this appendix, the datasheet of the PT5402 pressure sensor is presented. The datasheet can be found in the following link [43]: PT5402 Datasheet

J

Datasheet LC500

In this appendix, the datasheet of the LC500 load cell is presented. The datasheet can be found in the following link [117]: LC500 Datasheet