Prifysgol Abertawe
Swansea University

UK Atomic Energy Authority

Computational Heat Transfer Optimisation in Nuclear Fusion Reactors Using Data-Driven Machine Learning

Daniela M. Segura Galeana

Submitted to the Faculty of Science and Engineering in fulfilment
of the requirements for the degree of

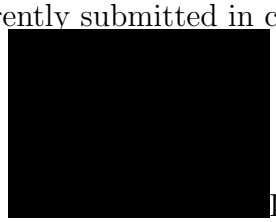
Doctor of Philosophy

Swansea University

2025

Declarations

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

 Daniela Minerva Segura Galeana

Signed: _____

10th August 2025

Date: _____

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

 Daniela Minerva Segura Galeana

Signed: _____

10th August 2025

Date: _____

I hereby give consent for my thesis, if accepted, to be available for electronic sharing

 Daniela Minerva Segura Galeana

Signed: _____

10th August 2025

Date: _____

The University's ethical procedures have been followed and, where appropriate, that ethical approval has been granted.

 Daniela Minerva Segura Galeana

Signed: _____

10th August 2025

Date: _____

Acknowledgements

I would like to express my deepest gratitude to Professor Antonio J. Gil and Professor Michael G. Edwards, for their unwavering support and encouragement through their supervision. Their guidance and criticism helped improve my work, their belief in me –even at times that I doubted myself– kept me going, and the science they taught me altogether have made of me a better scientist, and I feel honoured to have worked with them. I sincerely thank Doctor Aleksander Dubas, whose insight on Computational Fluid Dynamics and experience in High-Performance Computing contributed extensively to the development of Hammerhead. His mentorship in modelling was invaluable not only for the project, but for me personally as an avid enthusiast of CFD. To Doctor Andrew Davis and the UKAEA, without them this project would not have been possible. To the EPSRC DTP ADDED project, my full dedication to this thesis would not have been possible without the financial support of RCUK Energy Programme, grant number EP/T012250/1, and the ERDF via the Welsh Government.

The journey that has led me to here has given me confluence with fellow researchers and long lasting friends, which have become a second family. To Dani and Eugenio, the little piece of Mexico here in Swansea with me; to Nin, who showed me the courage to fight for what you believe in; to Sam and Thomas, who gave me a home when I was furthest from mine –and showed me that deadlines are no excuse when it comes to cooking good food–. To my CFD teacher Rafael Medina, thank you for sharing your own passion of fluid dynamics with me, without it I would not be here.

Finally, this work is dedicated to my family. To my mother Arcelia, who has believed in my success since the moment I was born and whose objections have always come from the best part of her heart. To Espe and Ico, my sister and brother, whose mockery of my dedication led me into learning how to relax and enjoy the moment. To my dad Fernando, you provided every precious opportunity to grow in any direction we wanted, without you I wouldn't be who I am now. To my boyfriend Alex, your consistent support, utter belief and constant supply of resources, knowledge and assistance have made possible not only this project, but so many others in life.

This thesis is the collective contribution of all of your support, without you it would not be the conception it is. Thank you all.

Abstract

In the field of energy applied to fluid dynamics, the net power output of a system is dependant upon the thermal and hydraulic behaviour of the flow in interaction with surrounding objects. For cooling systems applications, the interest relies on the understanding and optimisation of the Thermo-Hydraulic Performance (THP) research. The THP optimisation relevant to this research is that of shape parametrisation. It has been demonstrated [1] that utilising a biomimetic geometry on the fluid-facing surface of a pipe, based on the skin of fast sharks, improves hydraulic performance. However, the design of such geometry introduces a number of challenges that need to be addressed. First, the thermal and hydraulic mechanisms of the flow have a coupled nature, with a trade-off between them leading to a Multi-Objective Optimisation (MOO) problem. Second, the use of Computational Fluid Dynamics (CFD) to analyse the coupled flow behaviour. This is a cost-effective alternative to experimental testing and a favourable tool for geometry parametrisation. However, CFD implies a back-and-forth process between design and modelling over several iterations of shape parameters, which can make the development of optimal geometries highly time-consuming for the required level of accuracy. Therefore, industry demands for increasing levels of accuracy and the need for optimising the design process itself prompts the search for new means to undertake these tasks. In the ambition to address the above challenges within the computational design process, this thesis puts forward a comparative data-driven computational framework with an associated open-source software. The software generates both a database containing CFD high-fidelity models of parametrised geometries, as well as surrogate models that are trained with such data to approximate the THP optimisation process. The thesis puts forward three novelties. First, the study of the influence that simplified biomimetic geometries have in the coupled thermo-hydraulic behaviour of the flow in pipes. Second, a comparative analysis using alternative Machine Learning (ML) techniques in order to identify the advantages and drawbacks of each algorithm across different approaches. The considered ML methods are Radial Basis Function (RBF) [2] interpolation, Neural Networks (NN) [3] and Gaussian Processes (GP) [4], with the addition of Proper Orthogonal Decomposition (POD) as a dimensional reduction method prior to the models' training to potentially ease the computational expense [5]. Third, the framework's main direction is to use ML algorithms to enrich the search for an optimal

geometry within the considered parametric space. The efforts made to reach the objectives of this research have resulted into the open-source software Hammerhead integrated environment for high-fidelity and ML based computational design optimisation. This new framework is developed and implemented using Python3 language, including compatibility with OpenFOAM open-source CFD platform.

Research output

Journal publications

- **D.M. Segura Galeana**, A.J. Gil, M.G. Edwards, A. Dubas, A. Davis. Integrated environment for Machine Learning aided heat transfer optimisation in pipe systems. *Under preparation.*

- **D.M. Segura Galeana**, A.J. Gil, M.G. Edwards, A. Dubas, A. Davis. CFD based momentum and heat transfer optimisation study of shape parametrisation in pipe systems. *Under preparation.*

Conference presentations

- **D.M. Segura Galeana**, A.J. Gil, M.G. Edwards, A. Dubas, A. Davis. Towards computational heat transfer optimisation in nuclear fusion reactors using data-driven Machine Learning. *UK Association for Computational Mechanics*, Durham, United Kingdom, April 2024.

- **D.M. Segura Galeana**, A.J. Gil, M.G. Edwards, A. Dubas, A. Davis. Towards computational heat transfer optimisation in nuclear fusion reactors using data-driven Machine Learning. *SACEGME Postgraduate Research Student Seminars*, Swansea, United Kingdom, October 2023.

- **D.M. Segura Galeana**, A.J. Gil, M.G. Edwards, A. Dubas, A. Davis. Computational heat transfer optimisation in nuclear fusion reactors using physics enhanced Machine Learning. *UK Association for Computational Mechanics*, Warwick, United Kingdom, April 2023.

Research posters

- **D.M. Segura Galeana**, A.J. Gil, M.G. Edwards, A. Dubas, A. Davis. Towards computational heat transfer optimisation in a blanket's cooling system using data-driven

Machine Learning. *FuseNet PhD Event*, Lausanne, Switzerland, August 2023. **Best poster award.**

- **D.M. Segura Galeana**, A.J. Gil, M.G. Edwards, A. Dubas, A. Davis. Towards computational heat transfer optimisation in nuclear fusion reactors using data-driven Machine Learning. *SOFE*, Oxford, United Kingdom, July 2023.
- **D.M. Segura Galeana**, A.J. Gil, M.G. Edwards, A. Dubas, A. Davis. Towards computational heat transfer optimisation in nuclear fusion reactors using data-driven Machine Learning. *Zienkiewicz Institute for Modeling, Data and AI Postgraduate Research Conference*, Swansea, United Kingdom, May 2023.

Contents

List of Figures

List of Tables

Listings

Abbreviations

AI	Artificial Intelligence
AOF	Aggregated Objective Function
CFD	Computational Fluid Dymanics
CNN	Convolutional Neural Networks
DE	Differential evolution
DNS	Direct Numerical Simulations
FDM	Finite Difference Method
FVM	Finite Volume Method
GP	Gaussian Processes
GPT	Generative Pre-trained Transformer
Hammerhead	Harmonic MOO Model Expedient for the Reduction Hydraulic-loss and Heat-transfer Enhancement Asset Design
KLD	Karhunen-Loeve Decomposition
k-NN	k-Nearest Neighbors
LDA	Linear Discriminant Analysis
MCHS	Micro-Channel Heat Sink
ML	Machine Learning
MMS	Method of Manufactured Solutions
MOO	Multi-Objective Optimisation

MSE	Mean Square Error
MVN	Multi-Variate Normal
NN	Neural Networks
PCA	Principal Component Analysis
PDE	Partial Differential Equations
PIMPLE	Combination of PISO and SIMPLE algorithms
PISO	Pressure-Implicit with Splitting of Operators
POD	Proper Orthogonal Decomposition
RANS	Reynolds Averaged Navier-Stokes
RBF	Radial Basis Function
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks
SGD	Stochastic Gradient Descent
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
SVD	Singular Value Decomposition
SVM	Support-Vector Machines
THP	Thermo-Hydraulic Performance

Nomenclature

Fluid mechanics

$\dot{D}$	Dissipation rate in flow
E	Specific total energy
e	Specific internal energy
f_Ω	Volume source contribution flux
$\boldsymbol{f}_c$	Convection flux in fluid flow
$\boldsymbol{f}_d$	Diffusion flux in fluid flow
$\boldsymbol{f}_E$	Eulerian flux, total sum of fluxes
f_f	Dissipation rate
f_h	Advected heat flux
$\boldsymbol{f}_{\boldsymbol{n}}$	Surface source contribution flux
h	Specific enthalpy
$\boldsymbol{I}$	Identity tensor
k	Specific kinetic energy
k_d	Diffusivity coefficient, thermal conductivity
p	Flow pressure field
Pr	Prandtl number
Q	Arbitrary flow quantity
$\dot{Q}$	Net power output of flow

Fluid mechanics

$\dot{Q}_\Omega$	External heat source
Re	Reynolds number
S	Entropy
T	Temperature field
V	Specific volume of the fluid
$\dot{V}$	Volumetric flow rate
$\boldsymbol{v}$	Flow velocity field
α	Thermal expansion coefficient
Γ	Control surface
μ	Dynamic viscosity of the fluid
ρ	Fluid density within control volume
$\boldsymbol{\sigma}$	Stress tensor
$\boldsymbol{\tau}$	Deviatoric stress tensor
ν	Kinematic viscosity of the fluid
ϕ	Arbitrary flow quantity
Ω	Control volume
ς_p	Specific heat capacity at constant pressure
ς_v	Specific heat capacity at constant volume
∇	Differential operator

High fidelity modelling

$\mathrm{d_{EG}}$	Distance between E and G cell centres
E	Centre of current computed cell
EG	Vector relating E and G neighbouring cells
$\mathrm{f_E}$	Implicit contribution of volume source fluxes
$\mathrm{f_S}$	Non-linear contribution of fluxes in linearisation
$\mathrm{f_{SE}}$	Linearisation coefficient of E cell contribution in the face S
$\mathrm{f_{SG}}$	Linearisation coefficient of G cell contribution in the face S
G	Centre of neighbour of current computed cell

High fidelity modelling

$\boldsymbol{g}$	Volume sources
h_g	Grid spacing
l_{max}	Maximum length side of the cell
l_{min}	Minimum length side of the cell
M	Number of measurement locations of a given function
$\dot{m}_{\mathrm{S}}$	Mass flow rate
$\dot{\mathrm{m}}_{\mathrm{S}}$	Linearised mass flow rate
n_c	Complex eigenvalues
n_r	Real eigenvalues
n_s	Number of snapshots or sets of data of a given function
$\dot{q}_{\mathrm{E}}$	Heat source
r_h	Mesh refinement ratio
$\mathbf{S}_{\mathrm{E}}$	Surface normal vector of the current element
S_{E}	Bounding surface of current computed cell
S_{EG}	Surface separating E and G neighbouring cells
$\mathbf{S}_{\mathrm{n}}$	Normal component of vector relating E and G neighbouring cells
$\mathbf{S}_{\mathrm{t}}$	Tangengial component of vector relating E and G neighbouring cells
p_g	Order of accuracy of the numerical method
V_{E}	Volume of current computed cell
y^+	Dimensionless wall distance
λ	Eigenvalues of a matrix
ε	L^2-norm root mean square error

Surrogate modelling

a_i	Independent coefficient functions, definition for a given dependent function
$\mathbf{f}$	GP function prior, real process
$\mathbf{f}_*$	GP conditional distribution posterior test
$\mathbf{G}$	Right singular vectors of matrix $\boldsymbol{Y}$
$\mathbf{g}^l$	Activation functions for l layer in a NN

Surrogate modelling

$\mathbf{H}$	Left singular vectors of matrix $\mathbf{Y}$
$\hat{\mathbf{h}}$	Neural network's hypothesis for a given function
J	Least MSE based loss of a NN
$\mathbf{K}$	GP kernel function covariance prior or training set
$\mathbf{K}_*$	GP kernel function covariance training-test set
$\mathbf{K}_{**}$	GP kernel function covariance test set
L	Number of layers in a NN
$\mathrm{L_J}$	Marginal log likelihood function
N	Number of neurons per layer in a NN
$\mathbf{P}$	POD coefficients or modes
$\mathbf{P}_k$	Truncated POD coefficients or modes
p	Polynomials of degree $k-1$ in d unknowns
R	Number of features for the surrogate model input tensor
$\mathbf{W}$	Matrix of surrogate model hyperparameters
$\mathbf{w}$	Matrix of weight coefficients of radial basis functions
$\mathbf{X}$	Matrix of input features of a given function
$\boldsymbol{x}$	Vector of measurement locations of a given function
$\mathbf{x}$	Surrogate model tensor of input features
$\mathbf{Y}$	Matrix function to estimate through surrogate models
$\mathbf{y}$	Surrogate model target function, real process
μ	GP mean function prior
μ_*	GP mean function posterior test
$\mathbf{\Sigma}$	Diagonal matrix of singular values for matrix $\mathbf{Y}$
σ_n^2	GP variance parameter
φ_j	Radial basis functions
$\mathcal{N}$	Gaussian or normal distribution

Optimisation study

A_i	Amplitude of the i harmonic
A	Area of pipe entrance surface

Optimisation study

$\overline{f}_f$	Normalised dissipation rate (rough/baseline)
$\overline{f}_h$	Normalised advected heat flux (rough/baseline)
H	Number of harmonics in the equation
k_i	Wavenumber of the i harmonic
$\dot{Q}_0$	Baseline case net power output
$\dot{Q}_m$	Rough surface case net power output
$\overline{\dot{Q}}$	Normalised net power output (rough/baseline)
r	Pipe's radius
t	Pipe's cylinder thickness
$\overline{v}_x$	Flow average stream-wise velocity
$v_{x,max}$	Flow maximum stream-wise velocity, at the centre of the pipeline
Δy	$y-$direction element size

CHAPTER 1

Introduction

"There is nothing that defines who you are more than boundaries, whether you cross them or not, in every aspect of your life..."

— Guillermo del Toro, 1993

1.1 Motivation

The increasing energy demand, as well as the environmental pollution generated by the current energy production processes justify the need for a clean and virtually unlimited source of energy. The search for this source has led scientists to fusion energy. However, the task of generating energy from something as complex as plasma is not straightforward. Fusion has been successfully achieved on earth and yet it is not possible so far to reliably maintain it, much less harvest energy from it [6]. An effort at harnessing the energy from fusion is the experimental machine known as *tokamak*, a device that uses magnetic fields to confine the plasma from fusion in the shape of a torus [6–8]. This design operates like a conventional power plant, the plasma energy is absorbed as heat through the wall of the vessel, used to produce steam and then electricity by means of turbines and generators. This requires a set of supporting systems that work towards plasma handling, power supply and energy harvesting [6, 9]. Researchers are now approaching the challenge of optimising each subsystem to be able to use a fusion reactor as our energy production supply [6, 9–11]. This means involvement of areas like heat transfer and fluid dynamics to search for a practical way of producing power from the heat released by fusion nuclear energy [12–17].
To do so, a critical structure to observe is the cooling system of the reactor. Cooling systems are encountered in multiple engineering devices, from laptops to aircraft engines and thermal power plants, working to maintain the device from exceeding temperature limits or to *extract energy from heat sources*. The challenge of harvesting energy this way is to simultaneously take advantage of the thermodynamic and hydraulic efficiencies of the pipe system [18–21], two fields that interact with each other and are analogous to the recoverable and dissipative energies in the flow, respectively. Generally, the trade-off between the two mechanisms means that maximising the recoverable power also increases the pressure loss, and conversely [18, 21, 22]. This results in a Multi-Objective Optimisation (MOO) problem with non-unique solutions. It is understood then that the Thermo-Hydraulic Performance (THP) of flow in a pipe system is crucial to optimise the power output of a fusion reactor. It has been reported [1, 21–32] that surface roughness has an impact on the behaviour of interest, and when the parameters of such roughness are designed correctly, this impact is towards increasing overall efficiency.
The application and comparison of different surface roughness geometries is considered here a shape parametrisation problem. Addressing the design of such geometries is a challenge that requires the use of high-fidelity Computational Fluid Dynamics (CFD) modelling. However, the expense of these simulations increases with a large parametric space, without the certainty of finding the geometry for the optimal THP. In addition, the onset of Machine Learning (ML) as a leading technique in science to emulate the behaviour of physical systems calls for an integration of state-of-the-art algorithms with high-fidelity modelling. In the interest of researching the open-ended optimisation of the coupled thermo-hydraulic behaviour of the flow through the exploratory design of simplified parametric geometries, a new merged modelling framework for real-time optimisation is developed in this study. The chart presented in Figure 1.1 outlines the research breakdown. This serves as a reference to situate at any time the individual topics of this project, their relation to each other and their contribution to the overall framework.

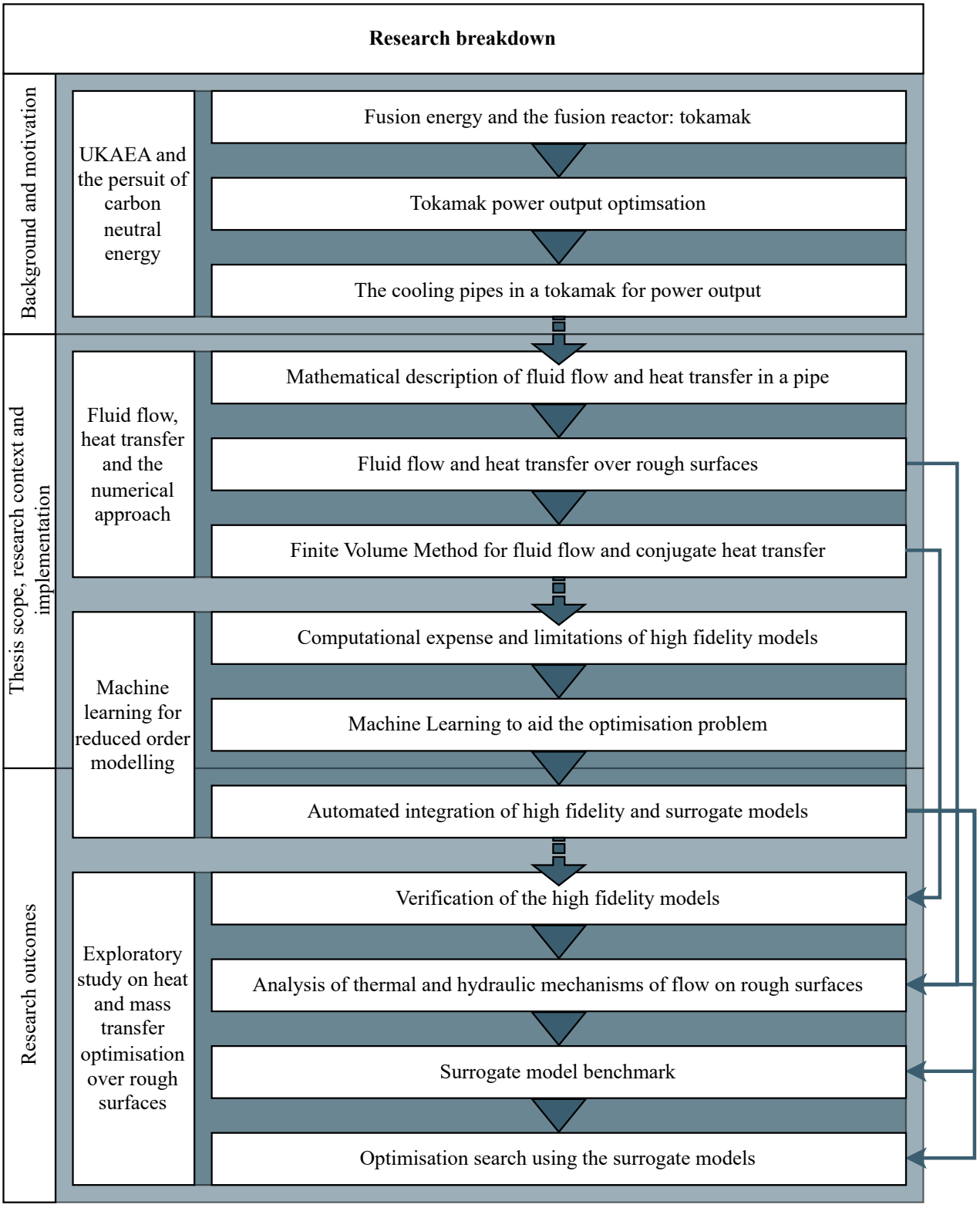


Figure 1.1: Research breakdown, content interrelation and role in the general context.

1.2 Fusion energy

The world-wide exploration of obtaining power from nuclear fusion began in the late 1940s. Yet, the story starts a century earlier with the problem of understanding the Sun. The discovery of energy conservation laws meant that the virtually unlimited and continuous supply of heat and light provided by this star had to come from an energy source. In the 1920s, Sir Arthur Eddington postulated our Sun and similar stellar objects as spheres of hot gas mainly composed of hydrogen. For the gas pressure to balance the inward gravitational forces within these stars, the temperature grows to around 15 million Kelvin. Under these conditions hydrogen nuclei fuse together to form a helium nucleus and release energy. The dominant process starts from hydrogen nuclei, where two protons fuse, releasing a positron and forming a deuteron. Once the deuteron is formed it lasts just a few seconds before another proton is added to form the light isotope of helium He^3. After an average of a million years, two He^3 nuclei fuse to form He^4, releasing protons and an amount of energy of $6e^8$ MJ per kilogram of hydrogen [8]. These timescales, however, are not practical. An approach to reduce the timescales is using a nucleus of deuterium to fuse with a nucleus of tritium –D-T reaction–, which produce the He^4 isotope as an α-particle[1] and release a neutron [7, 8]. In order to induce D-T reactions, it is necessary to overcome the mutual repulsion due to their positive charges, and as a result the cross-section[2] only becomes large enough at high energies. Thus, the D-T mixture is required to reach a sufficiently high temperature so the thermal velocities of the nuclei are high enough to react. This process is called *thermonuclear fusion*. Around 100 million Kelvin are necessary to support the process, at which the mixture called *plasma* is ionized. However, the extreme temperatures make confinement impossible by material walls, thus magnetic confinement is used in *tokamak reactors*.

The tokamak consists of poloidal-toroidal magnetic fields confining the plasma particles to a toroidal region [7, 8]. The magnetic fields have an associated flux change produced by currents in coils linking the torus. In present experiments, the toroidal currents are carried by a central solenoid as illustrated in Figure 1.2, and suitably placed poloidal/toroidal coils to control the position of plasma. The immediate physical barrier directly facing the hot plasma is the blanket system, as illustrated in Figure 1.2, covering the walls of the *vacuum vessel* [6, 10]. The blanket is composed of *modules*, as shown in Figure 1.3 that protect the rest of the reactor –including the cryogenic superconducting coils needed produce the strong magnetic fields required to initiate, confine, shape and control the plasma– by absorbing most of the radiative and particle heat fluxes from the plasma, as well as stopping or slowing down most of the neutrons that result from the fusion reactions. A primary objective of future experimental efforts is to use the released neutrons to breed tritium from lithium, an element with larger natural abundance compared to tritium. The design of the blanket modules is still under development, further information on this process can be found in references [10–17, 33–38]. Yet, the shielding and breeding roles are not the only intended by this system. The power output of the tokamak is provided by the cooling system of the blanket, presented in Figure 1.3. This apparatus employs a suitable coolant in forced convection conditions to carry away the heat produced by the energy of released fluxes and neutrons [10]. Therefore, from this stage of the process, obtaining power

[1] Particle composed of two protons and neutrons tightly bonded together

[2] The probability of a process to take place in particle collisions.

from nuclear fusion is similar to other thermal power generation processes.

1.3 The optimisation problem

The challenge for the blanket is to safely remove the heat as power output without exceeding material design limits [9]. To efficiently do so, the cooling system is required to maximise the heat extraction without losing much fluid flow energy. These two objectives are jointly

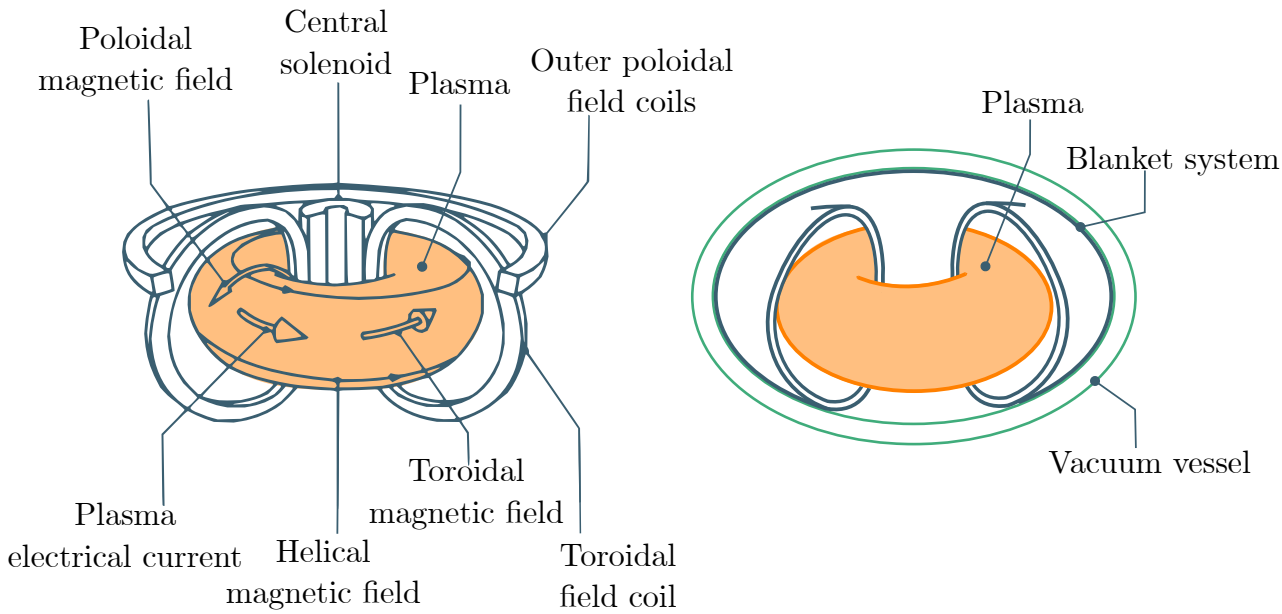


Figure 1.2: –Left– Arrangement of coils and currents in a tokamak, –right– simplified diagram of the tokamak's vacuum vessel and blanket system.

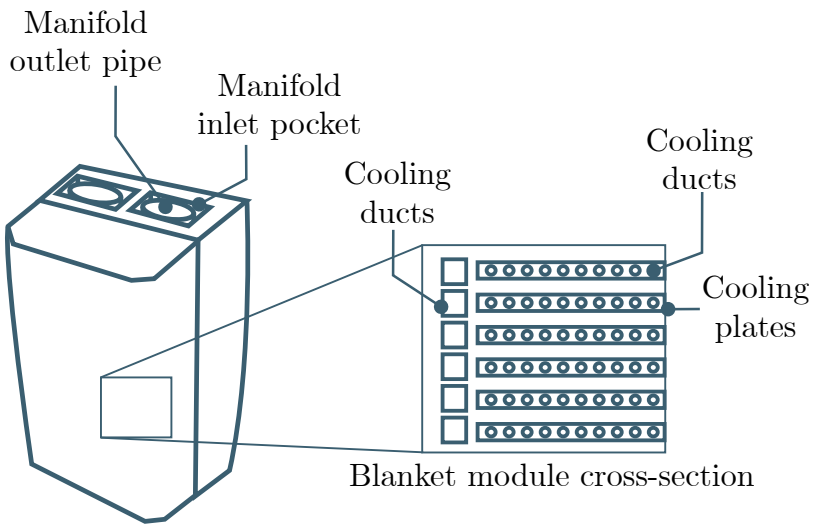


Figure 1.3: Simplified design of a blanket module with particular emphasis in the cooling system and pipes.

assessed as Thermo-Hydraulic Performance (THP), which can be optimised through different methods. The approach of interest within this project consists of varying the wetted surface geometries, introducing roughness on the inner surface of the pipe. This can be performed through ***topology optimisation***, which is a method that seeks the best layout of solid/fluid, independently of a previously assumed shape or configuration. The method is highly useful for understanding the conjugate behaviour of heat transfer where there are no manufacturing limitations in place. This provides great THP designs but with the likelihood of increasing the cost of production. In addition, most of these studies are done through adjoint algorithms and sensitivity analysis for the fluid dynamic and heat transfer optimisation [18, 19, 21, 39, 40].
The ***shape optimisation*** technique, on the other hand, assumes design parameters as boundaries to be modified by the optimisation process. This allows to build in limits and considers a lower amount of variables for the Multi-Objective Optimisation (MOO) problem. Efficient heat transfer without excessive pressure loss has been studied through the years by using pre-established biomimetic micro-structure designs as skin-friction reducing technologies [23]. Some researchers have observed a vast variety of the previously mentioned structures on the skin of fast sharks [25–28] and its effects on the flow behaviour, specially in drag reduction. The studies of shark skin micro-structures suggest that they can greatly decrease turbulent skin friction when properly designed [1, 23–25, 28–30]. On the other hand, heat transfer applications are still unclear for these structures [29]. To fully understand how it is possible to obtain a geometry that enhances heat transfer while decreasing skin friction, the concept of Micro-Channel Heat Sink (MCHS) may be used [32, 41]. MCHS refers to high surface area channels for heat transfer enhancement with lower temperature differences [22, 31]. This concept is now widely adopted after Tuckerman and Pease [32, 41] first proposed a geometry based on it. The shark skin riblets increase the wetted surface area, making it a potential candidate for effective heat transfer enhancement while reducing drag. Shark skin riblets are placoid scales as illustrated in Figure 1.4, which exhibit a considerable variability in size, ranging from 0.2 mm to 0.5 mm. Superficially, they seem to align to the direction of the fluid flow to efficiently delay vortex formation, but on closer inspection the riblets interlock with each other and do not align with the neighbouring scales. Moreover, the number of riblets per scale varies from one to seven for different species [25]. In addition, their performance depends on its design variables, including cross-section profiles, dimensions and configurations [23]. Thus, the interaction of shark skin riblets with the fluid flow becomes a three dimensional geometry dependent problem.

1.4 Computational Fluid Dynamics

To study the thermo-hydraulic behaviour of flow in pipe systems, it is necessary to understand the physics involved in the fluid flow. The Navier-Stokes equations, derived during the nineteenth century, are highly complex and considered unsolvable analytically without assumptions and simplifications. Computational Fluid Dynamics (CFD) works on the basis of numerical discretisation to approximate the Partial Differential Equations (PDEs) that describe the behaviour of the fluid flow, namely, conservation of mass, linear momentum and energy. There are plenty of discretisation methods available [43–45] which have been verified and validated for engineering problems. The Finite Volume Method (FVM) is widely used in

industry due to its geometry flexibility, memory usage and solution speed [44, 45]. This method transforms the PDEs over differential volumes into discrete algebraic equations over finite control volumes –the cells of a grid– via integrating the equations over each discrete volume. The system is then solved to compute the dependent variables for each control volume [45], with terms evaluated at the control volume faces, named fluxes. This ensures the conservative nature of the transport equations, making FVM an attractive method for numerical simulation of fluid flow with heat and mass transfer.
To calculate the full behaviour of the basic governing equations, albeit at a large computational cost, Direct Numerical Simulation (DNS) is required. This method solves the whole range of temporal and spatial scales of turbulence, which needs extremely small space and time discretisation [45]. A way to avoid using the whole scope of turbulence scales without reducing accuracy excessively is via Reynolds-Averaged Navier-Stokes (RANS), applying statistical averaging to approximate the random fluctuations of turbulence. RANS is broadly used in the field due to its reduced computational cost. Considering the amount of cases required to built the MOO problem database, RANS is the selected model to compute the fluid flow behaviour.
In addition to the need for high-fidelity modelling, the nature of thermal and hydraulic performance optimisation in pipe flows introduces the dilemma of coupling solid/fluid thermal interactions. The benefit of modelling the solid region is clear in the context of topology optimisation problems [18, 40], where the THP of the system is evaluated using an adjoint solver considering the layout of both solid and fluid regions. This is a desired feature since the adjustments in geometry will affect the thermal gradient in the solid as much as in the fluid. In order to model conjugate heat transfer, the thermal transport for the solid is built along with the fluid flow model [40].

1.5 Data-driven modelling in the context of optimisation

It has been established that a large number of shape configurations are required for the purpose of pattern recognition and optimisation. Yet, it is unfeasible to recreate all possible scenarios

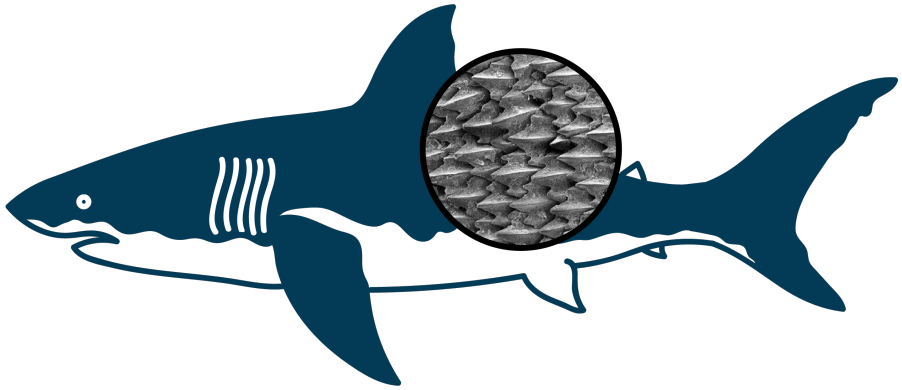


Figure 1.4: Dermal denticles of a porbeagle (*Lamna nasus*) shark. This could also be the dermal denticles of a spiny dogfish (*Squalus acanthias*) shark [42].

within CFD. Machine Learning (ML) is a technique used to develop statistical models from algorithms to process data, identify patterns and perform unseen tasks. These algorithms are used in artificial intelligence, generative pre-trained transformer tools and digital twins [46–50] in the context of engineering. The latter is of particular interest for this project, referring to a virtual model using real-time data to simulate the behaviour of a component or a system, while monitoring its operation. These *virtual twins* use ML algorithms to provide insights about performance optimisation and efficiencies of the system by identifying data patterns from the processing of large quantities of sensor data [51]. The data-driven reduced-order modelling used in digital twins is considered a real-time, complex virtual environment, but can be performed offline to obtain *surrogate models*, which have become a popular alternative to ease the computational penalty related to high-fidelity models [5, 52–57], with application in engineering areas such as magneto-mechanical problems [58], shape parametrisation [59], aerodynamic designs for aerofoils [52, 56, 59], and heat transfer problems [55, 60].

Consider each task is represented by a vector with a particular set of data parameters directly related to them [3, 61]. The underlying relationship can be difficult to see; however, if a large set is used to *train* or tune the hidden parameters of machine learning algorithms to represent the *target task vector*, a statistical model can be developed [62–64]. This is called *supervised learning*, where the data parameters correspond with their target task vectors, which aim to approximate *classification* or *regression* problems. *Classification* methods are designed to assign discrete categories to each parameter set, while *regression* outputs consist of continuous variables, more akin to understanding flow variables behaviour under shape parameters influence [3, 54, 61, 63]. Furthermore, more practical applications include *preprocessing* the data via *feature extraction* in order to transform the vectors into a space where either the pattern recognition problem is easier to solve or to ease computational costs in the *training phase* and real-time.

The most consistently used method for feature extraction and interpolation is the Proper Orthogonal Decomposition (POD) [53, 58, 59], which builds a low-dimensional approximate solution used to identify the dominant components, the POD coefficients. This has different interpretations depending on the field of study, and has been verified to have a great accuracy [52, 55, 56]. As previously stated, POD is used as an *unsupervised learning* method for surrogate modelling, where the target task vector is left unlabelled and the model does not respond to feedback from it. Instead, the model identifies commonalities in the data and infers features from these clusters, hence POD is more often used as a feature extraction technique.

Another approach that has become increasingly attractive for surrogate modelling is Artificial Neural Networks (NN) [3, 40, 52, 60, 63, 65–67]. NN's ability to perform predictions almost in real time after building the surrogate model has been exploited to observe either multi-output variables [40, 60] or the POD coefficients [5, 20, 52, 59] to then compute the MOO. Physics-Informed NNs are a notably interesting case for integrating laws of physics described by generalised non-linear PDEs, or in the case of CFD, the Navier-Stokes equations [20, 60].

An alternative surrogate model is Gaussian Processes (GP) [3, 4, 40, 62, 63, 68–73]. This is a Bayesian method that enables predictions with uncertainty estimates, with each linear combination of variables computed with Multi-Variate Normal (MVN) distribution [4, 61, 62]. The appealing quality of GPs is that as stochastic processes, they build adaptive fitting in regions of interest, with the downside of high sensitivity to the distribution of training data.

1.6 Scope and outline

This thesis takes an exploratory approach to the problem of THP optimisation for pipe flows in contact with parametrised geometries in the wetted surface, examining it through the perspectives of high-fidelity and surrogate modelling. The aim is to investigate how ML algorithms can support and enrich the computational design process within the context of open-ended research on optimisation for the coupled mechanisms of THP. The overall objectives are:

Obj I. Design open-source high fidelity computational models of steady-state, two-dimensional pipe cross-section flows in a conjugate heat transfer system with cold flow.

Obj II. Verification and validation of the numerical approach against analytical data and previous research studies.

Obj III. Explore parametrised geometries based on simplified biomimetic structures on the wetted surface of the pipe and their influence on heat transfer and flow behaviour.

Obj IV. Gain insight on the thermal and hydraulic mechanisms of the flow when influenced by different topologies.

Obj V. Identify the advantages and drawbacks of alternative ML techniques like RBF interpolation, NN, and GP when employed to evaluate the THP optimisation in pipe flows.

Figure 1.5 outlines the relation of each objective to the individual topics and global framework of the thesis. This guideline provides a visual aid to the extent and context of the project. The equations of motion for fluid flow and heat transfer are discussed followed by the numerical methods used to approach them. The mathematical formulations of the ML algorithms are postulated next along with POD, adopted as a dimension reduction method to assess the influence of data size and shape in the accuracy and computational cost of the surrogate models. As part of this project, a comparative data-driven computational framework was developed to approximate the THP optimisation process. This software, *Hammerhead*, incorporates the geometry parametrisation, high-fidelity modelling and simulation and surrogate model training using the high fidelity database. By the end of this thesis, the reader will have an understanding of the governing equations of fluid flow and heat transfer, their application in conjugate systems and THP, as well as the numerical and statistical methods used to approach the optimisation problem.

Outline of the thesis

Chapter 2: Aspects of fluid mechanics for conjugate heat transfer
Mathematical description of the general equations of fluid motion and heat transfer, as well as the derivation of THP from momentum and energy transport equations in conjugate systems.

Chapter 3: Elements of the high fidelity model implementation
Application of the FVM discretisation for the general equations of fluid flow and heat transfer for structured, body-fitted meshes, along with the presentation of the open-source platform

OpenFOAM v2212.

Chapter 4: Fundamentals of statistical "learning" in surrogate models
General formulation for the statistical learning techniques of POD, RBF, NN and GP.

Chapter 5: Integrated environment for Machine Learning based optimisation
Presentation of the Hammerhead software framework for process automatisation applied to high-fidelity database population and surrogate model training.

Chapter 6: High fidelity model: benchmark and verification
Benchmark of the high-fidelity models through $h-$refinement studies and analytical-numerical data comparison, presenting the FVM solutions to the THP optimisation problem.

Chapter 7: Surrogate models: benchmark and approximations
Benchmark of the surrogate models' architecture with varying input parameters and output dimensions with the verification of POD as a dimension-reduction technique for the output, presenting the models' solutions to the THP optimisation problem.

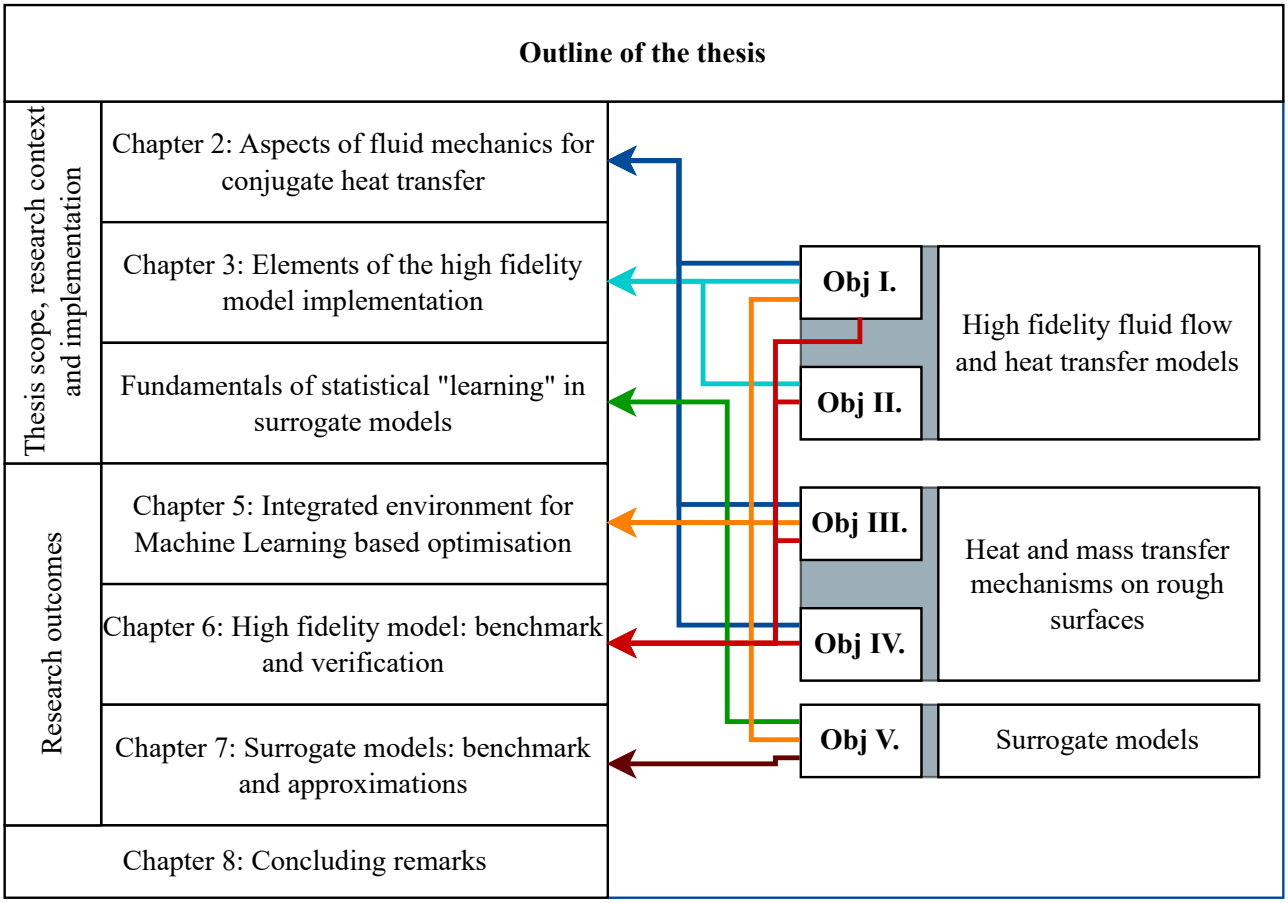


Figure 1.5: Overall objectives of the thesis and their relation to the framework.

Chapter 8: Concluding remarks
Summary of the novelties and future works presented by this research.

Appendix A: Mathematical foundations
Recollection of useful definitions and properties of mathematical formulations.

Appendix B: Fundamentals of fluid mechanics
Summary of basic concepts within fluid mechanics.

Appendix C: High fidelity model additional material
Review of additional information regarding the computational framework.

Appendix D: Hammerhead software additional material
Compendium of algorithms and snapshots of the open-source Hammerhead framework.

CHAPTER 2

Aspects of fluid mechanics for conjugate heat transfer

"When I meet God, I am going to ask him two questions: Why relativity? And why turbulence? I really believe he will have an answer for the first."

— Werner Heisenberg, Apocryphal attribution, 1976

2.1 Introduction

The field of fluid mechanics is concerned with the study of continua that deform without limit under shearing stress [74]. A particular area of fluid mechanics is the study of fluid motion, also known as *fluid dynamics*. To approach the problem of Thermo-Hydraulic Performance (THP) optimisation in pipe systems, it is crucial to understand the general behaviour of fluid flow from a physical and mathematical context. This chapter presents the fundamental physical laws for the description of fluid motion and heat transfer. The Eulerian framework is used to describe the governing equations of the continuum, with all the fields treated as continuous functions within a control volume. The derivation of such equations leads to a non-linear, underdetermined system that requires further assumptions and simplifications for it to be closed.
The incompressible steady-state Navier-Stokes equations form a powerful basis to model fluid flow problems that are considered *statistically* steady with no extreme discontinuities in the flow variable fields. These also form the basis for the derivation of the dissipation rate and advected heat flux, the THP quantities of interest defined to evaluate conjugate heat transfer systems. These quantities of interest are derived to obtain Multi-Objective Optimisation (MOO) target functions, applied to the parametrisation problem using biomimetic geometry structures on surfaces in contact with flow. The main objective of the chapter is to provide the basis of the underlying mechanisms of fluid flow in conjugate heat transfer to approximate a solution to the THP optimisation problem. It is recommended that the reader refers to Appendix A before reading this chapter.

2.2 General remarks on fluid motion

Since the phenomena considered are macroscopic, the fluid is regarded as a continuum [45, 74–77]. Under this hypothesis the properties of the fluid, or field variables, are treated as functions in space and time within the considered region, and obey suitable conservation laws [74, 75, 78, 79]. Consider a flow of fluid passing through an arbitrary volume fixed in space Ω bounded by a surface Γ with outward unit normal $\boldsymbol{n}$ as shown in Figure 2.1. The amount per unit mass of a flow quantity $Q(t)$ is represented by $\phi(\boldsymbol{x}, t)$. The Eulerian form of a general conservation law can be applied to the flow quantity Q as it convects through the fixed control volume Ω[1] [45, 76, 78, 80] as

$$\frac{d}{dt}\int_{\Omega}\rho\phi\, d\Omega + \int_{\Gamma}\boldsymbol{f}_c\cdot\boldsymbol{n}\, d\Gamma = -\int_{\Gamma}\boldsymbol{f}_i\cdot\boldsymbol{n}\, d\Gamma + \int_{\Omega} f_{\Omega}\, d\Omega\,, \tag{2.1}$$

where $\boldsymbol{f}_c = \rho\phi\boldsymbol{v}$ is the *convection* flux in the flow, ρ is the density and $\boldsymbol{v}$ as the fluid velocity, and f_{Ω} and $\boldsymbol{f}_i$ represent the volume source and the surface source –flux–, respectively. The volume source f_{Ω} measures the amount of Q inserted or removed within the volume, while the surface flux $\boldsymbol{f}_i$ measures the amount of Q flowing in and out across the surface boundary Γ [76, 80]. The integral form exhibits a number of valuable properties that allow it to satisfy conservation even after discretisation [76]. The most prominent property dictates that any variation of ϕ

[1] Recall that the Reynold's transport theorem states that $\frac{d}{dt}\int_{\Omega^*(t)}\rho\phi d\Omega = \frac{d}{dt}\int_{\Omega}\rho\phi d\Omega + \int_{\Omega}\mathrm{div}\,(\rho\phi\boldsymbol{v})\, d\Omega$, where $\Omega^*(t)$ is a moving control volume coinciding with the fixed control volume Ω.

depends only on the flux contributions across the surface Γ, assuming the absence of volume sources. The local differential form of the conservation law applied in Ω can then be derived by applying the Gauss theorem to (2.1) and introducing the divergence operator $\nabla\cdot$ as

$$\frac{\partial \rho\phi}{\partial t} + \nabla \cdot (\rho\phi \boldsymbol{v}) = -\nabla \cdot \boldsymbol{f}_i + f_\Omega \,. \tag{2.2}$$

Remark 1. *Note that if the flow variable $\boldsymbol{\phi}$ is a vector, the convective flux $\boldsymbol{f}_c$ is re-written as*

$$\boldsymbol{f}_c = \rho\boldsymbol{\phi} \otimes \boldsymbol{v} \,. \tag{2.3}$$

Both $\boldsymbol{f}_i$ and $\boldsymbol{f}_c$ transform into second order tensors. This results in the conservation law (2.2) re-written as

$$\frac{\partial \rho\boldsymbol{\phi}}{\partial t} + \nabla \cdot (\rho\boldsymbol{\phi} \otimes \boldsymbol{v}) = -\nabla \cdot \boldsymbol{f}_i + \boldsymbol{f}_\Omega \,, \tag{2.4}$$

where the body source transforms into a vector.

Remark 2. *Another possible flux-relevant contribution to the conservation equation is diffusion, related to the molecular thermal agitation, which can be present even when fluids*

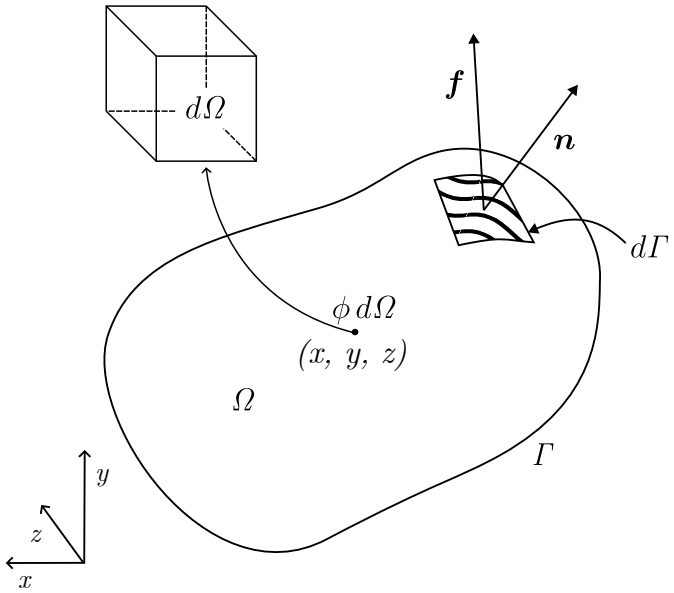


Figure 2.1: General form of a conservation law for a scalar quantity in a control volume $\mathrm{d}\Omega$ [76].

are at rest. In the process of diffusion, the local concentration of quantity ϕ tends to go from a high value to a low value until uniformity is reached. As described by Fick's law [76]

$$\boldsymbol{f}_d = -\boldsymbol{k}_d \nabla \phi \,, \tag{2.5}$$

where $\boldsymbol{k}_d$ is a diffusivity second order tensor. This positive definite second order tensor is often treated as a uniform constant –which is how it will be considered in this study– because of negligible changes due to temperature, location or orientation. For a vector variable, we use the fourth order tensor $\boldsymbol{K}_d$

$$\boldsymbol{f}_d = -\boldsymbol{K}_d : \nabla \boldsymbol{\phi} \,. \tag{2.6}$$

The addition of the diffusivity $\boldsymbol{f}_d$ results in the equation of the scalar of vectorial forms – assuming isotropic diffusivity $\boldsymbol{k}_d = k_d \boldsymbol{I}$, $\boldsymbol{K}_d = k_d \boldsymbol{I}$–

$$\frac{\partial \rho \phi}{\partial t} + \nabla \cdot (\rho \phi \boldsymbol{v}) = \nabla \cdot (k_d \nabla \phi) - \nabla \cdot \boldsymbol{f}_i + f_\Omega \,, \tag{2.7}$$

and for a vector variable $\boldsymbol{\phi}$

$$\frac{\partial \rho \boldsymbol{\phi}}{\partial t} + \nabla \cdot (\rho \boldsymbol{\phi} \otimes \boldsymbol{v}) = \nabla \cdot (k_d \nabla \boldsymbol{\phi}) - \nabla \cdot \boldsymbol{f}_i + \boldsymbol{f}_\Omega \,. \tag{2.8}$$

The motion of fluid is governed by the transport and conservation of three[2] basic properties: mass, momentum and energy [45, 75–79, 81, 82]. The set of these governing equations based on (2.2) applied to a viscous fluid is known as the Navier-Stokes equations.

2.3 The governing equations of fluid flow and heat transfer

By applying the general expressions (2.7) and (2.8) to the specific quantities of mass, momentum and energy, the basic fluid dynamic equations can be derived. These equations describe the behaviour of the system [76], yet further assumptions provide simplifying conditions to reduce the basic governing equations to more cost-effective computations [76, 77].
Newtonian, steady-state, incompressible fluid flow is of interest, as it mathematically decouples the energy conservation equation from mass and linear momentum [83], and reduces the mass conservation equation to the incompressibility constraint. This assumption results in the set of equations considered as the scope of this project.

2.3.1 Conservation of mass

The law of mass conservation states that *the time rate of change of mass in a fixed volume is equal to the net rate of flow of mass across the surface* [78, 79, 81]. This principle indicates that in the absence of mass sources and sinks, the total mass of the body remains constant [45, 83].

[2] This number rises to four when including the angular momentum, which implies symmetry of the *stress tensor*.

Thus, mass cannot disappear nor be created in a fluid system.
The flow variable is $\phi = 1$, leaving only the density ρ to work with. Note that mass can only be transported through convection, therefore, the fluxes are reduced to the convective term, which can be written as

$$\boldsymbol{f}_c = \rho \boldsymbol{v}\,, \quad \boldsymbol{f}_i = \boldsymbol{0}\,, \quad f_\Omega = 0\,, \quad \boldsymbol{f}_d = \boldsymbol{0}\,, \tag{2.9}$$

thus resulting in the total mass of fluid flowing in and out of the volume Ω as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0\,, \tag{2.10}$$

also known as the *continuity equation* [45, 74–76]. With the assumption of incompressibility[3], the rate of change of mass is zero $\frac{d\rho}{dt} = 0$ [45, 76, 79], so (2.10) can be expanded

$$\underbrace{\frac{\partial \rho}{\partial t} + \nabla \rho \cdot \boldsymbol{v}}_{=\frac{d\rho}{dt}} + \rho \nabla \cdot \boldsymbol{v} = 0\,, \tag{2.11}$$

thus the continuity equation (2.10) is simplified to the divergence free condition for the velocity

$$\nabla \cdot \boldsymbol{v} = 0\,. \tag{2.12}$$

2.3.2 Conservation of linear momentum

The principle of conservation of linear momentum indicates that *the time rate of change in linear momentum of a body is equal to the vector sum of all external forces acting on that body* [45, 74, 77–79, 83]. This expression represents $\phi = \boldsymbol{v}$, and the fluxes are defined as

$$\boldsymbol{f}_c = \rho \boldsymbol{v} \otimes \boldsymbol{v}\,, \quad \boldsymbol{f}_i = -\boldsymbol{\sigma}\,, \quad \boldsymbol{f}_\Omega = -\rho \boldsymbol{g}\,, \quad \boldsymbol{f}_d = \boldsymbol{0}\,, \tag{2.13}$$

where the body source $\boldsymbol{f}_\Omega$ represent the weight of material per unit mass in the presence of a gravitational field[4] and $\boldsymbol{\sigma}$ is the Cauchy stress tensor. In the context of this project, the gravitational field is neglected $\boldsymbol{f}_\Omega = \boldsymbol{0}$. Substituting the fluxes into the conservation law from (2.7), the momentum conservation equation is

$$\frac{\partial \rho \boldsymbol{v}}{\partial t} + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v}) = \nabla \cdot \boldsymbol{\sigma}\,. \tag{2.14}$$

The inability to sustain shear stress of a fluid translates in the stress tensor $\boldsymbol{\sigma}$ consisting of two terms. The Cauchy –total internal– stress[5] is decomposed as

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + \boldsymbol{\tau}, \tag{2.15}$$

[3] The effects of pressure changes on density are negligible, which is a good approximation for fluid flow under Mach ≈ 0.3 [45].

[4] In the particular case of conservation of linear momentum, the volume sources are considered a vector field, not a scalar quantity.

[5] Due to conservation of angular momentum, the total stress tensor $\boldsymbol{\sigma}$ –and consequently $\boldsymbol{\tau}$– is symmetric.

where p is the *hydrostatic stress* or pressure, $\boldsymbol{\tau}$ is the *deviatoric viscous stress*, $\boldsymbol{I}$ is the second order identity tensor, and the product $p\boldsymbol{I}$ is the isotropic stress tensor [45, 74, 76–79, 81]. The matrix representation of $\boldsymbol{\sigma}$ in the standard Euclidean basis is

$$[\boldsymbol{\sigma}] = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} = - \begin{bmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & p \end{bmatrix} + \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix} . \tag{2.16}$$

The main diagonal elements of the shear stress tensor $\boldsymbol{\tau}$ are normal to the surface element $d\Gamma$ as shown in Figure 2.2, while the non-diagonal elements are the shearing contributions [74]. All in all, *the shear stresses represent the internal friction of fluid layers against each other* [76], acting as surface sources.

Remark 3. *The property that measures the fluid's internal friction, or its resistance to flow, is the dynamic viscosity μ. With this understanding, and assuming a Newtonian fluid, the shear stress tensor is a linear function of the strain rate tensor [45]*

$$\boldsymbol{\tau} = \mu \left[\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T - \frac{2}{3} (\nabla \cdot \boldsymbol{v}) \boldsymbol{I} \right] . \tag{2.17}$$

When this shear stress tensor is introduced into (2.14) for incompressible fluids (2.12)

$$\rho \frac{\partial \boldsymbol{v}}{\partial t} + \rho \nabla \cdot (\boldsymbol{v} \otimes \boldsymbol{v}) = -\nabla p + \nabla \cdot \left[\mu \left(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T \right) \right] . \tag{2.18}$$

Then, by neglecting the transient term, the steady-state incompressible Navier-Stokes equations of motion are obtained

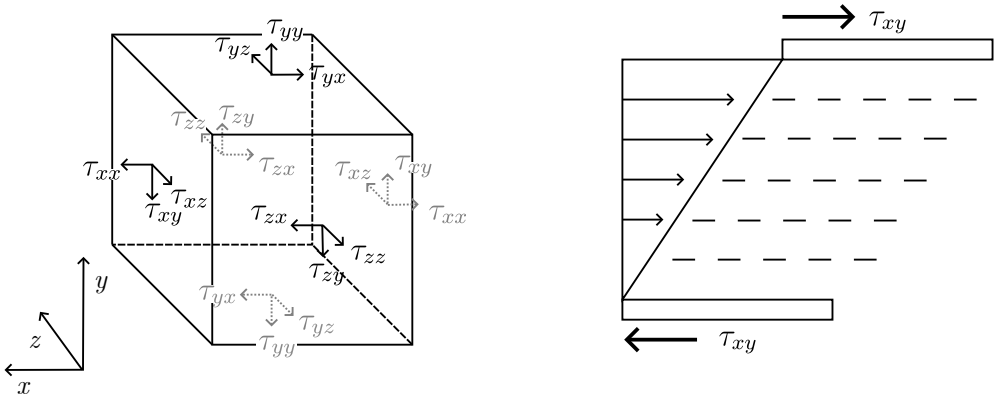


Figure 2.2: –Left– stress tensor components on the faces of a three-dimensional fluid element [78, 81]. –Right– Friction of fluid layers against each other in a fluid between flat plates when the top plate moves [74].

$$\rho\nabla\cdot(\boldsymbol{v}\otimes\boldsymbol{v}) = -\nabla p + \nabla\cdot\left[\mu\left(\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^T\right)\right]. \tag{2.19}$$

Due to incompressibility, the left-hand side term $\rho\nabla\cdot(\boldsymbol{v}\otimes\boldsymbol{v}) = \rho(\nabla\boldsymbol{v})\boldsymbol{v} + \rho\underbrace{(\nabla\cdot\boldsymbol{v})}_{=0}\boldsymbol{v}$ *can be rewritten as* $\rho(\nabla\boldsymbol{v})\boldsymbol{v}$.

2.3.3 Conservation of energy

The principle of conservation of energy, known as the First Law of Thermodynamics, states that *the time rate of change of total energy in an isolated system is equal to the sum of the rate of work done by applied forces and the change of added heat content per unit time* [45, 81]. Consider the specific total energy –per unit mass– as the sum of its specific internal and kinetic energies e and k, respectively

$$E = e + \frac{1}{2}\boldsymbol{v}\cdot\boldsymbol{v}. \tag{2.20}$$

As described previously, the total energy variation in a volume depends on the forces acting on it with the addition of heat sources transmitted to the system. By this understanding, and considering the volumetric sources as the sum of work of volume forces and heat sources other than conduction, the quantity $\phi = E$ can be substituted in the convective and diffusive fluxes to obtain

$$\boldsymbol{f}_c = \rho E\boldsymbol{v}, \quad \boldsymbol{f}_d = -k_d\nabla T, \quad \boldsymbol{f}_i = \boldsymbol{v}\cdot\boldsymbol{\sigma}, \quad f_\Omega = \dot{Q}_\Omega, \tag{2.21}$$

where T is the absolute temperature[6] and k_d the thermal conductivity. The sources $\boldsymbol{f}_i$, similarly to the momentum equation, derive from the stress tensor $\boldsymbol{\sigma}$, and $\dot{Q}_\Omega$ is the external heat source. Note that $\boldsymbol{f}_d$ (2.21) describes the diffusion of heat in a fluid at rest due to molecular conduction, called *Fourier's law of heat conduction* [76]. Substituting the fluxes into the general conservation law (2.2), the total energy equation is obtained

$$\frac{\partial\rho E}{\partial t} + \nabla\cdot(\rho E\boldsymbol{v}) = \nabla\cdot(k_d\nabla T) + \nabla\cdot(\boldsymbol{v}\cdot\boldsymbol{\sigma}) + \dot{Q}_\Omega. \tag{2.22}$$

This equation can be further transformed into the equations of specific internal energy e and specific enthalpy h. To rewrite the differential energy equation in terms of specific internal energy e, the inner product of the momentum equation (2.14) with the velocity vector $\boldsymbol{v}$ [45, 78, 81] leads to

$$\boldsymbol{v}\cdot\frac{\partial\rho\boldsymbol{v}}{\partial t} + \boldsymbol{v}\cdot[\nabla\cdot(\rho\boldsymbol{v}\otimes\boldsymbol{v})] = \boldsymbol{v}\cdot(\nabla\cdot\boldsymbol{\sigma}), \tag{2.23}$$

which can be shown –after some algebra [45]– to yield

$$\frac{\partial\rho k}{\partial t} + \nabla\cdot(\rho k\boldsymbol{v}) = \boldsymbol{v}\cdot(\nabla\cdot\boldsymbol{\sigma}), \tag{2.24}$$

[6] Since T is considered a variable per unit volume, the density ρ is dropped from the definition.

where $k = \frac{1}{2}\boldsymbol{v}\cdot\boldsymbol{v}$ is the kinetic energy. When subtracting (2.24) from (2.22), the specific internal energy equation can be obtained as

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho e \boldsymbol{v}) = \nabla \cdot (k_d \nabla T) + \boldsymbol{\sigma} : \nabla \boldsymbol{v} + \dot{Q}_\Omega \,. \tag{2.25}$$

To rewrite the internal energy equation in terms of the specific enthalpy h it is necessary to define this concept as *the sum of the system's specific internal energy and its pressure per unit mass* [45, 78, 81],

$$h = e + \frac{p}{\rho} \,. \tag{2.26}$$

By substituting (2.26) in (2.25), it yields

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho h \boldsymbol{v}) = \nabla \cdot (k_d \nabla T) + \boldsymbol{\sigma} : \nabla \boldsymbol{v} + \dot{Q}_\Omega + \frac{\partial p}{\partial t} + \nabla \cdot (p \boldsymbol{v}) \,, \tag{2.27}$$

which can be rewritten as

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho h \boldsymbol{v}) = \nabla \cdot (k_d \nabla T) + \boldsymbol{\tau} : \nabla \boldsymbol{v} + \dot{Q}_\Omega + \frac{\partial p}{\partial t} + (\nabla p) \cdot \boldsymbol{v} \,. \tag{2.28}$$

For a steady-state incompressible flow, the above equation simplifies to

$$\nabla \cdot (\rho h \boldsymbol{v}) = \nabla \cdot (k_d \nabla T) + \boldsymbol{\tau} : \nabla \boldsymbol{v} + \nabla \cdot (p \boldsymbol{v}) + \dot{Q}_\Omega \,. \tag{2.29}$$

Note that due to incompressibility, (2.25) or (2.28) are decoupled from equations (2.12) and (2.19). These equations add the primitive variable temperature T to the list of unknowns, but they can be further manipulated to render an equation in terms of T. To achieve this, recall that in general

$$\begin{aligned} \frac{\partial \rho \phi}{\partial t} + \nabla \cdot (\rho \phi \boldsymbol{v}) &= \rho \frac{\partial \phi}{\partial t} + \phi \frac{\partial \rho}{\partial t} + \phi \nabla \cdot (\rho \boldsymbol{v}) + \rho \boldsymbol{v} \cdot \nabla \phi \\ &= \rho \left(\frac{\partial \phi}{\partial t} + \boldsymbol{v} \cdot \nabla \phi \right) + \phi \underbrace{\left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) \right)}_{=0} \\ &= \rho \frac{d\phi}{dt} \,. \end{aligned} \tag{2.30}$$

Hence, $\frac{d\rho h}{\partial t} + \nabla \cdot (\rho h \boldsymbol{v}) = \rho \frac{dh}{dt}$ $(\phi = h)$. Assuming $h(t) = \overline{h}(p, S) = \overline{h}(p, \widetilde{S}(p, T)) = \widetilde{h}(p, T)$[7] where S[8] denotes entropy, we can use the Maxwell's relations for thermodynamic potentials derivations [84, 85] to define

$$\frac{dh}{dt} = \frac{\partial \widetilde{h}}{\partial p}\frac{dp}{dt} + \underbrace{\frac{\partial \widetilde{h}}{\partial T}}_{c_p}\frac{dT}{dt} \,, \tag{2.31}$$

[7] h, $\overline{h}$, $\widetilde{h}$ are different functional representations of the enthalpy.
[8] S, $\widetilde{S}$ are different functional representations of the entropy.

$$\frac{\partial \widetilde{h}}{\partial p} = \underbrace{\frac{\partial \overline{h}}{\partial p}}_{=V} + \underbrace{\frac{\partial \overline{h}}{\partial S}}_{=T}\frac{\partial \widetilde{S}}{\partial p} = V + T\frac{\partial \widetilde{S}}{\partial p}\,, \tag{2.32}$$

$$\frac{\partial \widetilde{S}}{\partial p} = \frac{\partial}{\partial p}\left(-\frac{\partial G}{\partial T}\right) = -\frac{\partial}{\partial p}\left(\frac{\partial G}{\partial T}\right) = -\frac{\partial \widetilde{V}}{\partial T}\,, \tag{2.33}$$

where ς_p is the specific heat at constant pressure, G is the Gibbs' energy and $V = 1/\rho$ is the specific volume. Considering the definition $\alpha = \frac{1}{V}\frac{dV}{dT}\big|_p$, and $V = \widetilde{V}(p,T)$[9], (2.31) and (2.32) become

$$\frac{d\widetilde{h}}{dp} = V\left(1-\alpha T\right)\,, \tag{2.34}$$

$$\frac{dh}{dt} = V\left(1-\alpha T\right)\frac{dp}{dt} + \varsigma_p\frac{dT}{dt}\,. \tag{2.35}$$

Since $V = 1/\rho$, (2.35) becomes [45, 86]

$$\frac{dh}{dt} = \rho\varsigma_p\frac{dT}{dt} + \left(1-\alpha T\right)\frac{dp}{dt}\,, \tag{2.36}$$

and thus, (2.28) can be transformed into the transport equation in terms of temperature as

$$\rho\varsigma_p\frac{dT}{dt} = \nabla\cdot\left(k_d\nabla T\right) + \alpha T\frac{dp}{dt} + \boldsymbol{\tau}:\nabla\boldsymbol{v} + \dot{Q}_\Omega\,. \tag{2.37}$$

Assuming ς_p is constant, the above equation can be rewritten as[10]

$$\varsigma_p\left(\frac{\partial \rho T}{\partial t} + \nabla\cdot\left(\rho T\boldsymbol{v}\right)\right) = \nabla\cdot\left(k_d\nabla T\right) + \alpha T\frac{dp}{dt} + \boldsymbol{\tau}:\nabla\boldsymbol{v} + \dot{Q}_\Omega\,. \tag{2.38}$$

Considering a steady-state incompressible problem, $\alpha = 0$, the transport of energy in terms of temperature is

$$\varsigma_p\nabla\cdot\left(\rho\boldsymbol{v}T\right) = \nabla\cdot\left(k_d\nabla T\right) + \boldsymbol{\tau}:\nabla\boldsymbol{v} + \dot{Q}_\Omega\,. \tag{2.39}$$

Remark 4. *Note that for incompressibility, as $\alpha = 0$ and ρ is constant, the first term in the right-hand side of (2.36) becomes*

$$\varsigma_p\frac{dT}{dt} = \frac{d}{dt}\left(h - \frac{p}{\rho}\right)\,, \tag{2.40}$$

[9] V, $\widetilde{V}$ are different functional representations of the specific volume.

[10] Recalling that the Reynold's transport theorem states that $\frac{d}{dt}\int_{\Omega^*(t)}\rho\phi d\Omega = \frac{d}{dt}\int_\Omega \rho\phi d\Omega + \int_\Omega \mathrm{div}\left(\rho\phi\boldsymbol{v}\right)d\Omega$, where $\Omega^*(t)$ is a moving control volume coinciding with the fixed control volume Ω.

or alternatively, in steady-state

$$\varsigma_p \nabla \cdot (\rho T \boldsymbol{v}) = \nabla \cdot \left(\left(h - \frac{p}{\rho} \right) \rho \boldsymbol{v} \right) . \tag{2.41}$$

Thus

$$\int_{T_0}^{T} dT = \frac{1}{\varsigma_p} \int_{t_0}^{t} \frac{d}{dt} \left(h - \frac{p}{\rho} \right) dt \,, \tag{2.42}$$

$$T = T_0 + \frac{1}{\varsigma_p} \int_{t_0}^{t} \frac{d}{dt} \left(h - \frac{p}{\rho} \right) dt \,. \tag{2.43}$$

In addition, with the assumption of incompressibility, the definition of specific heat capacity is the same at constant pressure and constant volume ($\varsigma_p = \varsigma_v$). This is proven in Appendix B.4 with the help of Maxwell's relations [84, 85].

2.3.4 Conjugate heat transfer

It is within the interest of this project to study the thermal and hydraulic effects of different shape parameters of a surface interacting with a fluid domain. In a system considering only a fluid *region* with volume Ω described by solely computing the Navier-Stokes equations previously established, this *surface* is part of the volume's outer boundary Γ. Nonetheless, in reality the variation of temperature within the fluid system depends on the interaction with a solid region, a process referred to as *conjugate heat transfer* [20]. For a rigid body at rest in domain Ω_s, deformation and movement are considered zero. Therefore, the external body forces do not exert any influence in the internal behaviour of the region, and due to velocity being zero in a rigid body at rest $\left(\boldsymbol{v}|_{\Omega_s} = \mathbf{0} \right)$ most of the terms from (2.28) are reduced

$$\frac{\partial \rho h}{\partial t} + \underbrace{\nabla \cdot (\rho \boldsymbol{v} h)}_{=0} = \nabla \cdot (k_d \nabla T) + \underbrace{\frac{dp}{dt}}_{= \frac{\partial p}{\partial t} + \nabla p \cdot \boldsymbol{v}} + \underbrace{\boldsymbol{\tau} : \nabla \boldsymbol{v}}_{=0} + \underbrace{\dot{Q}_\Omega}_{=0} , \tag{2.44}$$

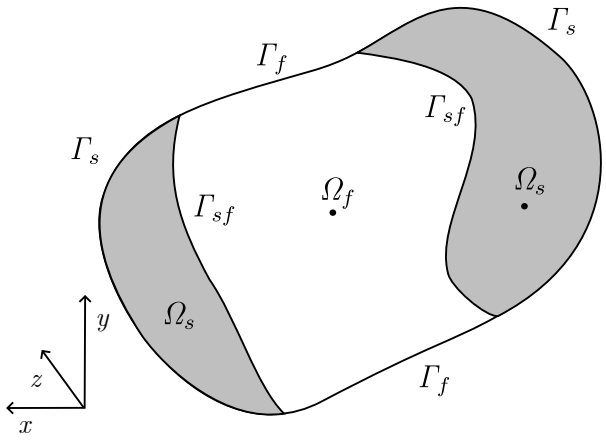


Figure 2.3: General control volume of a multi region, conjugate heat transfer system [20, 39].

which simplifies the enthalpy conservation equation for the solid region to

$$\frac{\partial \rho h}{\partial t} = \nabla \cdot (k_d \nabla T) + \frac{\partial p}{\partial t}\,, \tag{2.45}$$

and reduced in steady-state to

$$\nabla \cdot (k_d \nabla T) = 0\,. \tag{2.46}$$

The temperature field of both regions is analysed as a coupled system and it is necessary to build the thermal transport within the solid as well as the Navier-Stokes equations [40]. To observe the thermal fluid-structure interaction, Figure 2.3 represents the control volume Ω bounded by the control surface Γ, with solid and fluid subdomains, respectively denoted Ω_s and Ω_f, such that $\Omega = \Omega_f \bigcup \Omega_s$ and $\Omega_f \bigcap \Omega_s = \emptyset$. The fluid and solid variables are denoted by subscript f and s, respectively. The governing equations of fluid motion characterised in (2.29) describe the fluid region behaviour Ω_f, while the energy equation for the solid region Ω_s is the steady-state energy equation from (2.45)

$$\nabla \cdot (k_d \nabla T)|_{\Omega_s} = 0\,. \tag{2.47}$$

Further considerations have to be taken into account for the interface between regions denoted by a unit normal $\boldsymbol{n}$ at Γ_{fs}. To preserve continuity, the temperature shared between regions at this interface has to be the same, while heat flux leaving a region has to be the same entering the other [87]. Mathematically, this means that

$$T|_{\Omega_f} = T|_{\Omega_s}\,, \tag{2.48a}$$

$$(k_d \nabla T)|_{\Omega_f} \cdot \boldsymbol{n} = (k_d \nabla T)|_{\Omega_s} \cdot \boldsymbol{n}\,. \tag{2.48b}$$

Now, it is necessary to understand the behaviour of the system and underlying mechanisms of mechanical energy and heat transfer to undertake the derivation of the THP evaluation.

2.4 Thermo-hydraulic performance evaluation

It is clear from the equations in the previous sections that heat transfer between a solid body and fluid flow involves the physical motion of the fluid superimposed by a flow of heat, and that these fields interact [88, 89]. The THP optimisation involves minimising the dissipation rate of energy related to hydraulics while increasing the removal of excess heat from a solid surface to the fluid [18–20, 23, 32, 39, 90]. The dissipative and recoverable energies require a trade-off, since maximising the heat transfer involves increasing the pressure drop, analogous to the dissipation rate of energy. The trade-off at hand leads to a non-unique solution of the MOO problem, obtaining a set of solutions known as the Pareto front [18]. In literature, this impasse is overcome usually by introducing an Aggregated Objective Function (AOF) or weighted-sum approach based on a linear combination of both functions. However, this approach generates optimised structures that are influenced by the selected weights [39]. To outweigh the problems generated by AOF, the derivation of such objectives then requires reviewing the equation of kinetic energy as well as the equation of energy in terms of temperature from Section 2.3.

2.4.1 Derivation of dissipation rate

The resistance of the fluid to flow caused by the viscous effects leads to *losses* in energy, an effect also referred to as pressure loss or pressure drop [1, 23, 91]. For the hydraulic mechanism of the flow, the steady-state kinetic energy equation (2.24) is transformed using the Gauss theorem

$$\int_{\Gamma_f} (\rho k \boldsymbol{v}) \cdot \boldsymbol{n} \, d\Gamma = \int_{\Omega_f} \boldsymbol{v} \cdot (\nabla \cdot \boldsymbol{\sigma}) \, d\Omega \,. \tag{2.49}$$

The stress tensor related term can be expanded with the definition of (2.15) as

$$\begin{aligned} \boldsymbol{v} \cdot (\nabla \cdot \boldsymbol{\sigma}) &= \nabla \cdot (\boldsymbol{\sigma} \boldsymbol{v}) - \boldsymbol{\sigma} : \nabla \boldsymbol{v} \,, \\ &= -\nabla (p \boldsymbol{v}) + \nabla \cdot (\boldsymbol{\tau} \boldsymbol{v}) + \underbrace{p \nabla \cdot \boldsymbol{v}}_{=0} - \boldsymbol{\tau} : \nabla \boldsymbol{v} \,, \end{aligned} \tag{2.50}$$

which by using the Gauss theorem on the pressure-related term, the stress tensor can be written in integral form as

$$\int_{\Omega_f} \boldsymbol{v} \cdot (\nabla \cdot \boldsymbol{\sigma}) \, d\Omega = -\int_{\Gamma_f} \boldsymbol{v} \cdot p \boldsymbol{n} \, d\Gamma + \int_{\Gamma_f} \boldsymbol{v} \cdot \boldsymbol{\tau} \boldsymbol{n} \, d\Gamma - \underbrace{\int_{\Omega_f} \boldsymbol{\tau} : \nabla \boldsymbol{v} \, d\Omega}_{\dot{D}} \,. \tag{2.51}$$

The second term in the right-hand side is considered typically negligible compared to the other two terms. The double dot product from the last term represents the Bernoulli description of turbulent viscous dissipation rate $\dot{D}$ between inlet and outlet boundaries, and is used to evaluate the hydraulic losses in optimisation studies [18–20, 90]. By substituting (2.51) into (2.49) we can obtain the objective function for dissipation rate f_f as

$$f_f(\boldsymbol{v}, p) = \dot{D} = -\int_{\partial\Omega_f} (p + \rho k) \, \boldsymbol{v} \cdot \boldsymbol{n} \, d\Gamma \,. \tag{2.52}$$

The definition of dissipation rate from (2.52) will be used as the hydraulic performance objective function within the MOO problem.

2.4.2 Derivation of advected heat flux

Now consider heat transfer as the exchange of thermal energy within the control system flowing from higher to lower temperature regions [45, 88]. Similarly to the dissipation rate, the derivation of an objective function related to heat transfer comes from the steady-state energy transport equation in terms of temperature from (2.39) in the form of surface integral, and by assuming no direct heat sources/sinks $\dot{Q}_\Omega = 0$

$$\int_{\Gamma_f} (\rho \varsigma_p T) \, \boldsymbol{v} \cdot \boldsymbol{n} \, d\Gamma = \underbrace{\int_{\Gamma_f} (k_f \nabla T) \cdot \boldsymbol{n} \, d\Gamma}_{\dot{Q}} + \underbrace{\int_{\Omega_f} \boldsymbol{\tau} : \nabla \boldsymbol{v} \, d\Omega}_{\dot{D}} \,. \tag{2.53}$$

It is worth noting that the viscous dissipation rate $\dot{D}$ appears as the second term on the right-hand side in (2.53), which relates both objective functions, facilitating a solution for the MOO problem. In the above equation, $\dot{Q}$ is the description of the advected thermal flux [18–20, 90] derived from (2.53) as f_h, which will be refered to as the net thermal power

$$f_h(\boldsymbol{v}, T) = \dot{Q} = \int_{\partial\Omega_f} (\rho\varsigma_p T)\, \boldsymbol{v} \cdot \boldsymbol{n}\, d\Gamma - \dot{D}\,. \tag{2.54}$$

These objective functions (2.52) and (2.54) based on fluid energy transport are related via $\dot{D}$. Even though most research works include a linear combination of the derived functions as the aggregated objective function to be solved by either an adjoint solver or a parametrisation study, this project will focus on the definition of the net thermal power output as the objective function.

2.5 Wall-bounded flow characteristics

According to Prandtl [88, 89], the viscous effects can be confined in most applications to the *boundary layer*, a thin region near the body surface illustrated in Figure 2.4. A fundamental assumption of a submerged body is that the surface-adjacent sublayer of fluid is at rest relative to the body. The boundary layer then becomes the region in which the fluid velocity changes from free-stream to the zero value at the body surface, encasing most of the velocity changes within it. This concept is translated to heat transfer, where the *thermal boundary layer* refers to the region in which the majority of temperature changes occur very close to the surface. This leads to the assumption that the flow is essentially *inviscid* and *non-heat-conductive* outside the boundary layer [88].
These boundary layer characteristics are consistent for *laminar* flows, where the flow near the wall moves along smooth paths in a streamlined fashion. This generally occurs in small ducts and low velocity flows, so when increasing the flow velocity or reducing the pipe's radius to a critical value, chaotic and random changes start to appear in the flow variables in time and

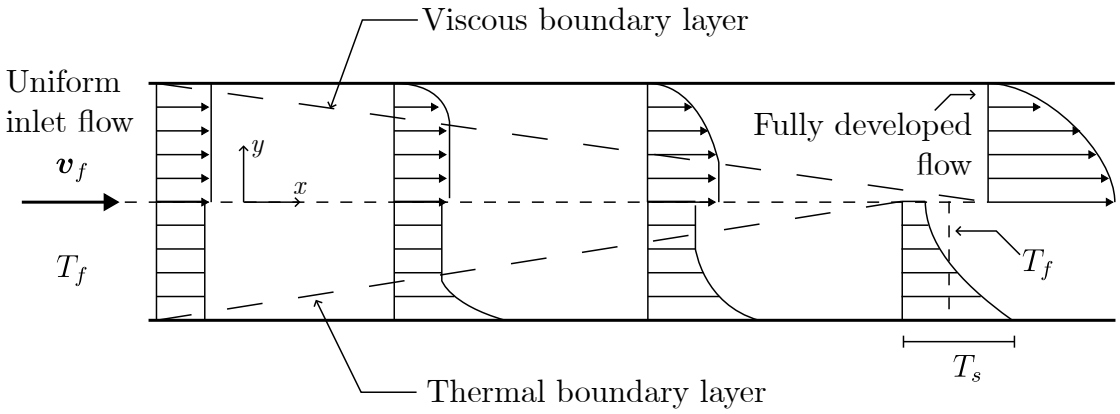


Figure 2.4: Velocity and temperature profiles in the development of flow in a tube [88].

space. This regime of flow is called *turbulence*, and in-between laminar and turbulent regimes there is a *transition* region. Within the transition and turbulent regimes, the inertial effects or *turbulent momentum* become sufficiently large to overcome the damping effects of viscous stresses, building up any small disturbance originated in the flow and increasing the boundary layer thickness, changing its characteristics. A *turbulent boundary layer* has three regions: the laminar or viscous sublayer, the buffer layer and the logarithmic layer as shown in Figure 2.5. The first, as it states in the name, shares the characteristics defined for laminar flows, where the viscous effects are dominant and the fluctuations are damped. The logarithmic layer is where turbulent momentum is dominant and the law of the wall[11] is applicable. The buffer layer has characteristics of both viscous and logarithmic sublayers, a high bulk of fluctuations and their interaction in this layer increases turbulent production in a process called *turbulent mixing*. In the context of heat transfer, the temperature profile within the *thermal* boundary layer follows the law of the wall when situated inside the turbulent *hydraulic* boundary layer.

Again, outside of the boundary layer, flow can be considered inviscid and non-heat-conductive, but within the boundary, turbulent coherent structures called *eddies*[12] or vortices are generated and dissipated due to the mentioned disturbances [23, 88]. The eddies that form in the streamwise direction are dominant in the viscous sublayer, moving along the surface and usually remain attached to the wall[13] until they dissipate in the cross-stream direction as shown in

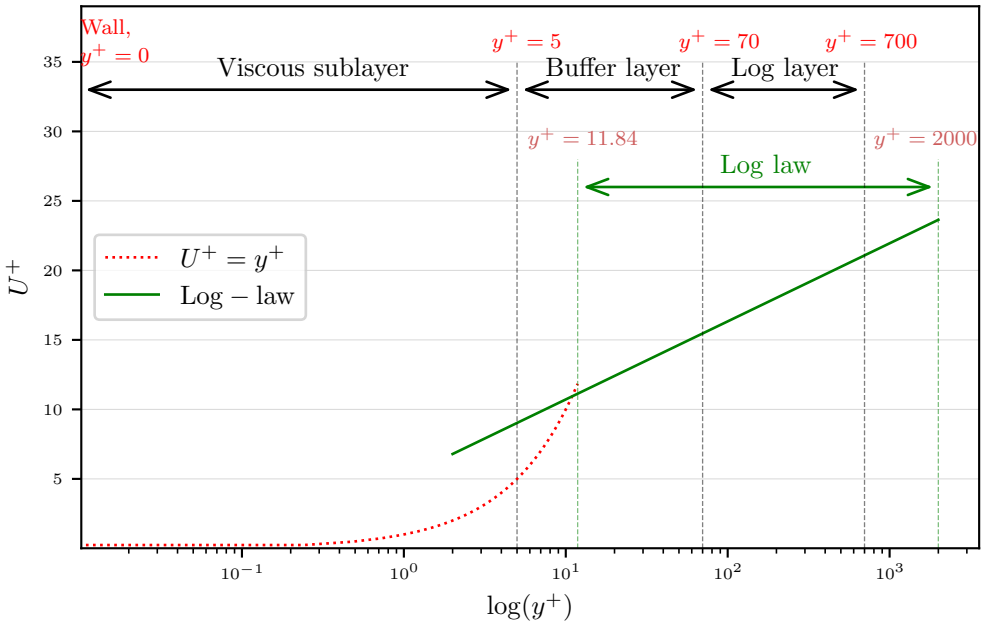


Figure 2.5: Mean velocity profile U^+ against dimensionless wall distance y^+ for boundary layer flow [92]. The y^+ is briefly described in Appendix B.1.

[11] The logarithmic law of the wall states that the average velocity at any point within the boundary sublayer of a turbulent flow is proportional to the logarithm of the point's distance to the surface.

[12] Swirling portions of fluid with fields of reverse current associated with turbulent momentum transfer.

[13] According to the Attached Eddy Hypothesis (AEH) [92, 93].

Figure 2.6, interacting with the surface and other eddies. This behaviour causes an increase in shear stress and, consequently, in the flow overall dissipation rate, while the specific heat is advected by the cross-stream turbulent mixing.

2.5.1 Surface roughness

The geometry of the wall is another factor that introduces disturbances in the flow and influences the transport of heat and mass. The disturbances caused by surface *roughness* disrupt the viscous sublayer, inducing changes in shear stress and advected heat, and ultimately leading to the formation of a *roughness sublayer* [92], the region associated with the flow influenced by roughness. This new sublayer has a thickness of 2-3 times the roughness height[14] and counteracts as a viscous-buffer sublayer situated under the logarithmic layer. Depending on the roughness height and its effects on the flow, it can be classified into three different roughness regimes [92, 94]:

- *Hydraulically smooth*, where the elements are contained within the viscous sublayer and have negligible impact on it;
- *Transitionally rough*, where the elements are contained within the viscous sublayer but have an observable impact on the flow behaviour;
- *Fully rough*, where the elements protrude into the upper layers and the viscous sublayer fully collapses.

The kind of impact this has in the flow depends on the roughness characteristics, incorporated with random or regular features of varying frequencies on height and density, either protruding

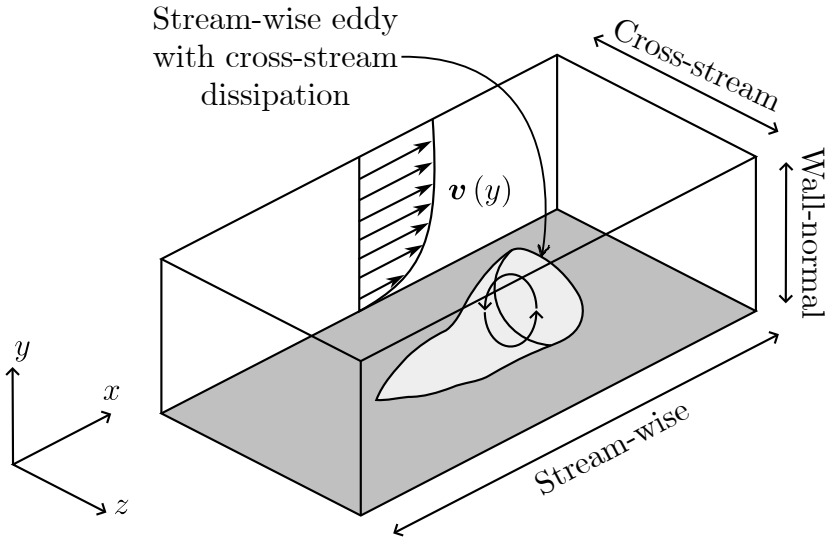


Figure 2.6: Schematic of a stream-wise eddy attached to the wall, dissipating in the cross-stream direction as it moves downstream.

[14] Equivalent sand-grain roughness [92, 94].

over the smooth surface or recessed under it. When these features are used correctly to design a surface geometry, they can present favourable control over the viscous sublayer [23, 92] and therefore, over the underlying mechanisms of the flow. Nature has become the inspiration for the design of these structures, particularly the texture on the skin of fast swimming sharks.
The small riblet denticles covering shark skin, illustrated in Figure 1.4, create a hydrophobic surface with roughness ranging from 100nm to 10µm in size, and have been studied to reduce pressure and skin friction losses [1, 25, 95–98], as well as enhancing heat transfer [32, 99–102]. Most of these studies are restricted to stream-wise continuous riblets, aligned parallel to the flow direction and with cross-section profiles of rectangular –blade–, parabola –scalloped–, triangular –sawtooth– and trapezoidal ribs, as shown in Figure 2.7. Additionally, the study of heat transfer over surface roughness is limited [23, 92] and there is conflicting evidence between experimental and computational methods [23, 103, 104], which has been associated with experimental errors due to free convection, lateral conduction and thermal radiation [23, 29, 101]. The lack of broad geometric parameter studies alongside the conflicting available data leads to a deficit in the fundamental understanding of fluid flow over rough surfaces. However, rough-wall flow characteristics have been identified and theories describing the mechanisms affecting the flow have been recognised in literature [23, 92].

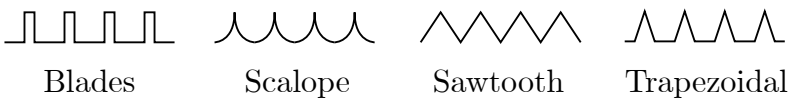


Figure 2.7: Conventional cross-section profiles adopted in the design of experimental and numerical biomimetic riblets [23].

2.5.2 Fluid flow and heat transfer over rough surfaces

The interactions between the flow and the wall are governed by the *form* and *contact* area of the surface, which resist the movement of the flow adjacent to the wall by blockage effects and frictional forces. These effects are analogous to the pressure drop and viscous shear stress contained within the boundary layer described in Section 2.5, and have a simultaneous influence on heat transfer, which is affected by the wetted surface area. The main consequence of these interactions is the formation of flow instabilities: Tollmien-Schlichting or 2D waves, secondary flow, reverse flow, separation and recirculation.
In general, it is well established that adding roughness on a surface in contact with fluid flow will increase the friction losses and pressure drop [92, 94], as well as enhance heat transfer due to the increase in wetted surface area [23, 92]. Furthermore, roughness can reduce the pipe cross-section average area if protruding from the surface or reduce it if is recessed, increasing the impact on fluid flow and heat transfer [92, 105][15]. Nevertheless, it is possible to use the flow mechanisms induced by roughness to generate favourable performance.
The formation of secondary flow is the most prominent behaviour observed in stream-wise continuous riblets [1, 23, 29, 91, 95, 98]. The fluid motions perpendicular to the main flow form

[15]Surface roughness relative to hydraulic diameter under 1% have a negligible effect.

when the inhomogeneities of the surface introduce gradients in the Reynolds stresses [92, 106] and therefore, in the velocity field and dissipation rate. This production and dissipation of turbulent kinetic energy close to the wall generates Low Momentum Pathways (LMP) in the valleys, associated with low shear stress, and High Momentum Pathways (HMP) with high shear stress in the peaks [92, 107–109]. LMPs bring the flow towards the wall, while HMPs carry it away from the wall which leads to the formation of stream-wise vorticity where these two regions interface, as shown in Figure 2.8. The structures are lifted from the wall [23, 25, 98], reducing the surface area with which they interact and consequently, the shear stress is reduced. The vortices then get pinned to the riblet peaks, which is thought to reduce the cross-stream fluctuations of the flow above the roughness features, weakening large-scale motions contained in the upper layers [23, 95, 98, 110, 111]. Martin [110] further discussed the behaviour impacted by secondary flow, reporting that when the vortices are not lifted, the velocity gradients lead to an increase in drag, while Benschop [112] observed no secondary flow influence in drag. The turbulence mixing from cross-stream fluctuations in secondary flow dominate the heat transfer distribution across the fluid, which is depleted if the mixing is reduced. This was observed in numerical results by Stalio and Nobile [29] and Jin [102], who reported increased drag coefficients whenever heat transfer enhancement occurred for structured riblets, contradicting the heat transfer enhancement with no penalty on drag observed in the experimental results of Walsh [99] and Lindemann [100].

However, the thermal interaction between the wall and the flow is not solely driven by the turbulent mixing, but can be controlled by inducing detached flow and recirculation. Detachment can happen as a wake behind elements with critical roughness height and separation width. The formed structures can develop reverse flow [92, 94, 113], creating stable or unstable counter-rotating vortices, as shown in Figure 2.8. Stable vortices get trapped and enhance thermal exchange between the wall and the flow with negligible hydraulic interaction, operating as part of the wall. Unstable eddies, on the other hand, are shed into the upper layers and increase dissipation rate in the flow, which promote turbulent mixing heat transfer. Separation with no eddy formation can reduce the thermal interaction by creating an insulation layer within the static flow [23], reducing significantly the heat exchange.

When considering laminar or underdeveloped turbulent flows, the instabilities generated by a rough surface can lead to a rapid transition to turbulence [114–117]. Natural transition is initiated by the generation of exponentially growing 2D waves as primary instabilities, followed

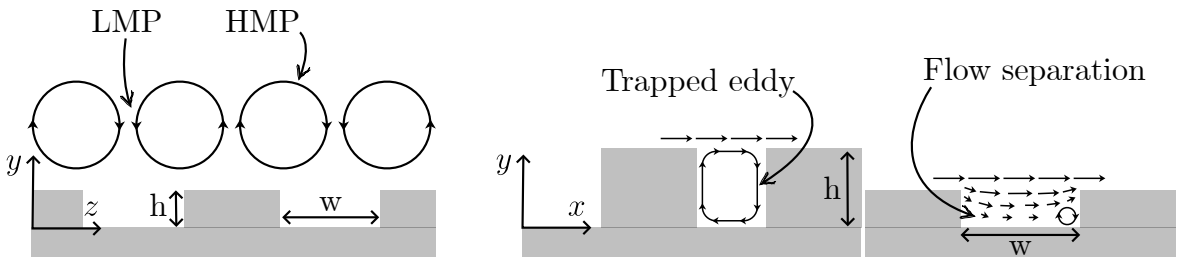


Figure 2.8: –Left– cross-stream view of secondary flow formation induced by stream-wise continuous riblets and –right– stream-wise view of trapped eddies and flow separation cases induced by stream-wise roughness.

by 3D secondary instabilities, like stream-wise and span-wise vorticity, that then finally break down into smaller structures, generating a fully developed turbulent boundary layer in the downstream region. By increasing the roughness on the wall the primary instabilities are linearly amplified or completely bypassed, a process that moves the transition region upstream. Bypass transition is induced by the 2 vortex formation mechanisms discussed previously, secondary and detached flow, where the disturbances introduced by roughness become stream-wise steady streak motions that become hairpin vortices[16].

Other relevant mechanisms observed in literature include the flow-wall conjugate interaction and the influence of the Prandtl number, described in Appendix B, in the heat transfer coefficients of the flow. Choi [101] observed an increase in the mean temperature gradient with a smaller thermal-viscous sublayer. Benhalilou and Kasagi [104] as well as Stalio and Nobile [29] observed high local convective heat transfer coefficients in the tip of the riblets contrasted with an abrupt reduction in the valleys with a fluid of low molecular $Pr = 0.71$, which led to no enhancement. When studying higher Pr fluids ($Pr = 100$), Benhalilou and Kasagi [104] state that the local convective heat transfer coefficients in the valleys have higher values than those at low Pr, indicating enhancement. This contradicts the findings of Wiesche [103], who reports heat transfer enhancement within the drag reduction regime at $Pr = 0.01$. Overall, the study of fluid flow and heat transfer over rough surfaces is an active area of research which requires further investigation.

[16]Thin, wall-attached corkscrew-like vortices with a potentially dominant role in the near-wall turbulence dynamics.

2.6 Conclusion

This chapter presented the governing equations of fluid motion in Section 2.2, with a focus on the of steady-state and incompressible mass, momentum and energy equations in Section 2.3 as relevant for this project. The conjugate heat transfer process was described in Section 2.3.4, introducing the thermal energy equation applicable to the solid region, as well as the interface conditions between fluid and solid. Using the flow characterisation from the equations of motion, the viscous dissipation rate and the advected heat flux were derived in Section 2.4 as the selected objective functions for the Thermo-Hydraulic Performance (THP) optimisation. These concepts are brought to the attention of the reader for their application in the optimisation problem of pipe flows with a suitable shape parametrisation in the wetted surface.

Mass	$\nabla \cdot \boldsymbol{v} = 0$	in Ω_f
Momentum	$\rho \nabla \cdot (\boldsymbol{v} \otimes \boldsymbol{v}) = \rho \boldsymbol{g} - \nabla p + \nabla \cdot \left[\mu \left(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T \right) \right]$	in Ω_f
Enthalpy	$\nabla \cdot (\rho \boldsymbol{v} h) = \nabla \cdot (k_d \nabla T) + \nabla \cdot (p \boldsymbol{v}) + \dot{Q}_\Omega$	in Ω_f
Temperature	$T = T_0 + \frac{1}{\varsigma_p} \int_{t_0}^{t} \frac{d}{dt} \left(h - \frac{p}{\rho} \right) dt$	in Ω_f
Temperature	$\nabla \cdot (k_d \nabla T) = 0$	in Ω_s
Interface temperature	$T\vert_{\Omega_f} = T\vert_{\Omega_s}$	on Γ_{sf}
Interface heat flux	$(k_d \nabla T)\vert_{\Omega_f} \cdot \boldsymbol{n} = (k_d \nabla T)\vert_{\Omega_s} \cdot \boldsymbol{n}$	on Γ_{sf}
Dissipation rate	$f_f(\boldsymbol{v}, p) = \dot{D} = - \int_{\partial \Omega_f} (p + \rho k)\, \boldsymbol{v} \cdot \boldsymbol{n}\, d\Gamma_f$	
Advected heat flux	$f_h(\boldsymbol{v}, T) = \dot{Q} = \int_{\partial \Omega_f} (\rho \varsigma_p T)\, \boldsymbol{v} \cdot \boldsymbol{n}\, d\Gamma_f - \dot{D}$	

Table 2.1: Fluid motion and conjugate heat transfer conservation laws in steady-state, incompressible conditions and THP evaluation derived functions.

CHAPTER 3

Elements of the high fidelity model implementation

"Any devide in science is a window on to nature, and each new window contributes to the breadth of our view."

— Cecil Frank Powell, 1903-1969

3.1 Introduction

In Chapter 2 the mathematical description of the fluid flow was introduced. This chapter briefly presents the cell-centred Finite Volume Method (FVM) formulation, extensively used in Computational Fluid Dynamics (CFD) [76, 78] and adopted by OpenFOAM v2212 to discretise the system of governing equations for fluid motion and heat transfer. OpenFOAM is a free, open-source platform that supports a wide range of fluid dynamic applications and is distributed with a library of pre-designed solvers. This software benefits from a large, active user community, which aids in the professional release of constant development and research [87, 118, 119]. Additionally, the use of FVM within OpenFOAM has been thoroughly studied and verified as a robust method to approach fluid flow behaviour [45, 120].
It has been established in Section 2.3.4 that the fluid-solid interaction influences the thermal and hydraulic mechanisms in pipe flows with varying wetted surface geometries. Recalling the assumption of incompressibility from Section 2.3, the OpenFOAM solver `chtMultiRegionSimpleFoam` is designed to address the conjugate heat transfer problem considered for the study at hand: Newtonian, steady-state, incompressible fluid flow in a two-dimensional pipe cross section with simplified geometries in the fluid-solid surface interface. The solver has been verified and validated [87] with the use of the SIMPLE algorithm to address the equation discretisation of incompressible flows.
The chapter gives a concise description of the general semi-discretised form of the conservation law applied to mass, momentum and energy equations. The domain discretisation and some mesh quality and sensitivity analyses are discussed to provide an understanding of numerical errors introduced by poor geometrical alignment of the underlying mesh. An overview of the open-source software OpenFOAM v2212 is provided along with the description of the solver `chtMultiRegionSimpleFoam` and the mesh generation tool blockMesh.

3.2 Numerical discretisation

The process of discretisation replaces the exact solution of the flow quantity $\phi(\boldsymbol{x}, t)$ with their discrete evaluation at a set of points. This *numerical solution* follows two levels of modelling, one in relation to the physical domain or *space discretisation*, and the second in relation to the physical phenomena in study, or *equation discretisation* [45, 76]. When discretising the physical *space*, the arbitrary volume Ω considered in Chapter 2 is tessellated into a computational domain via grid element subdivision as shown in Figure 3.1, to allow for computations of the discretised equations. Once the computational domain has been defined, the underlying PDEs are transformed into a set of algebraic equations.

3.2.1 Finite volume method

Considering a control volume with centroid E bounded by surface $\mathrm{S_E}$ and neighbouring cell with centroid G as shown in Figure 3.2. The sum of the fluxes from (2.2) is $\boldsymbol{f} = \boldsymbol{f}_c + \boldsymbol{f}_i + \boldsymbol{f}_d$, which are treated along with the volume sources f_Ω by FVM as

$$\int_\Gamma \boldsymbol{f} \cdot \boldsymbol{n}\, d\Gamma \approx \sum_{k=1}^{\mathrm{S_E}} \mathbf{f}_{\mathrm{E},k} \cdot \mathbf{S}_{\mathrm{E},k}\,, \tag{3.1a}$$

$$\int_{\Omega} f_{\Omega}\, d\Omega \approx \mathrm{f}_{\Omega_\mathrm{E}} \mathrm{V}_\mathrm{E}\,, \tag{3.1b}$$

where S_E is the number of faces of the considered control volume, $\mathbf{S}_{\mathrm{E},k}$ the outward unit normal of the face k, and $\mathbf{f}_{\mathrm{E},k}$ its corresponding flux evaluation, while V_E is the volume of element E and $\mathrm{f}_{\Omega_\mathrm{E}}$ it's corresponding volume source. Note that the new terms and subscripts are non-italic, compared to the fluxes and fields in (2.2). The distinction is brought to separate the continuum description from the numerical discretisation. With that said, the discretised form

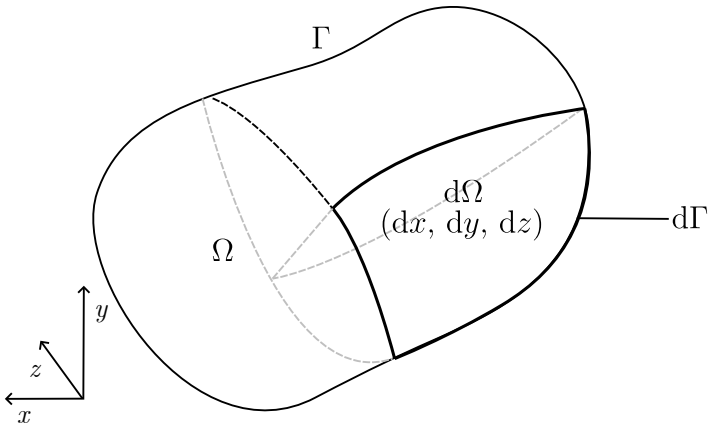


Figure 3.1: Subdivision of volume Ω into sub-volumes or control volumes dΩ [76].

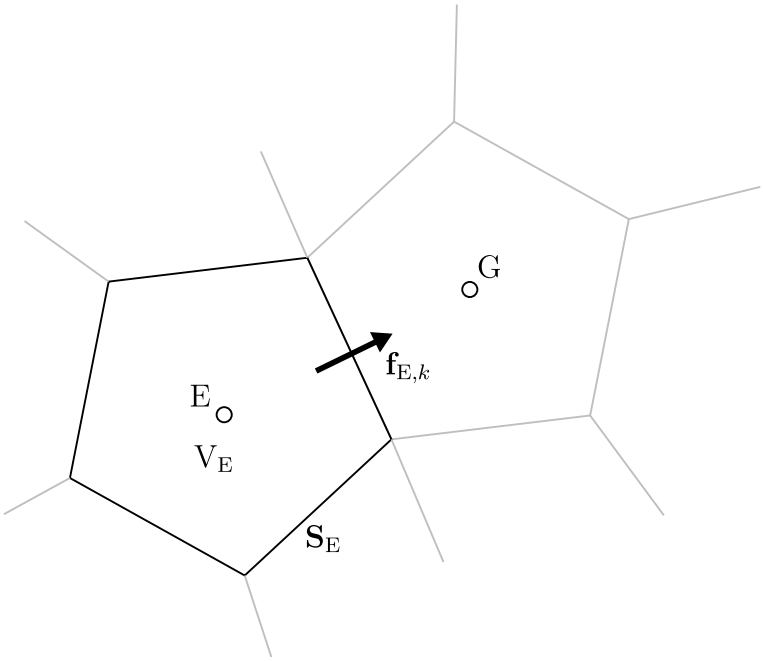


Figure 3.2: Discretised element with flux conservation across surface boundaries towards and from neighbouring cells.

of the steady-state conservation law from (2.2) is

$$\sum_{k=1}^{\mathrm{S_E}} \mathbf{f}_{\mathrm{E},k} \cdot \mathbf{S}_{\mathrm{E},k} = \mathrm{f}_{\Omega_\mathrm{E}} \mathrm{V_E} \,. \tag{3.2}$$

Consider a non-orthogonal grid such as the one shown in Figure 3.3, the position vector **EG** relating centroids E and G through the current face $\mathrm{S_{EG}}$ is not aligned to the vector $\mathbf{S_E}$. Consider $\mathbf{f}_{\mathrm{E},k}$ as a non-linear vector such that $\mathbf{f}_{\mathrm{E},k} = \mathbf{f}_{\mathrm{E},k}\left(\phi_\mathrm{E}, \phi_\mathrm{G}\right)$ and $\mathbf{S_E} = \mathbf{S_n} + \mathbf{S_t}$, then it is possible to write

$$\mathbf{f}_{\mathrm{E},k} \cdot \mathbf{S}_{\mathrm{E},k} = \mathrm{f}_{\mathrm{E},k}\phi_\mathrm{E} + \mathrm{f}_{\mathrm{G},k}\phi_\mathrm{G} + \mathrm{f_S}\left(\phi_\mathrm{E}, \phi_\mathrm{G}\right) \,, \tag{3.3}$$

$$\mathrm{f_E V_E} = \mathrm{f_E}\phi_\mathrm{E} + \mathrm{f_V}\left(\phi_\mathrm{E}\right) \,. \tag{3.4}$$

where $\mathrm{f}_{\mathrm{E},k}$, $\mathrm{f}_{\mathrm{G},k}$ and $\mathrm{f_E}$ are suitably defined coefficients for the flux contributions from E and G, and the volume sources from E, while $\mathrm{f}_{\mathrm{S},k}$ is a non-linear function that cannot be expressed in terms of ϕ_E or ϕ_G [45]. Hence

$$\sum_{k=1}^{\mathrm{S_E}} \mathbf{f}_{\mathrm{E},k} \cdot \mathbf{S}_{\mathrm{E},k} = \left(\sum_{k=1}^{\mathrm{S_E}} \mathrm{f}_{\mathrm{E},k}\right)\phi_\mathrm{E} + \underbrace{\sum_{k=1}^{\mathrm{S_E}} \mathrm{f}_{\mathrm{G},k}\phi_\mathrm{G}}_{=\sum_{\mathrm{GE}\wedge_\mathrm{E}} \mathrm{a_G}\phi_\mathrm{G}} + \sum_{k=1}^{\mathrm{S_E}} \mathrm{f}_{\mathrm{S},k} \,, \tag{3.5}$$

where $\wedge_\mathrm{E}$ are the centroids connected to G, and non-orthogonal corrections are reviewed further in Section 3.3.2. Thus

$$\left(\sum_{k=1}^{\mathrm{S_E}} \mathrm{f}_{\mathrm{E},k}\right)\phi_\mathrm{E} + \sum_{\mathrm{GE}\wedge_\mathrm{E}} \mathrm{a_G}\phi_\mathrm{G} + \sum_{k=1}^{\mathrm{S_E}} \mathrm{f}_{\mathrm{S},k} = \mathrm{f_E}\phi_\mathrm{E} + \mathrm{f_V}\left(\phi_\mathrm{E}\right) \,, \tag{3.6a}$$

$$\underbrace{\left(\left(\sum_{k=1}^{\mathrm{S_E}} \mathrm{f}_{\mathrm{E},k}\right) - \mathrm{f_E}\right)}_{=\mathrm{a_E}}\phi_\mathrm{E} + \sum_{\mathrm{GE}\wedge_\mathrm{E}} \mathrm{a_G}\phi_\mathrm{G} = \underbrace{\mathrm{f_V}\left(\phi_\mathrm{E}\right) - + \sum_{k=1}^{\mathrm{S_E}} \mathrm{f}_{\mathrm{S},k}}_{=\mathrm{b_E}} \,, \tag{3.6b}$$

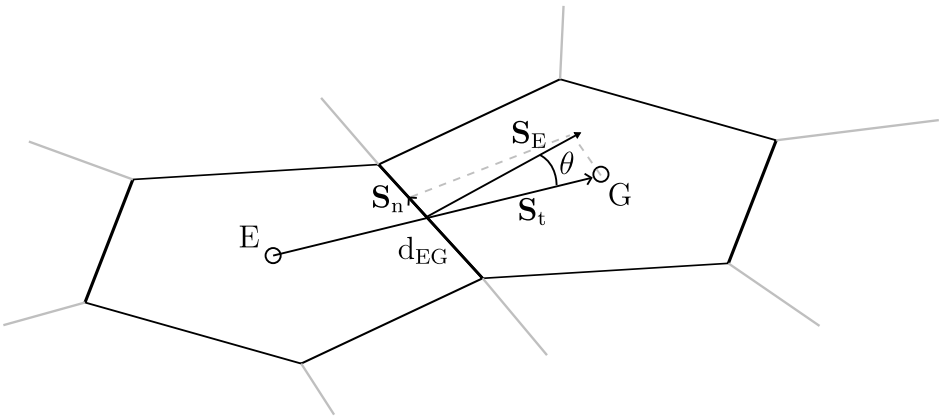


Figure 3.3: Non-orthogonality in an cell system, where $\mathbf{S_E}$ is decomposed to achieve flux linearisation. $\mathrm{d_{EG}}$ is the distance between E and G cell centres.

system/	constant/	0/
↪ controlDict	↪ polyMesh/	↪ field files
↪ fvSchemes	↪ thermoPhysicalProperties	
↪ fvSolution	↪ turbulenceProperties	
↪ blockMeshDict		
other dictionaries		

Table 3.1: OpenFOAM computer simulations directory configuration and general file names. Some other directories may be needed for particular solvers, such as `chtMultiRegionSimpleFoam`, which requires each regions' (fluid and solid) `changeDictionaryDict` to complement and fill the field files within `constant/`.

$$a_E \phi_E + \sum_{GE \wedge_E} a_{G,k} \phi_{G,k} = b_E \,, \tag{3.7}$$

3.3 OpenFOAM overview

OpenFOAM is an open source discretisation software developed by OpenCFD since 2004. The toolbox is used for the development of customised numerical solutions, with the added functionality of pre and post processing utilities [118, 119]. This software focuses in the solution of fluid mechanics, and it has become the leading open source software for CFD in industry and academia, across automotive, manufacturing, process engineering, among other areas of research and industry. The general structure of the software does not include a user interface, rather a directory which consists of two main branches: the OpenFOAM libraries and third-party libraries. This kind of structure is also inherited by the simulation configuration, where plain text input files are located in three main directories shown in Table 3.1. The file `controlDict` defines the simulation parameters, mainly the solver to use, time stepping, number of iterations to run and "mathematical precision" of the simulation. The other two files within `system/`, `fvSchemes` and `fvSolution`, contain respectively the numerical schemes for each equation term selected by the user, and the algorithm and mathematical models to use to solve the transport equations and each variable. The `polyMesh/` directory contains files with the mesh data, point, faces and edges locations, names and IDs, as well as boundaries. The `thermoPhysicalProperties` and `turbulenceProperties` contain, as they say, the constants and models for the material thermal parameters and turbulence, respectively. As for the `0/`, it contains the initial and boundary values. Because of its relevance to steady-state, incompressible conjugate heat transfer and therefore, to this project, the solver of choice within OpenFOAM libraries is the `chtMultiRegionSimpleFoam`, which is described next.

system/	constant/	0/
↪ Region/	↪ Region/	↪ p
↪ changeDictionaryDict	↪ thermoPhysicalProperties	↪ p_rgh
↪ fvSchemes	↪ turbulenceProperties	↪ T
↪ fvSolution	↪ polyMesh/	↪ U
↪ blockMeshDict	↪ boundary	↪ turulence related fields
↪ controlDict	↪ cellZones	↪ field files
↪ topoSetDict	↪ faces	
	↪ faceZones	
	↪ meshModifiers	
	↪ neighbour	
	↪ owner	
	↪ points	
	↪ pointZones	

Table 3.2: OpenFOAM's `chtMultiRegionSimpleFoam` computer simulations directory configuration and file names. The `Region/` directories are named after the specific region's name when running the topology setting file `topoSetDict`.

3.3.1 OpenFOAM's chtMultiRegionFoam solver

The solver `chtMultiRegionFoam` is a multi-physics solver for the conjugate heat transfer between fluid and solid regions. It supports multiple subdomains with physical phenomena described by different formulations of differential equations [87]. The `chtMultiRegionFoam` is a transient solver with the option of steady-state computations as `chtMultiRegionSimpleFoam` [87, 121]. The transient version uses the PIMPLE algorithm, which is a combination of PISO and SIMPLE algorithms. These are iterative procedures for coupling equations of momentum and mass conservation. The SIMPLE algorithm is used in the steady-state, incompressible version of the solver to address the coupling between the velocity and pressure, and the absense of pressure as a primary variable in the momentum and mass equations [45].

The mass and momentum equations solved in the fluid region of the steady-state solver are in the form presented in Chapter 2, summarised in Section 2.6 [121]. The particular file system of `chtMultiRegionSimpleFoam` is described in Table 3.2.

3.3.2 Numerical schemes

There are different methods to integrate the semi-discretised equations from Section 3.2.1 to obtain the face values ϕ_{E} and the coefficients related to $\mathbf{S}_{\mathrm{E},k}$. OpenFOAM includes different

discretisation processes for each term independent of the solver of choice. The relevant schemes for this project are presented in Table 3.3, with the definitions of current and neighbouring cells as stated in Section 3.2.1.
The interpolation schemes are the equations for the second order unbounded linear or *central difference* scheme, the second order unbounded linear-upwind or *central-upwind difference* scheme, and first order bounded *upwind* scheme. The central and central-upwind schemes, respectively, calculate the face value as

$$\phi_{\mathrm{S}} = \phi_{\mathrm{E}} + \frac{\phi_{\mathrm{G}} - \phi_{\mathrm{E}}}{\mathrm{d}_{\mathrm{EG}}} (\mathrm{d}_{\mathrm{SE}}) \,, \tag{3.8}$$

$$\phi_{\mathrm{S}} = \phi_{\mathrm{G}} + \frac{\phi_{\mathrm{G}} - \phi_{\mathrm{E}}}{\mathrm{d}_{\mathrm{EG}}} \mathrm{d}_{\mathrm{SG}} \,, \tag{3.9}$$

where d_{SE} is the distance between the current cell centre and the surface $\mathrm{S}_{\mathrm{E},k}$, and d_{SG} is the distance between the neighbouring cell centre and the surface $\mathrm{S}_{\mathrm{E},k}$. The central scheme uses a very simple linear interpolation at the face between two cells to compute the face value, taking both downstream and upstream cells, which leads to *convective instabilities* when applied to derivatives of odd order [45]. Convection is a highly directional processes, transporting the variable in the direction of the flow. To address this issue, upwind schemes compute the cell face value as a dependency on the upwind nodal value. The central-upwind scheme from (3.9) uses a biased stencil, and the value at the face is computed by extrapolation of the previous cell, not interpolation with the next cell, where the variable computation is done through the upwind scheme as

$$\phi_{\mathrm{S}} = \begin{cases} \phi_{\mathrm{E}}, & \text{if } \dot{\mathrm{m}}_{\mathrm{E},k} > 0 \\ \\ \phi_{\mathrm{G}}, & \text{if } \dot{\mathrm{m}}_{\mathrm{E},k} < 0 \end{cases} , \tag{3.10}$$

where $\dot{\mathrm{m}}_{\mathrm{E},k}$ is the mass flow rate from cell centre E, computed as

$$\dot{\mathrm{m}}_{\mathrm{E},k} = \sum_{k=1}^{\mathrm{S_E}} \rho_{\mathrm{E},k} \boldsymbol{v} \cdot \mathbf{S}_{\mathrm{E},k} = 0 \,. \tag{3.11}$$

Note that the upwind scheme has a truncation error known as the *stream-wise diffusion*, which stabilises the solution by keeping it bounded and physically correct, but makes the upwind scheme first order. In Table 3.3 the limiters refer to *gradient limiters*, which increase the stability of the method by bounding the face values to the neighbouring cells minimum and maximum limits. This limiter has a blending factor φ, which when under 1 allows the extrapolated value to exceed the neighbouring limits by a multiple of $\max(\phi_{\mathrm{G}}) - \min(\phi_{\mathrm{G}})$. The cellMDLimited introduces some diffusion to the solution, but helps instability, specially with high non-orthogonality in the mesh. Similarly, the grad $\mathbf{S}_{\mathrm{E}}$, or surface-normal gradient schemes, use a blending factor φ for meshes with high non-orthogonality. For the Laplacian terms, *uncorrected* scheme increases stability, while the *limited* with $\varphi = 1$ increases accuracy even with a relatively high non-orthogonality (maximum value of 70). For non-orthogonal corrections with high accuracy, the scheme used is the *Limited* with $\varphi = 1$, which uses an

Equation term	**Limiter**	**Interpolation**
$\frac{\partial \phi}{\partial t}$		Steady-state
$\nabla \phi$	CellMDlimited	Linear
$\nabla \cdot (\phi)$		Linear, Linear-upwind, upwind
Equation term	**Interpolation**	**grad $\mathbf{S}_\mathrm{E}$**
$\nabla^2 \phi$	Linear	Limited and Uncorrected
$(\nabla \phi) \cdot \mathbf{S}_\mathrm{E}$		Limited

Table 3.3: OpenFOAM numerical schemes in use within the scope of this project applied to the different terms of the transport equation presented in Section 2.3.

explicit non-orthogonal correction based on

$$\nabla \phi_\mathrm{S} \cdot \mathbf{S}_\mathrm{n} = \left(\mathbf{S}_\mathrm{EG} - \frac{1}{\cos(\theta)} \mathbf{EG} \right) \cdot \nabla \phi_\mathrm{S} \,. \tag{3.12}$$

3.3.3 OpenFOAM's blockMesh

The OpenFOAM structured mesh generation utility, `blockMesh`, creates parametric meshes with edge grading local refinement and curve fitting transformation for each edge, built on top of multi-block meshes. This tool is extremely useful for simple geometry parametrisation, as it enables the use of a multi-block mesh as a base before deforming it into a particular shape. This way, the mesh data is shared along all the parametrisation cases.
The grid is generated from the file `blockMeshDict` located in `system/`, which contains the vertices locations that will form one or more hexahedral blocks. These blocks are defined by a list with the 8 vertex number identifiers like the one in Figure 3.4, each block with the number of cells and the cell expansion ratios for each direction. The straight line edges of the blocks can be then transformed with the additional entries of interpolation points for the curve. The boundary of the mesh has to be imposed in a list, broken into *patches* or regions which will be the identifiers for the fields fixed conditions to be imposed on the field data files. Each patch contains a face defined by 4 of the vertices forming the block.
For multi-block connection, it is necessary to either *match* or *merge* the faces that are contiguous. Matching faces are not considered boundaries, and any boundary name given to either of the faces will be ignored by OpenFOAM. To merge faces, it is necessary to define them in the patches list and add them to the `mergePatchPairs` list, where the first will become the defined patch, while the second will be projected onto the faces of the first. For a full tutorial on this utility, please refer to OpenFOAM's tutorials [122].

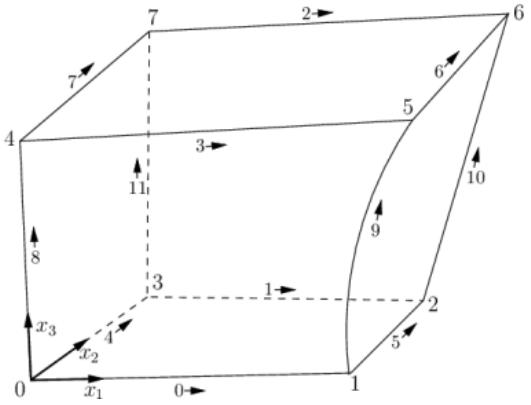


Figure 3.4: Block built in `blockMesh` with vertex locations, each with an identification number [122].

3.4 Conclusion

In this chapter, the domain and equation discretisation methodologies used within OpenFOAM are applied to the mathematical notions described in Chapter 2. The numerical discretisation follows the cell-centred FVM described in section 3.2, characterised by the mesh flexibility towards complex geometries whilst maintaining high accuracy. This method requires a boundary flux approach within each element, where the grid topology and quality, as well as the discretisation of each term in the equations, play a considerable role in the approximation's accuracy. Lastly, Section 3.3 is dedicated to an overview of the open-source platform OpenFOAM to inform the reader of the selected solver, `chtMultiRegionSimpeFoam`, the mesh generation tool available, `blockMesh`, and the relevant numerical scheme formulations provided by the software.

CHAPTER 4

Fundamentals of statistical "learning" in surrogate models

"After all, all devices have their dangers. The discovery of speech introduced communication—and lies. The discovery of fire introduced cooking—and arson. The discovery of the compass improved navigation—and destroyed civilizations in Mexico and Peru. The automobile is marvellously useful—and kills Americans by the tens of thousands each year. Medical advances have saved lives by the millions—and intensified the population explosion."

— Isaac Asimov, Robot visions, 1990

4.1 Introduction

Machine Learning (ML) is a field originated from the quest for Artificial Intelligence (AI), when some statistical algorithms were symbolically adopted to approach the problem of computers learning from data [123, 124]. Nowadays, AI is dominated by the emulation of reasoning and problem-solving abilities, whereas ML is concerned with classification and regression to make predictions from known data and perform pattern recognition. ML is a field that studies computer systems that adapt their response in order to perform unknown tasks by anticipating its behaviour based on previously obtained data [62, 64]. Within the computer system, functions are constructed to predict the missing information of the system or interpolate between the solutions for a new "cause", or feature.
ML is used to reduce the computational complexity of datasets with dimensionality reduction techniques, such as Linear Discriminant Analysis (LDA), Support-Vector Machines (SVM), autoencoders and Proper Orthogonal Decomposition (POD). POD is noteworthy due to its widespread use for feature extraction [5, 57, 125], where an integration with interpolation methods produces surrogate models with verified accuracy [52, 55, 56]. However, with the increasing abundance of data and processing power, algorithms have surged to compete with complex interpolation methods like Radial Basis Functions (RBF) [2, 126]. Examples include SVM, logistic regression, decision trees, k-Nearest Neighbors (k-NN), Gaussian Processes (GP) and Neural Networks (NN). NNs [3, 54, 63, 65, 123] emerged from linear regression models, but advances in deep learning have allowed them to outperform other ML algorithms through architectures such as deep, convolutional and recurrent networks, alongside transformers. Similarly, as a stochastic process, the rising use of GPs [4, 61, 62] for Bayesian optimisation problems takes advantage of their ability to adaptively model parametric regions of interest.
These surrogate models have turned into robust tools to approximate almost any engineering problem in real-time with low computational cost compared to high fidelity simulations, becoming increasingly popular to undertake optimisation processes. The advantage of surrogate over high fidelity modelling is apparent when taking into account the capability of interpolation between cases of the former, as well as modelling the mathematical relation of input features and output. Optimising the Thermo-Hydraulic Performance (THP) of pipe flows with varying wetted surface geometries represents a complex parametric study that can be aided by both dimensionality reduction and solution approximation techniques. Each approximation method has its benefits and disadvantages in computational cost and accuracy, and it is necessary to approach the modelling with an engineering basis and an *a priori* understanding of the problem. This chapter provides the mathematical descriptions of POD, RBF, NN and GP as the algorithms relevant to the framework of this project, along with the main numerical expressions for prediction and training of each corresponding model. A framework of the surrogate models' underlying mechanisms is presented to comprehend the processing of data during the *learning* process and after performing a new task.

4.2 Proper Orthogonal Decomposition

As stated before, one of the uses for ML is to perform dimensionality reduction and feature learning. This practice is significant for complex or high dimensional parametric

problems, where feature extraction enables the identification of latent representations in a lower dimensional space. Some models used to perform this task are Support-Vector Machines (SVM) and autoencoders, which are *supervised* learning methods developed for classification. SVM are linear classifiers based on the maximum-margin hyperplane model, while autoencoders are Neural Networks (NN) that encode and decode data to reconstruct inputs. Other methods, like Linear Discriminant Analysis (LDA) and Proper Orthogonal Decomposition (POD), identify combinations of variables to best describe the data, where LDA attempts to model the differences in *known* data classes, while POD extracts the *factors* to minimise possible variance in the data with no *a priori* knowledge of these factors.
POD stands out as an *unsupervised* method that enables the construction of low-dimensional approximate solutions from a high dimensional parameter identification problem [5, 57, 125]. In recent years, this method has been applied to parametric problems [52, 56, 58, 59], but it has various interpretations for different studies. The Principal Component Analysis (PCA) was developed within the field of statistics, Karhunen-Loeve Decomposition (KLD) appears in engineering research, and Singular Value Decomposition (SVD) is a discrete eigenvalue decomposition for non-square matrices [5, 125]. SVD is of particular interest since it provides a basis for the modal decomposition of an ensemble of functions. It is conducted to extract the dominate components as *mode shapes* or *POD modes*, from high-dimensional systems for subsequent use in Galerkin projections that yield low-dimensional dynamic models.
Consider a vectorial function $\mathbf{Y} \in \mathbb{R}^{\mathrm{N_d}}$ such that $\mathbf{Y}(\boldsymbol{\omega}, \mathbf{x})$ is

$$\mathbf{Y}(\boldsymbol{\omega}, \mathbf{x}) \approx \sum_{i=1}^{\mathrm{N}} a_i(\boldsymbol{\omega})\, \boldsymbol{\gamma}_i(\mathbf{x}) \ . \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^{\mathrm{N_x}}$ are measurement locations within the domain Ω, $\boldsymbol{\omega} \in \mathbb{R}^{\mathrm{N_s}}$ are the features at which the snapshots[1] or sets of data were taken, and where $\mathrm{N} \leq \mathrm{N_s} \leq \mathrm{N_d}$. The target is then to select an N large enough –but still relatively small compared to the whole dataset– to construct an approximation of $\mathbf{Y}$. This leads to the search for a set of coefficient functions $a_i(\boldsymbol{\omega})$ and basis functions $\boldsymbol{\gamma}_i(\mathbf{x})$ to correspond with this selection [57, 125].

First consider $\mathbf{D}$ as a matrix of snapshots of the form

$$\mathbf{D} = [\mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_{\mathrm{N_s}}]^T \ , \tag{4.2}$$

where $\mathbf{Y}_k = \mathbf{Y}(\boldsymbol{\omega}_k, \mathbf{x})$ is a matrix of snapshots. The truncated matrix of snapshots $\mathbf{D}$ of rank N is composed as [5, 52, 53, 57, 58, 125]

$$\begin{aligned} \mathbf{D} \approx \mathbf{D}_{\mathrm{N}} &= \mathbf{H}_{\mathrm{N}} \boldsymbol{\Sigma}_{\mathrm{N}} \mathbf{G}_{\mathrm{N}}^T \\ &= \sum_{i=1}^{\mathrm{N}} \underbrace{\mathbf{h}_i \sigma_i}_{:=\mathbf{p}_i} \mathbf{g}_i^T \\ &= \sum_{i=1}^{\mathrm{N}} \mathbf{p}_i \mathbf{g}_i^T \\ &= \mathbf{P}_{\mathrm{N}} \mathbf{G}_{\mathrm{N}}^T , \end{aligned} \tag{4.3}$$

[1] These snapshots refer to sets at which data is collected. In the context of this project, the features for each snapshot are amplitude and wavenumber of a harmonic function. An example is presented in Chapter 6.

where the columns of $\mathbf{H}_{\mathrm{N}}$ represent the left singular vectors of $\mathbf{D}$, $\mathbf{G}_{\mathrm{N}}^{T}$ is the transpose of $\mathbf{G}_{\mathrm{N}}$ with each column representing the right singular vectors. The matrix $\mathbf{\Sigma}_{\mathrm{N}} \in \mathbb{R}^{\mathrm{N_s}} \times \mathbb{R}^{\mathrm{N_x}}$ has all elements equal to zero except those along the main diagonal, or *rectangular diagonal matrix*. The diagonal consists of $\mathrm{N} = \min(\mathrm{N_s}, \mathrm{N_x})$ non-negative numbers σ_i, called the singular values of $\mathbf{D}$, which are uncorrelated and arranged in decreasing order as [5, 57]

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{\mathrm{N}} \,. \tag{4.4}$$

To construct an optimal low-order approximation of the function or a lower rank of matrix $\mathbf{D}$, the diagonal matrix $\mathbf{\Sigma}$ is truncated by the first $\mathrm{N} \lll \mathrm{N_x} \ll \mathrm{N_s}$ singular values into *the leading principal minor* $\mathbf{\Sigma} \in \mathbb{R}^{\mathrm{N}} \times \mathbb{R}^{\mathrm{N}}$ [52, 53, 57, 125]. Although, this task is not trivial, a discussion on the tolerance cut-off for the value of N is presented in Chapter 7. Therefore, the low-rank matrix $\mathbf{P}_{\mathrm{N}} \in \mathbb{R}^{\mathrm{N_s}} \times \mathbb{R}^{\mathrm{N}}$ is in a subspace as a low-dimensional model of the high-dimensional system $\mathbf{Y}$. These models are used to then reconstruct *a posteriori* the data using $\mathbf{G}_{\mathrm{N}}^{T}$.

Remark 5. *To observe the connection of SVD with eigenvalue decompositions [57, 125], it is necessary to pre-multiply (4.3) with its transpose*

$$\mathbf{D}^{T}\mathbf{D} = \left(\mathbf{H}\mathbf{\Sigma}\mathbf{G}^{T}\right)^{T} \mathbf{H}\mathbf{\Sigma}\mathbf{G}^{T} = \mathbf{G}\left(\mathbf{H}\mathbf{\Sigma}\right)^{T} \mathbf{H}\mathbf{\Sigma}\boldsymbol{G}^{T} \,. \tag{4.5}$$

Taking into account the orthogonality of $\mathbf{H}$ *and* $\mathbf{G}$ *this expression becomes*

$$\mathbf{D}^{T}\mathbf{D} = \mathbf{G}\mathbf{\Sigma}^{T}\mathbf{\Sigma}\mathbf{G}^{T} = \mathbf{G}\mathbf{\Sigma}^{2}\mathbf{G}^{T} \,, \tag{4.6}$$

where $\mathbf{\Sigma}^{2} \in \mathbb{R}^{\mathrm{N_x}} \times \mathbb{R}^{\mathrm{N_x}}$ *is the diagonal matrix with its elements as square singular values. This can be rearranged in the eigenvalue decomposition as*

$$\left(\mathbf{D}^{T}\mathbf{D}\right)\mathbf{G} = \mathbf{\Sigma}^{2}\mathbf{G} \,, \tag{4.7}$$

which shows that columns of $\mathbf{G}$ *are eigenvectors of* $\mathbf{D}^{T}\mathbf{D}$ *and the singular values* $\mathbf{\Sigma}$ *are the square roots of the eigenvalues of* $\mathbf{D}^{T}\mathbf{D}$*. This same process can be applied to the left singular vectors by pre-multiplying* $\mathbf{D}\mathbf{D}^{T}$*. In that regard, the expression leads to*

$$\left(\mathbf{D}\mathbf{D}^{T}\right)\mathbf{H} = \mathbf{\Sigma}^{2}\mathbf{H} \,, \tag{4.8}$$

showing that columns of $\mathbf{H}$ *are eigenvectors of* $\mathbf{D}\mathbf{D}^{T}$*.*

4.3 Radial Basis Function

An interpolation method can be used to approximate $\mathbf{Y}(\boldsymbol{\omega}, \mathbf{x})$ from (4.1) or $\mathbf{P}_{\mathrm{N}}$ from (4.3). Although there are a variety of methods for interpolation–piecewise constant, linear, polynomial, spline, function approximation, inverse distance weighting–, a Radial Basis Function (RBF) is the most attractive method due to its properties. RBF uses a linear

Distributions	
Gaussian	$\varphi\left(r\right) = e^{-(\epsilon r)^2}$
Multiquadric	$\varphi\left(r\right) = \sqrt{1 + \ (\epsilon r)^2}$
Inverse multiquadric	$\varphi\left(r\right) = \frac{1}{\sqrt{1 + \ (\epsilon r)^2}}$

Table 4.1: Example functions of RBF distributions available in the `Scipy` package `interpolate.RBFInterpolator` within Python [127].

combination of functions to estimate high dimensionality problems [2, 52, 126], which makes it an efficient method for complex parametric spaces. However, the main appeal lies in the correspondence of its formulation with (4.1) as

$$\mathbf{y}\left(\boldsymbol{\omega}\right) = \sum_{j=1}^{\mathrm{N_s}} \mathrm{w}_j \varphi\left(\boldsymbol{\omega}\right) \ , \tag{4.9}$$

where φ represents the basis function equation $\boldsymbol{\gamma}_i$ from (4.1) and w_j represents the coefficient functions a_i in (4.1). Note that this function is a linear system and is only dependent on $\boldsymbol{\omega}$, no longer dependent on the measurement locations $\mathbf{x}$. In this equation, w_j is the unknown weight coefficient, and φ is a kernel function. There are multiple choices of kernel functions, such as Gaussian, multiquadric and inverse multiquadric, described in Table 4.1. Within this study, the polynomial distributions are neglected due to their high sensitivity to the polynomial degree [127]. The goal of RBF is to find an interpolant $\mathbf{y}\left(\boldsymbol{\omega}\right)$, $\boldsymbol{\omega} \in \mathbb{R}^d$ in d unknowns, such that given the matrix of snapshots $\mathbf{D} = \left[\mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_{\mathrm{N_s}}\right]^T$ of known data, $\mathbf{y}\left(\boldsymbol{\omega}_i\right) = \mathbf{D}_i$ as a linear combination of translates of φ. If the data point $\boldsymbol{\omega}_j$ is recognised as the *centre* of φ, the available data can be used as interpolation –collocation– points by taking them from the set of unknown points to shift the functions [2, 126]

$$\mathbf{y}\left(\boldsymbol{\omega}\right) = \sum_{j=1}^{\mathrm{N_s}} \mathrm{w}_j \varphi\left(||\boldsymbol{\omega} - \boldsymbol{\omega}_j||\right) \ . \tag{4.10}$$

For (4.10) to have a unique solution, $\varphi\left(\boldsymbol{\omega}\right)$ is required to be non-singular [126]. The RBF method gives an approximation valid for the domain where the original data is situated, but the interpolation's error is influenced by several conditions such as the distribution of snapshots and the choice of kernel function [2, 125]. To fit the interpolant $\mathbf{y}\left(\boldsymbol{\omega}\right)$ to the given data snapshots $\mathbf{D}$, the linear system is solved for the weight coefficients w_j with respect of $\mathbf{D}$. The memory cost of this process increases quadratically with the amount of data points in $\mathbf{D}$ [127].

4.4 Neural Networks

An alternative method used to approximate $\mathbf{Y}\left(\boldsymbol{\omega}, \mathbf{x}\right)$ from (4.1) involves surrogate models based on *statistical learning*[2]. Some of the most common algorithms include decision trees,

[2]Statistical inference problem of finding a predictive function based on data to generalise unseen cases in the dataset [128, 129].

SVM, linear regression and Neural Networks (NN). Note that Gaussian Processes and other previously discussed models are neglected in the list, as this method and its alternatives are considered in Section 4.5. Decision trees are heuristic, open-box models that recursively split the data into subsets by evaluating the feature values against a criterion to predict a target variable. These models are easy to interpret and visualise, but the method is more suitable for data mining than parametric optimisation problems [3, 54, 130]. SVMs and linear regression, on the other hand, can be applied to parametric problems; however, as linear models, their performance may be limited for high-dimensional datasets with non-linear correlations in the data [3, 54, 131]. The method considered to be most suitable for parametric optimisation problems is NNs as they are *non-linear statistical learning* models that map an input to an output, learning the underlying relation between them [3, 65].
Albeit, NN algorithms encompass a wide class of architectures and learning methods. Transformers and feedforward, recurrent (RNN) and convolutional (CNN) networks use multilayered architectures to model non-linearity [3, 54, 65, 123]. CNNs are kernel-based feedforward NNs which, alongside transformers, are used for image reconstruction and natural language processing. RNNs use later processing stages to feed earlier stages in a sequence, mostly used for natural language processing, text and speech recognition.
The most common architecture is the feedforward. This is a two-stage model widely used for both regression or classification problems including parametric optimisation, typically represented as in Figure 4.1. The vertical subsection of units or *neurons* visible in Figure 4.1 is called a layer. The "$\boldsymbol{\omega}$" and "$\mathbf{y}$" are referred as the input and output layers, respectively, while the in-between layers are *hidden layers* comprised of *hidden units*. A feedforward NN relates each layer with the next without cycles, recurrent or *convolutional* calculations between layers. This means that the network will only feed the calculations to forward layers, and the signals are passed only in the forward direction through the different layers of neurons. It is a very simple yet non-linear model where the hypothesis is represented by the *hidden units*, exempting the input and output, and is named as such because the calculations and values are not directly observed [3, 65].

4.4.1 General architecture

Consider a network of L layers with N neurons per layer as shown in Figure 4.1. Each layer $l = 1, ...,$L has a weight matrix $\mathbf{W}^l \in \mathbb{R}^{N^l} \times \mathbb{R}^{N^{l-1}}$, a bias unit vector $\mathbf{b}$ related to the weight matrix, the input vector $\boldsymbol{\omega}$ and the output vector $\mathbf{y}$ [3, 54, 63, 123]. The forward computations of each layer are

$$\begin{aligned} \mathbf{z}^l &= \mathbf{W}^l \mathbf{g}^l \left(\mathbf{z}^{l-1} + \mathbf{b}\right) \\ &= \sum_{i=1}^{N} \mathbf{w}_i^l \mathrm{g}_i^l \left(\mathrm{z}_i^{l-1} + \mathbf{b}^l\right) , \end{aligned} \tag{4.11}$$

where

$$\mathbf{z}^0 = \boldsymbol{\omega} , \quad \mathbf{y} = \mathbf{z}^{\mathrm{L}} . \tag{4.12}$$

The hypothesis $\mathbf{y}$ is an approximation of the vectorial function $\mathbf{Y}$ referred to in Section 4.2, which makes $\mathbf{y}$ depend on the vector-valued input $\boldsymbol{\omega} \in \mathbb{R}^{\mathrm{N_s}} \times \mathbb{R}^{\mathrm{R}}$ with R feature components.

The function $\mathbf{y}(\boldsymbol{\omega})$ is selected via (4.11) and (4.12) based on a training set, with $\mathrm{N_s}$ input vector samples [64]. For the purpose of simplifying, the input vectors $\mathbf{x}$ are referred as *features*, and the output of the function $\mathbf{y}$ is called an *estimate*. The individual functions of the vector function $\mathbf{g}^l = \left[\mathrm{g}_1^l, ..., \mathrm{g}_N^l\right]^T$ are referred as the *activation functions* of each unit [54]. These can be any custom equation, most NNs related libraries in any software package already have optimised functions for particular situations. The chosen equation for all g_i^l for the $i-$th neuron is the *Sigmoid* given by

$$\mathrm{g}_i^l\left(\mathrm{z}\right) = \frac{1}{1+e^{-\mathrm{z}}}. \tag{4.13}$$

This information is then forwarded from layer to layer until the output is reached, with the output of one layer becoming the input of the next [123]. According to the universal approximation theorem [65], *feedforward neural networks with a sufficient level of complexity and non-constant activation functions can approximate any continuous function*. Nevertheless, these often need a lot of training data to represent a highly non-linear behaviour. Adding to this, these networks do not constraint their output to a physically reasonable range, which in turn limit their ability to extrapolate beyond their training data.

4.4.2 Backpropagation method

The training method used to obtain the weight parameters for a feedforward NN is a form of gradient descent called backpropagation [64, 123] that is a generalisation of the least Mean Square Error (MSE) method to deal with multilayer networks. With a set of examples

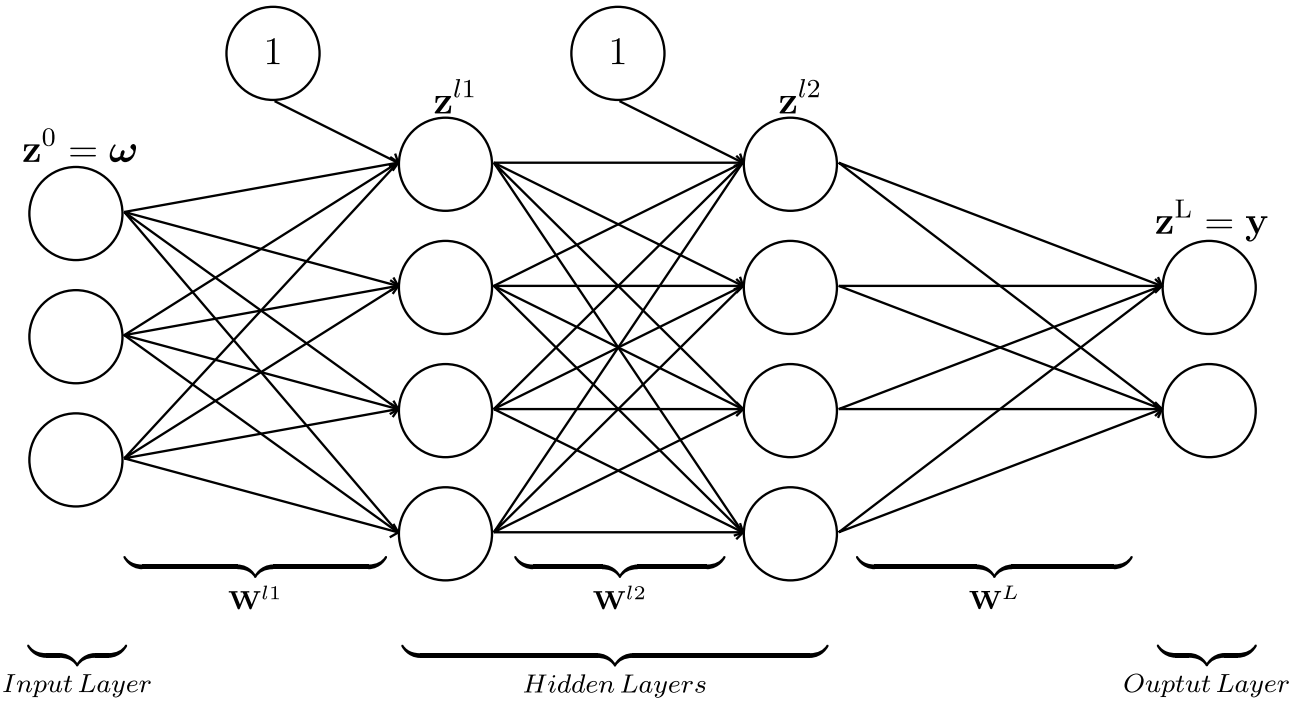


Figure 4.1: A muti-layer, multi-neuron feedforward network.

$\{\boldsymbol{\omega}_1, \mathbf{Y}_1\}$, ..., $\{\boldsymbol{\omega}_{\mathrm{N_s}}, \mathbf{Y}_{\mathrm{N_s}}\}$ as input with the corresponding target output exists as an available training set, the performance index of the backpropagation algorithm is the MSE [3, 64, 123]:

$$\mathrm{J}\left(\mathbf{W}\right) = \frac{1}{\mathrm{N_s}} \sum_{q=1}^{\mathrm{N_s}} ||\mathbf{Y}_q\left(\boldsymbol{\omega}_q\right) - \mathbf{y}_q\left(\boldsymbol{\omega}_q, \mathbf{W}\right)||^2 , \tag{4.14}$$

which with regularisation it becomes

$$\mathrm{J}\left(\mathbf{W}\right)_{regularised} = \frac{1}{\mathrm{N_s}} \sum_{q=1}^{\mathrm{N_s}} ||\mathbf{Y}_q\left(\boldsymbol{\omega}_q\right) - \mathbf{y}_q\left(\boldsymbol{\omega}_q, \mathbf{W}\right)||^2 + \underbrace{\frac{\lambda}{2\mathrm{N_s}} \sum ||\mathbf{W}||^2}_{regularisationterm} , \tag{4.15}$$

where $\mathbf{y}_q$ is the hypothesis response of the network for the features $\boldsymbol{\omega}_q$ and $\lambda > 0$ the manually tuned regularisation parameter. At each $k-$th iteration, there will be a gradient estimate of the form

$$\nabla \mathrm{J}\left(\mathbf{W}\right) = \frac{\partial \mathrm{J}}{\partial \mathbf{W}^l} . \tag{4.16}$$

Consider the sensitivity $\mathbf{s}_i^{\mathrm{L}}$ at last layer output $\mathrm{z}_j^{\mathrm{L}}$ defined by

$$\begin{aligned} \mathrm{s}_j^l &= \frac{\partial \mathrm{J}}{\partial \mathrm{z}_j^{\mathrm{L}}} \frac{\partial \mathrm{g}_j\left(\mathrm{z}_i^{\mathrm{L}-1}\right)}{\partial \mathrm{z}_i^{\mathrm{L}-1}} \\ \mathbf{s}^{\mathrm{L}} &= \frac{\partial \mathrm{J}}{\partial \mathbf{z}^{\mathrm{L}}} \frac{\partial \mathbf{g}\left(\mathbf{z}^{\mathrm{L}-1}\right)}{\partial \mathbf{z}^{\mathrm{L}-1}} , \end{aligned} \tag{4.17}$$

where i and j denote the connection between the i-th neuron of layer l and j-th neuron of layer $l-1$, respectively. The error is propagated back from the output to the in-between layers an all the way to the input in which the sensitivity at layer l is computed from the sensitivity at layer $l+1$ with the definition of J from (4.14)

$$\begin{aligned} \mathbf{s}^l &= \frac{\partial \mathrm{J}}{\partial \mathbf{z}^l} \frac{\partial \mathbf{g}\left(\mathbf{z}^{l-1}\right)}{\partial \mathbf{z}^{l-1}} \\ &= \left(\left(\mathbf{W}^{l+1}\right)^T \mathbf{s}^{l+1}\right) \frac{\mathbf{g}\left(\partial \mathbf{z}^{l-1}\right)}{\partial \mathbf{z}^{l-1}} . \end{aligned} \tag{4.18}$$

Considering that $\mathrm{J}\left(\mathbf{W}\right)$ is a composite function of the form $\mathrm{J}\left(\mathbf{g}\left(\mathbf{z}^{l-1}\right)\right)$ and $\mathbf{W}$ is a component of $\mathbf{z}$ the gradient can then be derived using the chain rule

$$\begin{aligned} \frac{\partial \mathrm{J}\left(\mathbf{g}\left(\mathbf{z}^{l-1}\right)\right)}{\partial \mathbf{W}^l} &= \underbrace{\frac{\mathbf{g}\left(\partial \mathbf{z}^{l-1}\right)}{\partial \mathbf{W}^l}}_{=\mathbf{z}^{l-1}} \frac{\partial \mathrm{J}}{\partial \mathbf{z}^l} \frac{\partial \mathbf{g}\left(\mathbf{z}^{l-1}\right)}{\partial \mathbf{z}^{l-1}} \\ &= \mathbf{z}_i^{l-1} \mathrm{s}^l . \end{aligned} \tag{4.19}$$

The chain rule is effectively implemented in the backpropagation algorithm, which allows to propagate the sensitivities backwards and update the weights through (4.19). The weights $\mathbf{W}^l$ are adjusted in the network through the steepest descent algorithm for the iteration $k+1$.

Remark 6. *Stochastic Gradient Descent (SGD) is one of the most basic and used algorithms, and is quite useful to illustrate backpropagation as well as the use of the sensitivity* $\boldsymbol{s}^l$ *to update the weights on each input presented to the network [3, 52, 63, 64, 123]*

$$\mathrm{w}_{i,j}^{l}\big|_{k+1} = \mathrm{w}_{i,j}^{l}\big|_{k} - \alpha \left.\frac{\partial \mathrm{J}}{\partial \mathrm{w}_{i,j}^{l}}\right|_{k} , \tag{4.20}$$

where α *is the learning ratio.*

4.5 Gaussian Process

An alternative approach to the surrogate modelling of $\mathbf{Y}(\boldsymbol{\omega}, \mathbf{x})$ from (4.1) is Gaussian Processes (GP), a kernel-based method which defines a probability distribution over functions fitting a given dataset. This method has similarities with logistic regression from the probability perspective, and with k-Nearest Neighbors (k-NN) and RBF interpolation as a kernel-based algorithm. Logistic regression methods model the probability of an output in terms of the input, where the output is a set of binary dependent variables with the labels 0 and 1. This is useful for binary and Softmax classification problems where the probability of an outcome is necessary and ultimately not developed for regression or parametric optimisation problems. k-NN is a non-parametric *lazy* learning method, where the training data is stored to make predictions for a new input from interpolation with the nearest samples. This is a very simple model that yields consistent results and is used for a broad class of applications like regression, classification, feature extraction and dimensionality reduction. However, k-NN becomes computationally and memory expensive in the prediction stage for large datasets, a property shared with GP which has a cubic computational complexity [63]. Despite this drawback, the availability of a predictive probability distribution is a compelling advantage that can outweigh the computational cost of GP. Lastly, RBF is discussed in Section 4.3 and the Gaussian RBF is one of the *kernel* functions available in `GPytorch` [132], providing a good benchmark between interpolation and GP modelling.
It is important to note that a GP is a collection of random functions, with any finite sample of which have joint Gaussian distributions $\mathcal{N}$ [4, 61, 62]. This means that the model is completely specified by its mean function $\boldsymbol{\mu}(\boldsymbol{\omega})$ and covariance or *kernel* function $\mathbf{K}(\boldsymbol{\omega}, \boldsymbol{\omega}')$, described for a real process $\mathbf{f}(\boldsymbol{\omega})$ as

$$\boldsymbol{\mu}(\boldsymbol{\omega}) = \mathbb{E}[\mathbf{f}(\boldsymbol{\omega})] , \tag{4.21}$$
$$\mathbf{K}(\boldsymbol{\omega}, \boldsymbol{\omega}') = \mathbb{E}[(\mathbf{f}(\boldsymbol{\omega}) - \boldsymbol{\mu}(\boldsymbol{\omega})) \otimes (\mathbf{f}(\boldsymbol{\omega}') - \boldsymbol{\mu}(\boldsymbol{\omega}'))] , \tag{4.22}$$

Thus, a GP distribution of $\mathbf{f}(\boldsymbol{\omega})$ is expressed as

$$\mathbf{f}(\boldsymbol{\omega}) \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{\omega}), \mathbf{K}(\boldsymbol{\omega}, \boldsymbol{\omega}')) . \tag{4.23}$$

In the model, for every given input $\boldsymbol{\omega}$, there is an associated function output $\mathbf{f}(\boldsymbol{\omega})$ at that location. If two points $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_j$ are close together, they are considered similar by the

kernel, and therefore their respective function outputs $\mathbf{f}(\boldsymbol{\omega}_i)$ and $\mathbf{f}(\boldsymbol{\omega}_j)$ are expected to be close. The GP model then defines a distribution over functions whose shape are defined by the kernel $\mathbf{K}$ [4, 62]. There are a variety of functions available to compute the kernel in `GPyTorch`, yet most of these options are specialised for a specific objective[3]–`CosineKernel`, `CylindricalKernel`, `PeriodicKernel`, `SpectralDeltaKernel`, `SpectralMixtureKernel`– or require extra parameters for which they are highly sensitive to–`PiecewisePolynomial`, `PolynomialKernel`– [68, 132] and will be neglected in this study. Therefore, `LinearKernel`, `RQKernel`, `MaternKernel` and a limiting case of the latter, `RBFKernel` will form the preliminary benchmark of GP models, presented in Appendix D.2.
This first step of the process is considered the *prior*, which specifies the functions' properties and does not depend on training data [62]. In light of having a *prior*, GP models require *a priori* knowledge of the relations between input $\boldsymbol{\omega}$ and output $\mathbf{f}(\boldsymbol{\omega})$ to correctly select the kernel and mean functions.

4.5.1 Posterior of a Gaussian Process

The regression process using GP requires the computation of a posterior joint distribution. Let $\mathbf{f}$ be the known function values from the observed data $\boldsymbol{\omega}$, while $\mathbf{f}_*$ a set of function values at new location points $\boldsymbol{\omega}_*$ from a test set, the joint distribution is written in the form of

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{bmatrix} \right) , \tag{4.24}$$

where $\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{\omega}_i)$, $i = 1, ..., \mathrm{n}_s$ for the training means and analogously for the test means $\boldsymbol{\mu}_*$, and the covariance terms are $\mathbf{K} = (\boldsymbol{\omega}, \boldsymbol{\omega}')$ for training set, $\mathbf{K}_* = (\boldsymbol{\omega}, \boldsymbol{\omega}_*)$ for trainging-test set, and $\mathbf{K}_{**} = (\boldsymbol{\omega}_*, \boldsymbol{\omega}'_*)$ for test set [4, 62]. This allows to compute the conditional distribution of $\mathbf{f}_*$ with a given $\mathbf{f}$ by using the Marginal and Multi-Variate Normal (MVN) conditional distribution [4]

$$\mathbf{f}_* | \mathbf{f} \sim \mathcal{N}\left(\mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{f},\, \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* \right) . \tag{4.25}$$

This is the posterior distribution for any random set of cases $\boldsymbol{\omega}_*$ when a training set $\boldsymbol{\omega}$ is available, and constitutes the central equation for GP predictions. Note that the kernel function is equal to the prior distribution minus a positive term dependent on the training inputs, resulting in the posterior variance always being smaller than the prior variance. Realistically, it is typical to have access to only noisy observations. This issue is addressed through the assumption of additive Gaussian noise in the outputs, which with the Gaussian distribution of (4.23) gives [4, 61, 62, 68]

$$\begin{aligned} \mathbf{y}(\boldsymbol{\omega}) &= \mathbf{f}(\boldsymbol{\omega}) + \boldsymbol{\epsilon} , \\ \boldsymbol{\epsilon} &\sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\sigma_n^2 \right) , \\ \mathbf{y}(\boldsymbol{\omega}) &\sim \left(\boldsymbol{\mu}(\boldsymbol{\omega}) , \mathbf{K}(\boldsymbol{\omega}, \boldsymbol{\omega}') + \sigma_n^2 \boldsymbol{I} \right) , \end{aligned} \tag{4.26}$$

[3] These functions do not fit the form of the flow variable profiles or THP evaluation variables nor the parametric surfaces as presented in Section 6.3.

where σ_n^2 is the variance parameter. The joint distribution from (4.24) can be recomputed to account for noise

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \boldsymbol{I} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{bmatrix} \right), \tag{4.27}$$

while the conditional distribution (4.25) is recomputed as

$$\mathbf{f}_* | \mathbf{y} \sim \mathcal{N}\left(\bar{\mathbf{f}}, \operatorname{cov}(\mathbf{f}_*) \right), \tag{4.28}$$

where

$$\bar{\mathbf{f}} := \mathbb{E}\left[\mathbf{f}_* | \boldsymbol{\omega}, \mathbf{y}, \boldsymbol{\omega}_* \right] = \mathbf{K}_*^T \left(\mathbf{K} + \sigma_n^2 \boldsymbol{I} \right)^{-1} \mathbf{y}, \tag{4.29}$$

$$\operatorname{cov}(\mathbf{f}_*) = \mathbf{K}_{**} - \mathbf{K}_*^T \left(\mathbf{K} + \sigma_n^2 \boldsymbol{I} \right)^{-1} \mathbf{K}_*. \tag{4.30}$$

The variance function $\operatorname{cov}(\mathbf{f}_*)$ exhibit that uncertainty in predictions depends on the input values alone, and not on the observed output $\mathbf{y}$, which is a distinctive characteristic of GP models [4, 61]. As stated before, even in the light of training data and with the addition of noise modelling, enough prior information about this dataset is needed to confidently specify a prior mean and covariance functions.

Remark 7. *The multivariate normal distribution approach is used for systems with multiple correlated feature variables. The distribution function for an* R*-dimensional feature system is defined as*

$$\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \mathbf{K}) = \frac{1}{(2\pi)^{\mathrm{R}/2} \det(\mathbf{K})}^{1/2} \exp\left[-\frac{1}{2} (\boldsymbol{\omega} - \boldsymbol{\mu})^T \mathbf{K}^{-1} (\boldsymbol{\omega} - \boldsymbol{\mu}) \right]. \tag{4.31}$$

The covariance matrix $\mathbf{K}$ *is symmetric and stores the pairwise covariance of all jointly modeled random variables [4].*

Remark 8. *In the case of a single prediction point* $\boldsymbol{\omega}_*$*, denoting the covariance matrix between the test point and the training set as* $\mathbf{k}(\boldsymbol{\omega}_*) = \mathbf{k}_*$*, then equations (4.29) and (4.30) are reduced to*

$$\bar{\mathbf{f}} = \mathbf{k}_*^T \left(\mathbf{k} + \sigma_n^2 \boldsymbol{I} \right)^{-1} \mathbf{y}, \tag{4.32}$$

$$\operatorname{cov}(\mathbf{f}_*) = \mathbf{k}_{**} - \mathbf{k}_*^T \left(\mathbf{k} + \sigma_n^2 \boldsymbol{I} \right)^{-1} \mathbf{k}_*. \tag{4.33}$$

Note that this mean prediction is a linear combination of the observed data $\mathbf{y}$ *with* n_s *snapshots, which can be written in the form*

$$\bar{\mathbf{f}}(\mathbf{x}_*) = \sum_{i=1}^{\mathrm{n}_s} \alpha_i \mathrm{k}_i(\boldsymbol{\omega}_i, \boldsymbol{\omega}_*). \tag{4.34}$$

In this form, called often the linear predictor [61], the coefficient is $\boldsymbol{\alpha} = (\mathbf{k} + \sigma_n^2 \boldsymbol{I})^{-1} \mathbf{y}$*, which makes it resemble the form of an RBF interpolation function (4.10). This formulation is one of the options available in* GPyTorch *[68] to handle multi-output regression models, where each output is an independent model with different kernels and weight coefficients. However, the multi-output* BatchIndependentMultiTaskGPModel*, treats each output as independent models that share the same kernel function. According to* GPyTorch *[68], a batch model reduces the computational expense, since the relation between outputs is not computed, which is advantageous when using POD to pre-process the training data, since POD modes are independent of each other.*

4.5.2 Training and the marginal likelihood

A separate *training* process is required in order to update the mean and covariance functions given the available dataset. These are parametrised in terms of the weight coefficients $\mathbf{W}$, introduced in the form of a *hierarchical prior*, which favour vague prior information assumptions [61, 62]

$$\begin{aligned} \boldsymbol{\mu}(\boldsymbol{\omega}_i) &= \mathbf{W}_{\boldsymbol{\mu}}^T \boldsymbol{\omega}_i + \mathbf{b}\,, \\ \mathbf{K}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_j) &= \mathbf{W}_{\mathbf{K}}^2 \boldsymbol{\omega}_i^T \boldsymbol{\omega}_j + \mathbf{b}\,, \end{aligned} \tag{4.35}$$

where $\mathbf{W}_{\boldsymbol{\mu}}$ and $\mathbf{W}_{\mathrm{K}}$ are introduced to indicate if the hyperparameters belong to the mean or covariance functions. Thanks to this process it is possible to compute the probability of the data given the hyperparameters through the log marginal likelihood $\mathrm{L_J}$ –or evidence–

$$\mathrm{L_J} = \log p(\mathbf{f}|\boldsymbol{\omega}, \mathbf{W}) = -\frac{1}{2}\mathbf{f}^T \mathbf{K}^{-1} \mathbf{f} - \frac{1}{2}\log|\mathbf{K}| - \frac{\mathrm{n}_s}{2}\log 2\pi\,, \tag{4.36}$$

which for noisy observations can be written as

$$\mathrm{L_{J,\sigma}} = \log p(\mathbf{y}|\boldsymbol{\omega}, \mathbf{W}) = -\frac{1}{2}\mathbf{y}^T \left(\mathbf{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \mathbf{y} - \frac{1}{2}\log|\mathbf{K} + \sigma_n^2 \boldsymbol{I}| - \frac{\mathrm{n}_s}{2}\log 2\pi\,. \tag{4.37}$$

A gradient estimate is used to find the values of the hyperparameters which optimise (4.36) and (4.37), similarly than it is done with the mean square error gradient from Section 4.4.2

$$\frac{\partial \mathrm{L_J}}{\partial \mathbf{W}_{\boldsymbol{\mu}}} = -\mathbf{f}^T \mathbf{K}^{-1} \frac{\partial \boldsymbol{\mu}}{\partial \mathbf{W}_{\boldsymbol{\mu}}}\,, \tag{4.38}$$

$$\frac{\partial \mathrm{L_J}}{\partial \mathbf{W}_{\mathbf{K}}} = \frac{1}{2}\mathrm{Tr}\left(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \mathbf{W}_{\mathbf{K}}}\right) + \frac{1}{2}\mathbf{f}^T \frac{\partial \mathbf{K}}{\partial \mathbf{W}_{\mathbf{K}}} \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \mathbf{W}_{\mathbf{K}}} \mathbf{f}\,. \tag{4.39}$$

Note that on both (4.36) and (4.37) the likelihood consists of three terms, two of which are of particular interest. The first is a data-fitting measure, as it is the only training output data dependent, while the second is called a *complexity penalty term*, which measures and penalises the complexity of the model. These two interact in such a way that the tradeoff between penalty and data-fitting is automatic [61, 62] which proves useful to simplify training.

Remark 9. *The term marginal refers to the marginalisation over the function values* $\mathbf{f}$*, which means the model is non-parametric. The equation is computed from the integral of the likelihood times the prior from Bayesian inference [61]*

$$p(\mathbf{y}|\boldsymbol{\omega}) = \int p(\mathbf{y}|\mathbf{f}, \boldsymbol{\omega})\, p(\mathbf{f}|\boldsymbol{\omega})\, \mathrm{d}\mathbf{f}\,. \tag{4.40}$$

Under a set of observations with no noise, the model is purely Gaussian $\mathbf{f}|\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$ *and this equation results in (4.36), while when the likelihood is a factorised Gaussian* $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 \boldsymbol{I})$ *such that noisy observations are accounted for, this will result in (4.37).*

4.6 Conclusion

This chapter introduced the mathematical models of machine learning algorithms used within the context of this work for reduced order modelling of the numerical solutions of fluid flow and heat transfer. The dimensionality reduction problem is addressed using POD, presented in section 4.2, which provides a low-rank approximation of a high-dimensional matrix. In Section 4.3, RBF blueis put forward as the interpolation method that constitutes the basis for the benchmark of low-order approximation models since it is considered highly-accurate. Another statistical model of choice is an NN, presented in Section 4.4, an attractive method compared to other algorithms due to its modular architecture, non-linearity and capability to adjust to high-dimensional data with ease. The last ML algorithm proposed for this project is GP, presented in Section 4.5. The advantages of this model rely on the usage of probability distribution for the approximation, which gives the user an idea of regions where the algorithm needs data points as known examples. This project considers these four surrogate modelling methods relevant for the study of engineering problems within the fields of fluid flow and heat transfer.

POD	
Dimensionality reduction and feature learning method	
Vectorial function $\mathbf{Y}$	$\mathbf{Y}(\boldsymbol{\omega}, \mathbf{x}) \approx \sum_{i=1}^{\mathrm{N}} a_i(\boldsymbol{\omega}) \boldsymbol{\gamma}_i(\mathbf{x})$
Dataset matrix $\mathbf{D}$ representing $\mathbf{Y}$	$\mathbf{D} = [\mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_{\mathrm{N_s}}]^T \approx \mathbf{D}_{\mathrm{N}} = \mathbf{H}_{\mathrm{N}} \boldsymbol{\Sigma}_{\mathrm{N}} \mathbf{G}_{\mathrm{N}}^T$
Low-dimensional $k < \mathrm{N}$ model $\mathbf{P}_{\mathrm{k}}$ of the high-dimension function $\mathbf{Y}$	$\mathbf{P}_{\mathrm{k}} = \mathbf{H}_{\mathrm{k}} \boldsymbol{\Sigma}_{\mathrm{k}}$
RBF	
Kernel based interpolation method	
Linear combination of basis functions with weight coefficients α_{ij} and kernel functions φ_j	$\mathbf{y}(\boldsymbol{\omega}) = \sum_{j=1}^{\mathrm{N_s}} \mathrm{w}_j \varphi(\boldsymbol{\omega})$
NN	
Non-linear statistical learning method	
Forward computation of layer $\mathbf{z}^l$ with hyperparameters $\mathbf{W}^l$ and activation function g_i^l	$\mathbf{z}^l = \mathbf{W}^l \mathbf{g}^l \left(\mathbf{z}^{l-1} + \mathbf{b}\right)$
Sigmoid activation function g_i^l	$\mathrm{g}_i^l(\mathrm{z}) = \frac{1}{1+e^{-\mathrm{z}}}$
GP	
Kernel based predictive probability distribution method	
Distribution of a real process $\mathbf{f}(\boldsymbol{\omega})$ with mean $\boldsymbol{\mu}$ and kernel function $\mathbf{K}$	$\mathbf{f}(\boldsymbol{\omega}) \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{\omega}), \mathbf{K}(\boldsymbol{\omega}, \boldsymbol{\omega}'))$
Conditional distribution of new location $\mathbf{f}_*$	$\mathbf{f}_* \vert \mathbf{f} \sim \mathcal{N}\left(\mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{f},\ \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_*\right)$

Table 4.2: Basic formulation of the methods behind the four relevant ML algorithms reviewed in this Chapter.

CHAPTER 5

Integrated environment for Machine Learning based optimisation

"Do everything by hand, even when using the computer."

— Hayao Miyazaki, 1997

5.1 Introduction

As the world moves towards Generative Pre-trained Transformer (GPT) tools being used for all sorts of tasks, to fully use the potential advantages of surrogate models for Multi-Objective Optimisation (MOO) on shape parametrisation problems, the process has to be approached with an engineering basis. Most of these GPT models use public domain data with no understanding of a-priori knowledge and ill-posed problems, which leads to the model *hallucinating*. From an engineering perspective, the model's ill-posedness leads to wrong results that have an impact on designs and systems' behaviour, which is unwanted when looking for an optimal solution to fluid flow and heat transfer. In this scenario, the better the surrogate model recognises the patterns of the problem from high fidelity data, the more confidence the user has in the results. Therefore, there is a need for a tool that links both stages, the high fidelity model generation and surrogate modelling, with the context of a-priori knowledge to aid in optimising the engineering design process. This thesis puts forward the open-source **Hammerhead software**, designed and developed from scratch as part of this work to fill the need of an easy-to-use robust toolkit for the semi-automation of both high fidelity and surrogate modelling for the MOO problem of Thermo-Hydraulic Performance (THP) in cooling systems. The software is developed in Python and designed to have compatibility with OpenFOAM and Python's Machine Learning (ML) packages. The functions that form Hammerhead as a semi-automated ML-aided optimisation toolkit include shape parametrisation, mesh generation with parametrised surfaces, simulation running on multiple shape and Re parameters, tensor generation from high fidelity data, tensor processing through Singular Value Decomposition (SVD), normalisation and THP evaluation, multiple ML algorithms, surrogate model architecture modularity and benchmark, and shape optimisation through inverse analyses.

5.1.1 Potential impact

Hammerhead's source code is available on GitHub at `https://github.com/Dani-Darko/Hammerhead`. As stated before, this is an open-source, easy-to-use semi-automated toolkit designed and developed within the context of this project to approach the problem of ML-aided THP optimisation in cooling systems. The production of Hammerhead gives rise to potential implications, including:

I I. Open-source semi-automated tool that can be used to expand on the thermal and hydraulic evaluations of flow over surface geometries, which is an active area of research [92], with minor adjustments to the source code and an integrated aid of surrogate models for optimisation at **low user cost**.

I II. Intermediate adjustments to the source code can be made to **adapt Hammerhead to work with other** surface geometries, high fidelity models, simulation software, surrogate modelling algorithms, data processing methods, objective functions and fields of engineering.

I III. Further development can lead to a **generalised ML-aided engineering** toolkit.

This chapter presents the methodology used in Hammerhead to bridge these tasks and automate the workflow of ML-aided optimisation of THP in pipe flows. The main process,

computations and features of the functions mentioned previously are described, and the chart in Figure 5.1 outlines the workflow adopted by Hammerhead. These serve as a reference to provide expertise on the framework that Hammerhead encompasses.

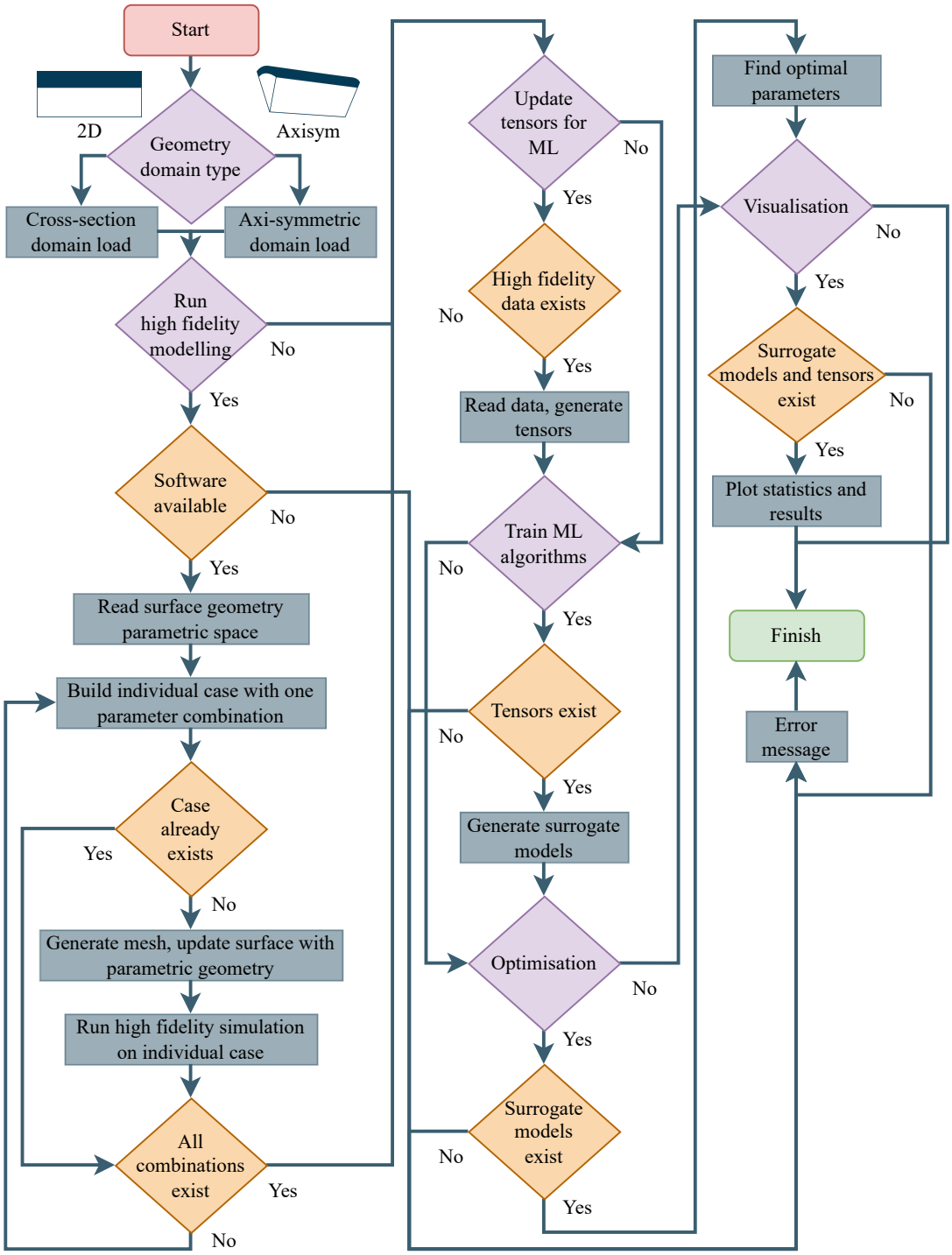


Figure 5.1: Hammerhead flowchart, where software decisions are marked as yellow, user inputs are purple and Hammerhead functions handle the actions in blue.

5.2 Hammerhead software package

The Harmonic MOO Model Expedient for the Reduction Hydraulic-loss/Heat-transfer Enhancement Asset Design (Hammerhead) [133] has been developed as a software package to semi-automate the THP optimisation process of cooling systems based on pipe shape and flow parameters using a mix of high fidelity simulations and surrogate models. This software uses Computational Fluid Dynamics (CFD) data obtained through OpenFOAM v2212, with the opportunity of adding compatible data formats. Hammerhead is written in Python 3, and uses `PyQt5` libraries for its user interface, with the added capability to run on console and cluster environments. A combination of `NumPy` and `PyTorch` is used for the computationally expensive calculations, while most numerical data is stored using fixed data-type `PyTorch` tensors to minimise memory footprint and computation time. Hammerhead's functions can be performed fully independent of each other, as long as the information required for each is available. The following is a list of Hammerhead's functions and a brief description of each step:

- **High fidelity database population**: A high fidelity model database is built through OpenFOAM's `chtMultiRegionSimpleFoam` solver with user input parameters for the flow's *Re* and the pipe's surface harmonic shapes.
- **Tensor data update**: From the data sample built with the high fidelity database population, tensors are updated and stored with relevant data for the surrogate problem – shape and flow features, THP, flow variables, POD modes, amongst others–.
- **Surrogate model training**: ML algorithms are trained with the tensor data stored based on the user's selected algorithm and training parameters –these include the validation split, number of random initialisations, neurons and layers for Neural Networks (NN), and kernels for Radial Basis Functions (RBF) and Gaussian Processes (GP)–.
- **Optimal parameters' search**: The optimisation process uses the user's selected surrogate models as a function to find a *global minima* within the parameter space bounds through `scipy.optimize.differential_evolution` tool.
- **Plotting and data visualisation**: Data analysis is performed through diverse plots available related to the surrogate models' architecture benchmark, surrogate model's Mean Square Error (MSE) based performance, optimisation process search, flow variables and THP.

The command to start Hammerhead in console mode is [133]

```
python ./src/Hammerhead/hammerhead.py --domain 2D --console
```

relative to the project's root directory, where `--domain` is a required parameter with `[2D, axysim]` options available. Two tasks run by default when Hammerhead is launched in console mode, HFM database population and tensor data update, the remaining functions have to be manually invoked. To avoid either of these default tasks, supply the arguments `--noHFM` and/or `--noTensor`. All computations –excluding surrogate model training– are parallelised. To select the number of processes, the argument `--nProc` can be supplied, which

will default to the maximum number of available cores if unspecified. The parallelisation is handled by Python's `multiprocessing` library. Hammerhead can also be started with a user interface, if the argument `--console` is neglected. The main window of Hammerhead is presented in Figure 5.2, where the three buttons on the left represent the high fidelity data population, surrogate model training and plotting data and visualisation tools. The rest of Hammerhead functions are embeded within these. The two buttons on the right represent the domain selection, 2D or axi-symmetric, and *run* functions, which will start the process presented in the flowchart from Figure 5.1.

Figure 5.2: Hammerhead software main window.

system/	constant/	0/
↪ Region/	↪ Region/	↪ p
↪ changeDictionaryDict	↪ thermoPhysicalProperties	↪ p_rgh
↪ fvSchemes	↪ turbulenceProperties	↪ T
↪ fvSolution	↪ polyMesh/	↪ U
↪ blockMeshDict	↪ boundary	↪ turulence related fields
↪ controlDict	↪ cellZones	↪ field files
↪ topoSetDict	↪ faces	
	↪ faceZones	
	↪ meshModifiers	
	↪ neighbour	
	↪ owner	
	↪ points	
	↪ pointZones	

Table 5.1: OpenFOAM's `chtMultiRegionSimpleFoam` directory configuration and file names. The `Region` directories are named after the specified region – `Helium` and `topWall` in this case– when running the topology setting file `topoSetDict`.

5.3 Database population

As stated before, the first step performed by Hammerhead is the high fidelity model shape parameter space database population. OpenFOAM is the only current solver compatible with Hammerhead for the high fidelity database population. This software uses a basic directory structure containing the minimum set of files required to run a simulation, which makes it portable and modular. The file structure shown in Table 3.2 is particular to the application of choice, `chtMultiRegionSimpleFoam`, used to construct the mesh, compute the initial and boundary conditions, and run the simulation within an OpenFOAM environment. Features may change from OpenFOAM version to version, where compatibility is currently verified for v2106 and v2212. The computation of high fidelity models in Hammerhead uses either one of two domain options available for the user: a two-dimensional pipe cross-section or an axisymmetric section *wedge*, as shown in Figure 5.3. The baseline case setup directories for either domain choice are stored in `$HHBASE/resources/base2DFiles/` and `$HHBASE/resources/baseAxisymFiles/`, where `$HHBASE` is the directory containing Hammerhead's main file `hammerhead.py`. These directories are preloaded with initial and boundary conditions from Figure 5.3, as well as the thermo-physical properties described in Table 6.5, excluding geometry parameters and Re, which are treated as user inputs. With

the shape related user inputs, Hammerhead computes each surface shape within the parameter space as a double harmonic function for either the two dimensional flat plate or the axisymmetric problem

$$r\left(x\right) = r_0 + \sum_{i=1}^{\mathrm{N_h}} \mathrm{A}_i \left(\cos\left(\pi \mathrm{k}_i x\right) + 1\right) , \tag{5.1}$$

where r_0 is the baseline pipe's radius, and A_i, $\mathrm{k}_i = 2\pi/\lambda$ the harmonic's $\mathrm{N_h}$ amplitude and wavenumber, respectively, and λ is the wavelength. Currently, Hammerhead's maximum harmonics $\mathrm{N_h}$ are 2. These parameters shape the surface as presented in Figure 5.4. Regarding the flow parameters, Hammerhead updates the velocity profile based on the user inputs, with the *Re* distribution as the flow parameter space. These inputs can be edited in the file `$HHBASE/resources/hfmParams.yaml` as in Listing 5.1.

```
#  I, mu, L, r and dx do not edit OpenFOAM files, used for Hammerhead
  computations
I: 0.05         # Turbulence intensity. Units: dimensionless, for Re > 2000.
mu: 0.001       # Dynamic viscosity. Units: pressure time [Pa s].
L: 2.0          # Pipe length. Units: length [m].
r: 0.2          # Pipe radius. Units: length [m].
dx: 900         # Element count in x.
# Re is used to update velocity profile and populate cases, the numbers
  displayed are default
Re_min: 1000  # Minimum Re. Units: dimensionless.
```

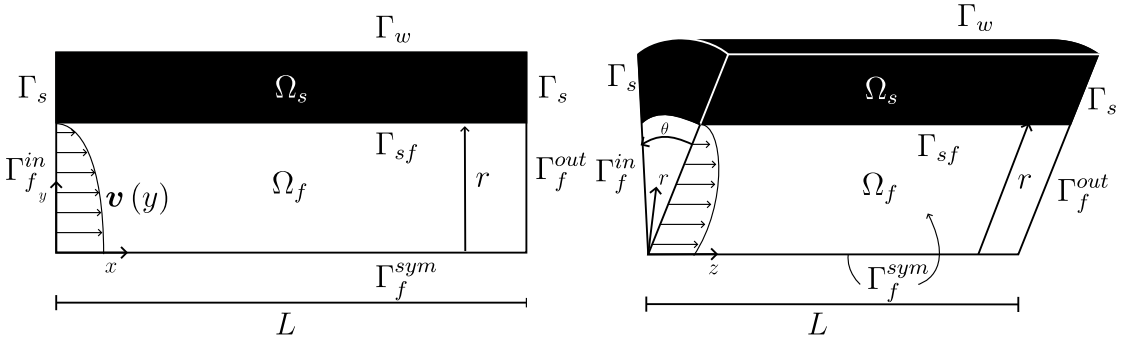


Figure 5.3: Domains of internal flow in a pipe represented by –left– 2D flow over a *smooth* surface and –right– a wedge of axi-symmetric flow over *smooth* surface.

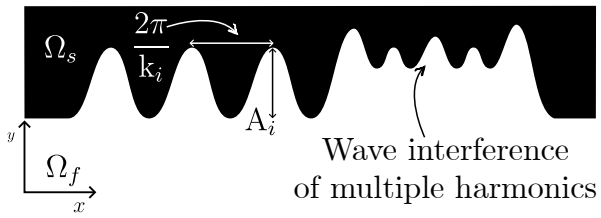


Figure 5.4: Stream-wise section of surface roughness resulting from (5.1). The interaction of 2 harmonics leads to a wave interference where the overlap can be constructive or destructive.

```
Re_max: 4000  # Maximum Re. Units: dimensionless.
Re_num: 4     # Number of evenly distributed $Re$ samples to use between
 Re_min and Re_max.
# The shape parameters for mesh generation, the numbers displayed are
 default
A1_min: 0.0   # Minimum main amplitude. Units: length [m].
A1_max: 0.01  # Maximum main amplitude. Units: length [m].
A1_num: 4     # Number of evenly distributed samples to use between A1_min
  and A1_max
k1_min: 0.0   # Minimum main wavenumber. Units: reciprocal length [m-1].
k1_max: 60.0  # Maximum main wavenumber. Units: reciprocal length [m-1].
k1_num: 4     # Number of evenly distributed samples to use between k1_min
  and k1_max.
A2_min: 0.0   # Minimum second amplitude. Units: length [m].
A2_max: 0.01  # Maximum second amplitude. Units: length [m].
A2_num: 4     # Number of evenly distributed samples to use between A2_min
  and A2_max.
k2_min: 0.0   # Minimum second wavenumber. Units: reciprocal length [m-1].
k2_max: 60.0  # Maximum second wavenumber. Units: reciprocal length [m-1].
k2_num: 4     # Number of evenly distributed samples to use between k2_min
  and k2_max.
```

Listing 5.1: User input parameters needed for flow and shape update computations in Hammerhead.

The high fidelity data population window from Figure 5.5 within the user interface, shows the shapes each parameter combination will make in the surface geometry, and the parameters can be edited here. When launching console mode, this information can be overridden through parameters of the same names like `--Re_min --Re_max --Re_num ...` [133]. After computing the shape and updating flow parameters for each unique case, the baseline directory structure `$HHBASE/resources/base*/` is copied to `$HHBASE/caseDatabase/` –including a subdirectory named after the selected domain, `2D` or `axisym` – with a new name system collecting all the relevant parameters for easy identification. For example, computing the case of $Re = 500$, $A_1 = 0.005$ m, $k_1 = 12.0$ m^{-1}, $A_2 = 0.002$ m, $k_2 = 60.0$ m^{-1} within the two dimensional flat plate, the new directory will be `$HHBASE/caseDatabase/2D/Re_500_A_0-005_0-002_k_12-0_60-0/` . This directory will contain changes in the velocity field within OpenFOAM file `./system/Helium/changeDictionaryDict` , relative to the baseline directory, to match the computed profile from the definition of Re, as well as added content to OpenFOAM file `./system/blockMeshDict` that fit the mesh's surface edges matching the pipe's surface to the corresponding harmonic shape computed from (5.1). When the new directory is ready with the updated files, the mesh is computed, topology is set and OpenFOAM `chtMultiRegionSimpleFoam` starts running on each individual unique case.

Remark 10. *Note that the computations are performed for each unique case, which implies ignoring the redundant cases, where the surface is the same as another previously established case. Hammerhead identifies all the unique cases to run, populates these and the information available after finishing the unique simulations is referred to by the duplicate case directories.*

Particularly for the baseline cases, where `A_0-0_0-0_k_0-0_0-0`*, Hammerhead will create the 0-0 while ignoring all other duplicates.*

OpenFOAM has different data formats depending on the package version and sampling choice. The high fidelity modelling format output employed by Hammerhead includes the OpenFOAM v2212 Sampling tool, for `cellPoint` data with no interpolation, of temperature, pressure and velocity variables, in the form of

$y-$ in 2D and $z-$ in Axisym–	T	p	v_x	v_y	v_z

This enables Hammerhead to have compatibility –or easily made compatible– with any file with such format. The file extension form is `.xy`. If another format or OpenFOAM version is being used, this can be added to the file `$HHBASE/resources/samplingFormats.yaml` to comply with Hammerhead compatibility, as shown next:

```
# As different versions of OpenFOAM use different output formatting, here
 we specify the files~(and corresponding columns in files) to read for
 each OpenFOAM version
# checkFor -> if the specified file is found, use this OpenFOAM version
# inlet*/outlet* -> first entry in list is the file name to open, second
 entry is the column to read
v2212:
    checkFor: inlet_T_p_U.xy
    inletU:   [inlet_T_p_U.xy,   3]
    inletT:   [inlet_T_p_U.xy,   1]
    inletp:   [inlet_T_p_U.xy,   2]
    outletU:  [outlet4_T_p_U.xy, 3]
    outletT:  [outlet4_T_p_U.xy, 1]
    outletp:  [outlet4_T_p_U.xy, 2]
v2106:
    checkFor: inlet_T_p.xy
    inletU:   [inlet_U.xy,     1]
    inletT:   [inlet_T_p.xy,   1]
    inletp:   [inlet_T_p.xy,   2]
    outletU:  [outlet4_U.xy,   1]
    outletT:  [outlet4_T_p.xy, 1]
    outletp:  [outlet4_T_p.xy, 2]
```

Listing 5.2: Sampling formats compatible with Hammerhead.

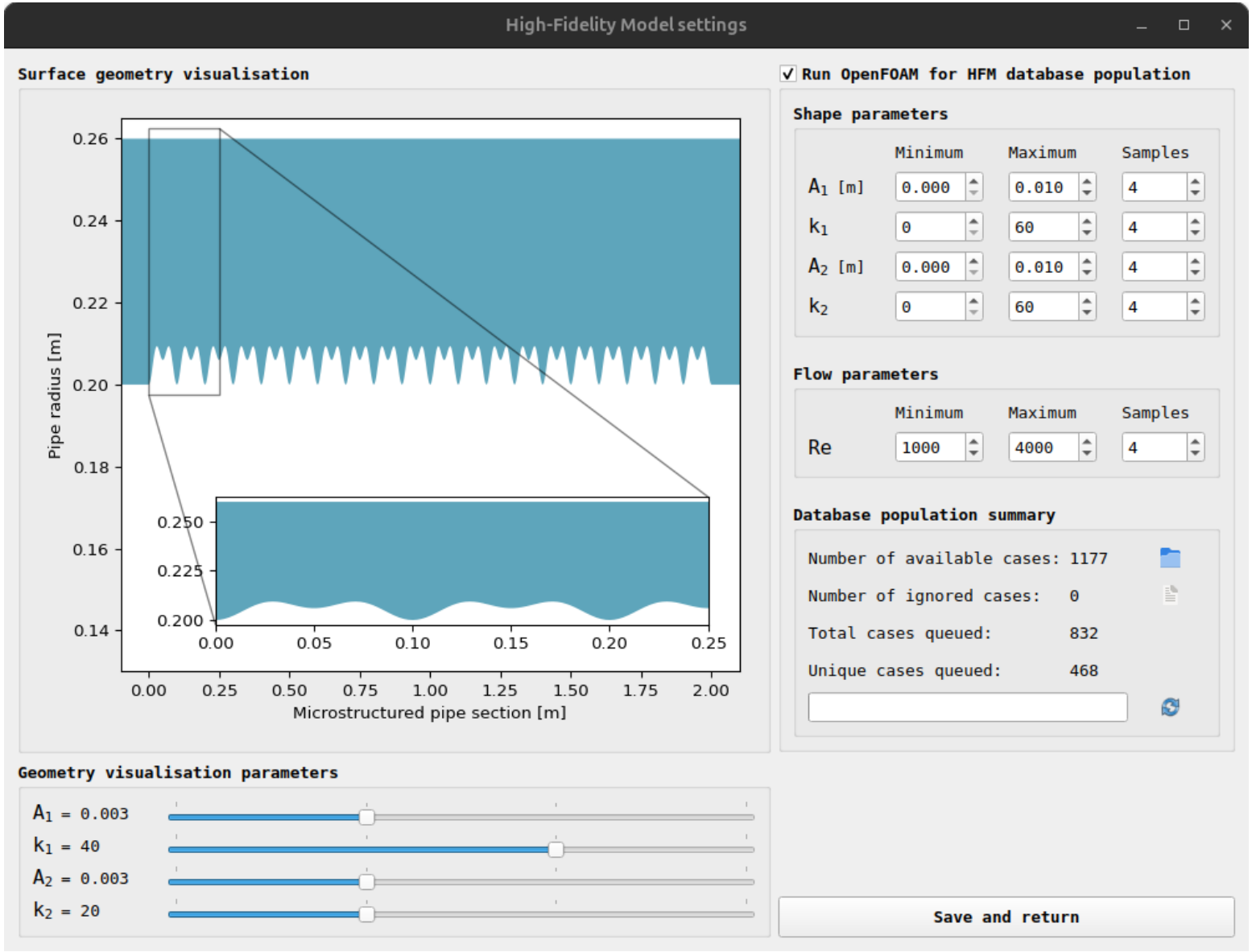


Figure 5.5: Hammerhead software high fidelity data population window.

5.3.1 Meshing process automation

A particularly interesting feature of Hammerhead is how it deals with mesh generation for this particular shape parametrisation problem. As described in Section 5.3, baseline case setup directories are included in Hammerhead resources, since all cases share most parameters. The only changes made to each new individual case are related to the harmonic shape of the pipe's surface as portrayed in Figure 5.3, and the velocity profile related to Re. The domain will be discretised from the geometry in Figure 5.3 to the mesh shown in the example from Figure 5.6. The first are mesh-relevant parameters, as they form the shape that the pipe's surface –and by proxy the mesh edges– will be fitted to. From Section 5.3 we gather that the mesh is updated through OpenFOAM's `blockMesh` utility file `./system/blockMeshDict`, relative to the baseline directory. This utility generates parametric, fully structured mesh blocks from vertex coordinates –8 per block, multiple blocks can be added–. Each new vertex added is labelled in ascending order, with Hammerhead default files including the vertices and blocks shown in Listings 5.3 and 5.4. The collection of two connecting vertices forms a straight block edge –7 edges per block–, identified by the connecting vertices labels. Since these edges are straight by default, Hammerhead uses the points computed from (5.1) as a list of coordinates to fit the edges to, which are added to the `edges` section of the file `./system/blockMeshDict`. This process is exemplified in 5.5 and 5.6 with the corresponding vertex labels for the fitted edge.

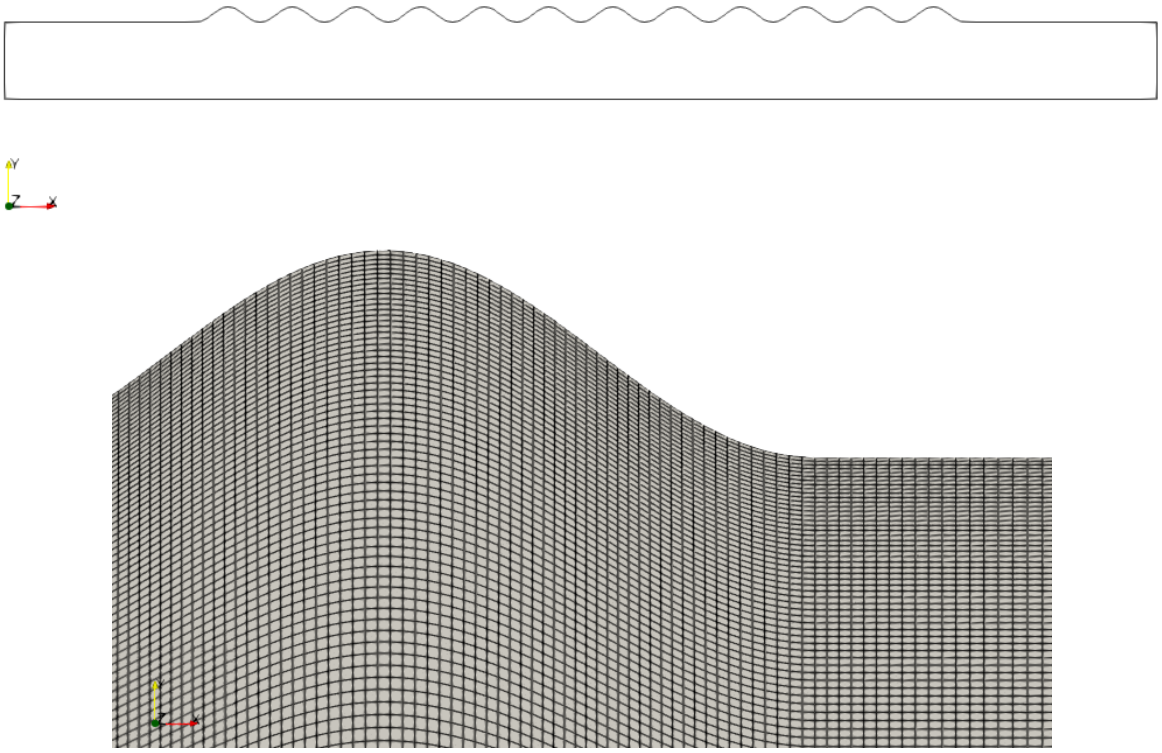


Figure 5.6: Example of domain discretisation and close-up of element size in the near-wall region close to the exit of the roughness area.

Added to this challenge, the existence of two regions for `chtMultiRegionSimpleFoam`'s conjugate heat transfer requires setting their topology for the simulation. Due to OpenFOAM's approach to this process, each new geometry fitted to the mesh impedes a correct topology setting. However, since the fitted meshes share IDs on faces, cell points and boundaries, the file `cellZones` –which stores the region topology information– can be inherited from the baseline case.

```
vertices
(
  (0 0 0) //0
  (2 0 0) //1
  (2 0.2 0) //2
  (0 0.2 0) //3
  (0 0 0.1) //4
  (2 0 0.1) //5
  (2 0.2 0.1) //6
  (0 0.2 0.1) //7
  (0 0.2 0) //8
  (2 0.2 0) //9
  (2 0.26 0) //10
  (0 0.26 0) //11
  (0 0.2 0.1) //12
  (2 0.2 0.1) //13
  (2 0.26 0.1) //14
  (0 0.26 0.1) //15
  (-0.5 0 0) //16
  (2.5 0 0) //17
  (2.5 0.2 0) //18
  (-0.5 0.2 0) //19
  (-0.5 0 0.1) //20
  (2.5 0 0.1) //21
  (2.5 0.2 0.1) //22
  (-0.5 0.2 0.1) //23
  (-0.5 0.2 0) //24
  (-0.5 0.26 0) //25
  (-0.5 0.2 0.1) //26
  (-0.5 0.26 0.1) //27
  (2.5 0.2 0) //28
  (2.5 0.26 0) //29
  (2.5 0.2 0.1) //30
  (2.5 0.26 0.1) //31
);

blocks
(
  hex (0 1 2 3 4 5 6 7) (900 100 1) simpleGrading (1 -6 1)
  hex (8 9 10 11 12 13 14 15) (900 10 1) simpleGrading (1 1 1)
  hex (16 0 3 19 20 4 7 23)  (100 100 1) simpleGrading (-4 -6 1)
  hex (1 17 18 2 5 21 22 6)  (100 100 1) simpleGrading (4 -6 1)
  hex (24 8 11 25 26 12 15 27) (100 10 1) simpleGrading (-4 1 1)
  hex (9 28 29 10 13 30 31 14) (100 10 1) simpleGrading (4 1 1)
);
```

Listing 5.3: Vertices and blocks for the two dimensional flat plate mesh blocks.

```
halfAngle 10.0; //- Half angle of axysimetric "wedge". Units: plane angle.
radius  0.2;    //- Radius of pipe inner surface. Units: length [m]
radius2 0.26;   //- Radius of pipe outer surface. Units: length [m]
y               #eval{ $radius*sin(degToRad($halfAngle)) };
z               #eval{ $radius*cos(degToRad($halfAngle)) };
minY            #eval{ -1*$y };
minZ            #eval{ -1*$z };
yWall           #eval{ $radius2*sin(degToRad($halfAngle)) };
zWall           #eval{ $radius2*cos(degToRad($halfAngle)) };
minYwall        #eval{ -1*$yWall };
minZwall        #eval{ -1*$zWall };

vertices
(
  (0 0 0)         //0
  (2 0 0)         //1
  (2 0 0)         //2
  (0  0 0)         //3
  (0  $minY $z)   //4
  (2 $minY $z)    //5
  (2 $y $z)       //6
  (0  $y $z)       //7
  (0  $minY $z)   //8
  (2 $minY $z)    //9
  (2 $y $z)       //10
  (0  $y $z)       //11
  (0  $minYwall $zWall)   //12
  (2 $minYwall $zWall)   //13
  (2 $yWall $zWall)       //14
  (0  $yWall $zWall)       //15
  (-0.5  0 0)         //16
  (-0.5  0 0)         //17
  (-0.5  $minY $z)    //18
  (-0.5  $y $z)       //19
  (2.5 0 0)         //20
  (2.5 0 0)         //21
  (2.5 $minY $z)    //22
  (2.5 $y $z)       //23
  (-0.5  $minY $z)         //24
  (-0.5  $y $z)         //25
  (-0.5  $minYwall $zWall)   //26
  (-0.5  $yWall $zWall)       //27
  (2.5 $minY $z)         //28
  (2.5 $y $z)         //29
  (2.5 $minYwall $zWall)   //30
  (2.5 $yWall $zWall)       //31
);

blocks
(
  hex (0 1 2 3 4 5 6 7) (650 8 100) simpleGrading (1 1 -6)
  hex (8 9 10 11 12 13 14 15) (650 8 10) simpleGrading (1 1 6)
  hex (16 0 3 17 18 4 7 19) (100 8 100) simpleGrading (-4 1 -6)
```

```
  hex (1 20 21 2 5 22 23 6) (100 8 100) simpleGrading (4 1 -6)
  hex (24 8 11 25 26 12 15 27) (100 8 10) simpleGrading (-4 1 6)
  hex (9 28 29 10 13 30 31 14) (100 8 10) simpleGrading (4 1 6)
);
```

Listing 5.4: Vertices and blocks for the axisymmetric section mesh blocks.

```
edges
(
  polyLine 3 2
  ((x y z)
  (x y z)
  ...)
  polyLine 7 6
  ((x y z)
  (x y z)
  ...)
  polyLine 8 9
  ((x y z)
  (x y z)
  ...)
  polyLine 12 13
  ((x y z)
  (x y z)
  ...)
);
```

Listing 5.5: Edge curving example for the two dimensional flat plate mesh blocks.

```
edges
(
  polyLine 4 5
  ((x y z)
  (x y z)
  ...)
  polyLine 7 6
  ((x y z)
  (x y z)
  ...)
  polyLine 8 9
  ((x y z)
  (x y z)
  ...)
  polyLine 11 10
  ((x y z)
  (x y z)
  ...)
);
```

Listing 5.6: Edge curving example for the axisymmetric section mesh blocks.

5.4 Data processing for surrogate models

To make the high fidelity data suitable for surrogate model usage, Hammerhead processes it into `PyTorch` tensors format. These are multi-dimensional matrices containing –in the case of Hammerhead– 32-bit floating point data type. To do so, after `$HHBASE/caseDatabase/` is available, Hammerhead works with two sets of data, input features and output results. In terms of high fidelity modelling, the first refers to the shape and flow parameters, where each set of features refers to a case or *snapshot* $\mathrm{N_s}$ in the context of ML. The parameters are the Harmonic variables A_i and k_i, along with Re, inferred from the case directory name for all snapshots to be stored in the same tensor. This results in a tensor $\mathrm{N_s} \times \omega_i$ with snapshot rows and feature columns. The treatment of this tensor depends on two factors, the Harmonics level and the usage of Re as an input feature. The sets of features evaluated in this study are

$$\boldsymbol{\omega}_2 = [\mathrm{A}_1, \mathrm{k}_1]\,, \qquad \text{when } Re = constant,\ \mathrm{N_h} = 1\,, \tag{5.2a}$$
$$\boldsymbol{\omega}_3 = [\mathrm{A}_1, \mathrm{k}_1, Re]\,, \qquad \text{when } \mathrm{N_h} = 1\,, \tag{5.2b}$$
$$\boldsymbol{\omega}_4 = [\mathrm{A}_1, \mathrm{k}_1, \mathrm{A}_2, \mathrm{k}_2]\,, \qquad \text{when } Re = constant,\ \mathrm{N_h} = 2\,, \tag{5.2c}$$
$$\boldsymbol{\omega}_5 = [\mathrm{A}_1, \mathrm{k}_1, \mathrm{A}_2, \mathrm{k}_2, Re]\,, \qquad \text{when } \mathrm{N_h} = 2\,. \tag{5.2d}$$

The output results are associated with one of three different tensors, depending on the processing of the data. The tensors are (a) flow variable inlet and outlet profiles called *spatial*, (b) SVD reduction of the latter named *modal*, and (c) THP evaluation functions referred to as *lumped*. These are

$$\mathbf{x}^{\dot{Q}} = [f_f, f_h]\,, \qquad \text{then } \mathrm{N}_x = 1\,, \tag{5.3a}$$
$$\mathbf{x}^{\Gamma_f} = \left[{v_x}^{in}, T^{in}, p^{in}, {v_x}^{out}, T^{out}, p^{out}\right], \quad \text{then } \mathrm{N}_x = 100\,, \tag{5.3b}$$
$$\mathbf{x}^{\mathbf{P}} = \left[\mathbf{P}^{in}_{v_x}, \mathbf{P}^{in}_{T}, \mathbf{P}^{in}_{p}, \mathbf{P}^{out}_{v_x}, \mathbf{P}^{out}_{T}, \mathbf{P}^{out}_{p}\right], \text{then } \mathrm{N}_x = 1 - 100\,, \tag{5.3c}$$

where $\mathbf{P}$ are the POD modes matrix $\mathbf{D}$ for each flow variable, computed through SVD in Section 7.2. The proposed $\mathbf{x}^{\dot{Q}}$ or *lumped* (L-) output is the smallest with $\mathrm{N}_x = 1$, based on the dissipation rate f_f and advected heat flux f_h from (2.52) and (2.54) –two surrogate models result from this output–. The output $\mathbf{x}^{\Gamma_f}$ or *spatial* (S-) has $\mathrm{N}_x = 100$ data points based on the flow variable's mesh inlet and outlet boundary values –six surrogate models result from this output, one for each variable–. The last dimension considered is $\mathbf{x}^{\mathbf{P}}$ or *modal* (M-) output, which reduces the complexity of the flow variable profiles, design to predict for the most prominent *modes* of each flow variable to compute profiles and THP mechanisms in the post-processing. The diversity of tensors creates the necessity of storing the relevant data into different directories, with the form `$HHBASE/mlData/*/Re_*_modes_*_harmonics_*/`, where the $*$ are filled with the information corresponding to the selected domain, the shape and flow features, as well as the *modes* computed from (4.3). The tensors are stored independently and regardless of the user's choice, to provide easy access and reduce the online computational cost. The output tensor selection for surrogate model training is left to the user, and will be discussed in Section 5.5.

To start the process, the data from the default OpenFOAM v2212 sampling files `inlet_T_p_U.xy` and `outlet4_T_p_U.xy` is loaded into Hammerhead. In these, the variable

columns are extracted as a tensor within a dictionary, where each variable is a key independent of each other. Similarly to the input features tensor, the process occurs for all different shape parameter snapshots, stored in the corresponding variable key's tensor. This results in a tensor per flow variable: $\mathrm{N_s} \times v_x$ inlet, $\mathrm{N_s} \times T$ inlet, $\mathrm{N_s} \times p$ inlet, $\mathrm{N_s} \times v_x$ outlet, $\mathrm{N_s} \times T$ outlet and $\mathrm{N_s} \times p$ outlet, with the dictionary including 6 keys, one per variable tensor. This dictionary makes the spatial output result, which is processed to obtain the other two. For the modal tensor, SVD is used to compute a low-rank matrix, truncating each variable tensor to the user input modes σ_N from the file `$HHBASE/resources/trainingParams.yaml` as described in Chapter 4.2. The low-rank matrix $\mathbf{P}_\mathrm{N} \in \mathbb{R}^{\mathrm{N_s}} \times \mathbb{R}^{\mathrm{N}}$ represents the POD coefficients or POD modes, the columns of $\mathbf{H}_\mathrm{N} \in \mathbb{R}^{\mathrm{N_s}} \times \mathbb{R}^{\mathrm{N_s}}$ represent the left singular vectors and $\mathbf{\Sigma}_\mathrm{N} \in \mathbb{R}^{\mathrm{N_s}} \times \mathbb{R}^{\mathrm{N}}$ the tessellated singular values. The tensor $\mathrm{N_s} \times \mathbf{P}_\mathrm{N}$ is created for each flow variable, with the dictionary retaining 6 keys, one per variable tensor. Regarding the lumped tensor, the flow variables from the spatial dictionary tensors are used to compute dissipation rate and advected heat flux from (2.52) and (2.54), respectively, which become the dictionary keys. The dictionary for these tensors has 2 keys, with tensors $\mathrm{N_s} \times f_f$ and $\mathrm{N_s} \times f_h$. The tensors within these three output results dictionaries, *spatial*, *modal* and *lumped*, along with the input feature tensors, are in various magnitudes and scales, thus need normalisation for the surrogate models. This is done independently for each feature in the input feature tensors, and for each variable measurement point in the output results tensors, normalising along the snapshot axis with

$$\omega_{j,normalised} = \frac{\omega_j - \omega_{j,min}}{\omega_{j,max} - \omega_{j,min}}, \tag{5.4a}$$

$$D_{j,normalised} = \frac{\mathrm{D}_j - \mathrm{D}_{j,min}}{\mathrm{D}_{j,max} - \mathrm{D}_{j,min}}, \tag{5.4b}$$

where the subscript j refers to the column of each output results tensor $\mathbf{D}$ and each feature in input features tensor $\boldsymbol{\omega}$. Then, the tensor dictionaries storage include the files with format {`variable key: [tensor rows, tensor columns]`} with format as follows:

```
"xData.pt"                # Dictionary containing input feature tensors,
 normalised and denormalised, with maximum and minimum values for
 normalisation purposes
    {
     x:          [snapshots, features]
     xExpanded:  [snapshots, features]
     xMax:       [1, features]
     xMin:       [1, features]
    }
"lumpedData.pt"           # Dictionary containing normalised lumped
 tensors, dissipation rate and advected heat flux
    {
     lumpedp: [snapshots, 1],
     lumpedT: [snapshots, 1]
    }
"lumpedDataExpanded.pt"   # Dictionary containing denormalised lumped
 tensors
    {
     lumpedp: [snapshots, 1],
     lumpedT: [snapshots, 1]
```

```
    }
"lumpedMax.pt"            # Dicitionary containing maximum values of
 lumped tensors, for normalisation purposes
    {
     lumpedp: [1, 1],
     lumpedT: [1, 1]
    }
"lumpedMin.pt"            # Dicitionary containing minimum values of
 lumped tensors, for normalisation purposes
    {
     lumpedp: [1, 1],
     lumpedT: [1, 1]
    }
"modalData.pt"            # Dictionary containing normalised modal tensors
  for each SVD processed flow variable, inlet and outlet velocity,
 temperature and pressure
    {
     inletU:  [snapshots, modes],
     inletT:  [snapshots, modes],
     inletp:  [snapshots, modes],
     outletU: [snapshots, modes],
     outletT: [snapshots, modes],
     outletp: [snapshots, modes]
    }
"modalMax.pt"             # Dicitionary containing maximum values of modal
  tensors, for normalisation purposes
    {
     inletU:  [1, modes],
     inletT:  [1, modes],
     inletp:  [1, modes],
     outletU: [1, modes],
     outletT: [1, modes],
     outletp: [1, modes]
    }
"modalMin.pt"             # Dicitionary containing minimum values of modal
  tensors, for normalisation purposes
    {
     inletU:  [1, modes],
     inletT:  [1, modes],
     inletp:  [1, modes],
     outletU: [1, modes],
     outletT: [1, modes],
     outletp: [1, modes]
    }
"spatialData.pt"          # Dictionary containing normalised spatial
 tensors for each flow variable, inlet and outlet velocity, temperature
 and pressure
    {
     inletU:  [snapshots, 100],
     inletT:  [snapshots, 100],
     inletp:  [snapshots, 100],
     outletU: [snapshots, 100],
     outletT: [snapshots, 100],
```

```
      outletp: [snapshots, 100]
    }
"spatialDataExpanded.pt"  # Dictionary containing denormalised spatial
 tensors
    {
     inletU:  [snapshots, 100],
     inletT:  [snapshots, 100],
     inletp:  [snapshots, 100],
     outletU: [snapshots, 100],
     outletT: [snapshots, 100],
     outletp: [snapshots, 100]
    }
"spatialDataMax.pt"       # Dicitionary containing maximum values of
 spatial tensors, for normalisation purposes
    {
     inletU:  [1, 100],
     inletT:  [1, 100],
     inletp:  [1, 100],
     outletU: [1, 100],
     outletT: [1, 100],
     outletp: [1, 100]
    }
"spatialDataMin.pt"       # Dicitionary containing minimum values of
 spatial tensors, for normalisation purposes
    {
     inletU:  [1, 100],
     inletT:  [1, 100],
     inletp:  [1, 100],
     outletU: [1, 100],
     outletT: [1, 100],
     outletp: [1, 100]
    }
"VTReduced.pt"            # Dicitionary containing G^T truncated right
 singular vectors of modal tensors, for SVD low-rank matrix calculation
 purposes
    {
     inletU:  [modes, 100],
     inletT:  [modes, 100],
     inletp:  [modes, 100],
     outletU: [modes, 100],
     outletT: [modes, 100],
     outletp: [modes, 100]
    }
```

Listing 5.7: Tensor files, dictionary keys and tensor dimensions computed by Hammerhead.

5.5 Surrogate model training

Hammerhead uses `SciPy`, `PyTorch` and `GPyTorch` packages to train respectively the Radial Basis Function interpolation (RBF), Neural Networks (NN) and Gaussian Processes (GP) as surrogate models. `SciPy` contains an optimised RBF function with various kernel options,

helpful in Hammerhead for the architecture modularity. In the case of `PyTorch` and `GPyTorch`, training and architecture is fully modular and built by the user. Hammerhead has a built-in base architecture for both NN and GP, with a basic training process, both explained in this section.
The user has several options to choose from based on the output results form and the ML algorithm to train. As stated in Section 5.4, the output results tensors available are *spatial* (S-), *modal* (M-) and *lumped* (L-), and with the Hammerhead compatible ML algorithms, the combination yields the next training collection In console, the argument for training is [133]

	RBF	NN	GP
L-	LRBF	LNN	LGP
S-	SRBF	SNN	SGP
M-	MRBF	MNN	MGP

Table 5.2: Collection of models available to the user in Hammerhead surrogate model training. For Hammerhead, the "-" could interfere with other arguments in console, therefore is dropped out of the call.

```
python ./src/Hammerhead/hammerhead.py --domain 2D --console --train
```

relative to the project's root directory, followed by either of the options in Table 5.2, like `--train SNN`. The short-form argument `-t` can also be used. When selecting the training mode, the process will occur for every directory of the form `$HHBASE/mlData/*/Re_*_modes_*_harmonics_*/`, as long as the tensors exist within it. This means that all the input feature collections $-\boldsymbol{\omega}_2$, $\boldsymbol{\omega}_3$, $\boldsymbol{\omega}_4$ and $\boldsymbol{\omega}_5-$ will be trained. The architecture of each model can be manipulated in the file `$HHBASE/resources/trainingParams.yaml` with format as follows:

```
layers: [1, 2, 3, 4]                          # Number of hidden layers for
  NN architecture
modes: 5                                      # Number of PCA modes to
 train with
neurons: [1, 2, 4, 8, 16, 32, 64, 128, 256]   # Number of neurons per layer
  for NN architecture
kernelsGP: [MaternKernel]                     # Type of kernel to use for
 GP, see gpytorch.kernels documentation [MaternKernel, RBFKernel, RQKernel
 , ...]
kernelsRBF: [inverse_multiquadric]            # Type of kernel to use for
 RBF, see scipy.interpolate.RBFInterpolator "kernel" parameter
samples: 3                                    # Number of times training is
  performed (each time with unique random hyperparameter initialisation)
validationSplit: [0.35]                       # Ratio of data used for
 validation (0.35 = 35% used for validation, 65% used for training)
```

Listing 5.8: User input parameters needed for surrogate model architecture and training in Hammerhead.

where `layers` and `neurons` are NN relevant, `kernelsGP` are for GP and `kernelsRBF` apply to RBF. Regarding the rest of parameters, they are pertinent to every ML; `modes` is the truncation number on POD modes for any M- model, `samples` is the number of times a single model is trained and stored independently, and `validationSplit` refers to the ratio at which the data is divided for training and validation. The validation set will not be used for training, but to see how much the model is over-fitting for the training set. The surrogate model window from Figure 5.7 within the user interface gives the user a summary of all the selected functions corresponding to ML training, tensor update and optimal parameter search. This information can be overridden within the window or with console arguments of the same name, as `--layers` `--neurons` `--modes` `--kernelsGP` `--kernelsRBF` `--samples` and `--valSplit`. With the understanding that the snapshots will be divided into training and validation sets, the random selection of these would lead inevitably to extrapolation. To avoid ill-posed models due to extrapolation without loosing the robustness of random snapshot selection for training, Hammerhead creates a priority mask based on the boundaries of the parameter space from high fidelity models. This mask includes the combination of shape and flow parameters maximum and minimum values –excluding the baseline case– as having the maximum priority for training, then the baseline, the maximum and minimum values of THP as the next in line, and the rest of snapshots randomly shuffled. This imposes that the parameter space boundaries will always be used for training, while the rest of the space can be randomly selected to validate or train, reducing the probability of ill-posedness from extrapolation.
The model training listings are presented in Appendix D. Since RBF is loaded from `SciPy` libraries, training is done via their modules and no class is required. For NN and GP, the training functions call a class, which although very simple, are unique to Hammerhead. These classes contain the NN and GP architecture modules. The NN base uses a combination of linear and sigmoid functions for each layer, and uses the sequential model from `PyTorch` to built up the layer-neurons modules. Both the RBF and GP use their respective library's kernel functions to build the modules. Lastly, the models' state after training are stored in dictionaries along with performance data and other model relevant information as in Listing 5.9. The information stored in this dictionary is used to interpolate with the models and to produce figures representing the performance and predictions of each model, described in Section 5.6.

```
{'epoch': epoch,                                # Epoch NN or GP stopped
 training
  'kernel': kernelGP,                            # Relevant just for GP,
 needed to predict
  'modelState': model,                           # Either rbfi for RBF,
 network.state_dict() for NN or model.state_dict() for GP
  'optimizerState': optimizer.state_dict(),      # Relevant for NN and GP
  'xTrain': xTrain,                              # GP requires the features
 training data for prediction
  'outputTrain': outputTrain,                    # GP requires the output
 results training data gor prediction
  'lossTrain': lossTrainList,                    # Training loss history per
 epoch, for RBF it is lTrain, which is a list with a single value
  'lossValid': lossValidList}                    # Validation loss history per
  epoch, just for NN
```

Listing 5.9: General form of the model state and performance data storage dictionary.

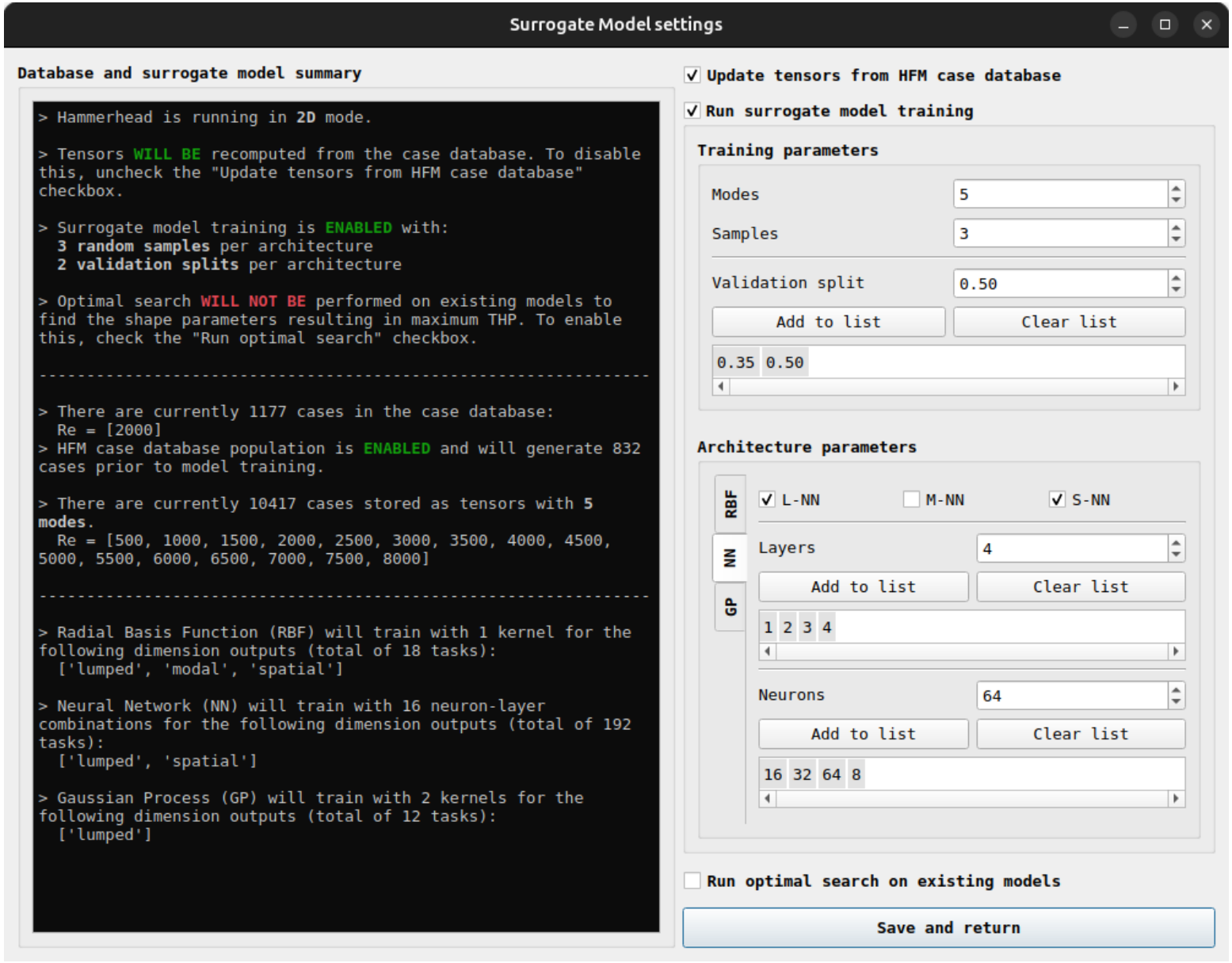


Figure 5.7: Hammerhead software surrogate model training window.

Remark 11. *Note that the RBF and NN are multi-output, which means a single model can handle a higher dimensional output results tensor. However, as stated in Section 4.5.1, the chosen GP method treats multi-output points independently by creating a batch model, reducing the computational cost of the training compared to computing the outputs' correlation. This decoupling is useful particularly for modal outputs, since the POD modes are independent of each other, but does not benefit spatial outputs, since the flow variable data points follow a profile relation. Additionally, GP creates not only a prediction, but an upper and lower limit range as a normal distribution of the prediction, a computationally and memory expensive task with increasing datasets compared to RBF and NNs due to the computation of the covariance matrix. However, this provides supplementary information about its performance and prediction confidence, visible in the figures created by Hammerhead with this models, presented in Section 5.6.*

5.6 Post-processing data and optimisation

When training is done, the data from the resulting models can be processed and analysed to obtain THP predictions, ML performance statistics and optimal shape and flow parameters searches. The processed data is reported through either written information –saved files with compiled information– and plot figures, which may vary depending on the output results basis. The console arguments to obtain these statistics are [133]

```
python ./src/Hammerhead/hammerhead.py --domain 2D --console --search
                                      --plot
```

relative to the project's root directory, followed by either of the options in Table 5.2, like `--search SNN`, where `-s` `-p` are the corresponding short-form arguments. The `-search` argument refers specifically to the optimisation process, considered an *inverse problem*, a situation in which the problem is solved from the "effects", or results, to identify the "causes" that lead to them [125]. The inverse problem in Hammerhead is treated as the *search* of optimal shape and flow parameters through maximising the THP function $\dot{Q}$ from (2.54). This process is handled by `SciPy optimize.differential_evolution`, a tool that uses a metaheuristic, non-gradient method to find the global minima of a multivariate function, described in Appendix D.1.

This argument can be used independent of `--plot` to obtain the optimal parameters, and yields the file `optimalSearchResults.pt`, which contains a dictionary of the form:

```
{'xPredExpanded': [1, features],                    # Optimal shape (and Re
 when applicable) parameters found by differential_evolution
 'thpHistory': [epochs],                            # THP values found on
 each epoch, plotting purposes
 'profileBaseline': [inpetp, outletU, outletT],     # Flow variable profile
 for the baseline case, None for L- models
 'profileOptimised': [inpetp, outletU, outletT]}    # Flow variable profile
```

```
    for the optimal THP case, None for L- models
```

Listing 5.10: Optimal search results dictionary. In the case of L- output results models, the variable profiles are not available, and these keys store `None` values.

The `-s` argument is needed before `-p` if the user wants to obtain optimal search related plots, otherwise the plotting process will provide the rest of figures, neglecting optimal search. Hammerhead splits this process five ways: prediction plots, subdivided into the THP and variable profile representations; ML performance statistics, subdivided into individual analyses and ML architecture benchmark; and the optimal THP search evolution. The four tasks are handled in parallel, with the first two being the most computationally expensive due to the need for prediction of considerable amounts of data. To built the database for these figures, Hammerhead loads the model state from the stored dictionary 5.9 of each model, and with several input feature tensors, gets the corresponding output results tensors. The plots obtained through this depend on the model's output results format, which for M- and S- include the variable profile. These representations can be seen in Figures 5.8 and 5.9.
Regarding the ML performance statistics, the dictionary 5.9 has the corresponding data, with little to no post-processing required. Similarly to the prediction plots, the ML performance statistics depend on the model's format, but this time they depend on the ML algorithm of the model, and not on the output results. The model's dictionary contains the loss J computed through the training process, which for RBF is only available after the training is finished, GP is capable to compute at each epoch for the training set, and for NN it is computed at each epoch for both training and validation sets. The last loss value, at the end of training, is used for all models to produce a benchmark as shown in Figure 5.10.

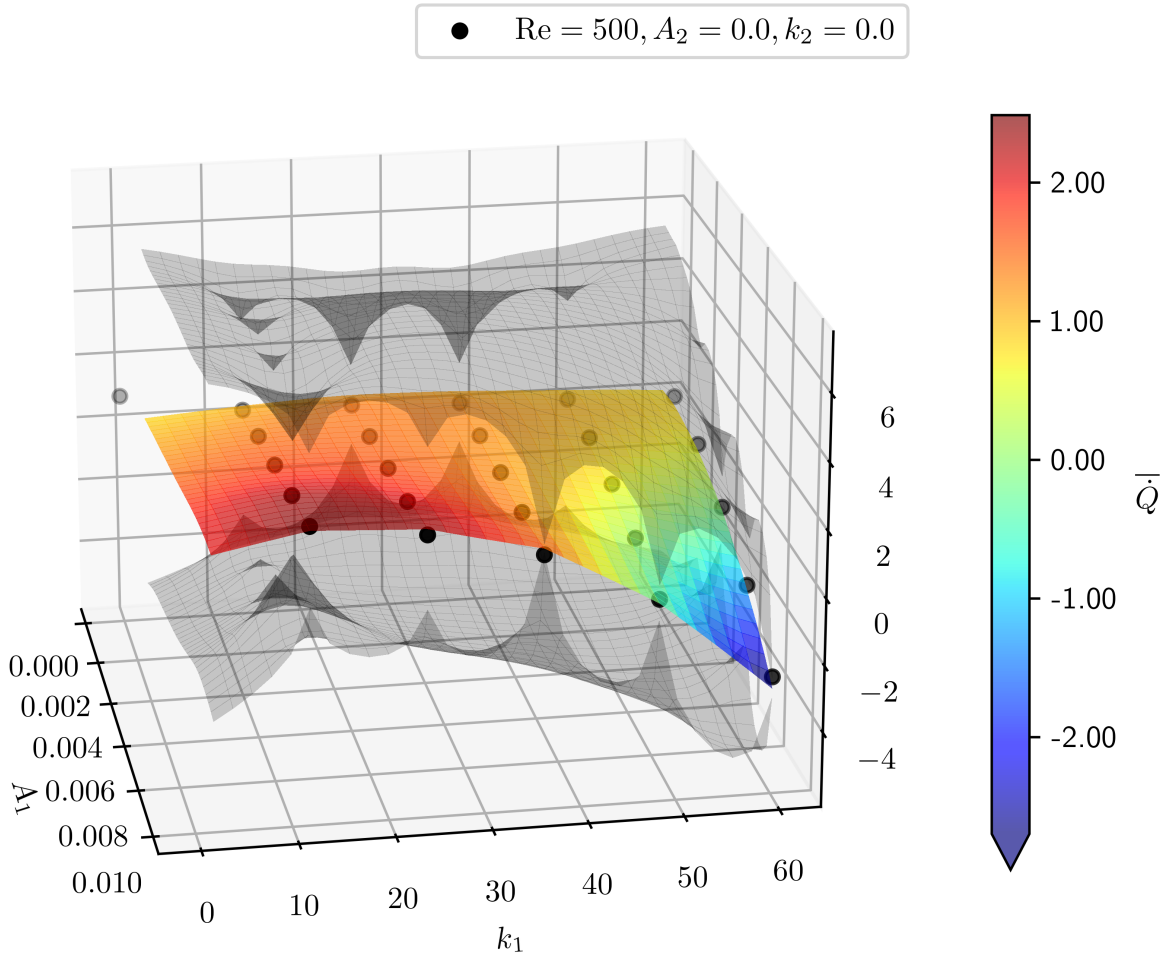


Figure 5.8: The comparative results of –surface– surrogate models' prediction of THP evaluation vs –black dots– the high fidelity data. The THP variable $\overline{\dot{Q}} = \dot{Q}_m/\dot{Q}_0$ is based on (2.52) and (2.54), where the subscript 0 refers to the baseline, smooth pipe case. The GP predictions include the confidence region from the normal distribution made by the kernel on each point, represented with the grey surfaces, while the other models do not include this feature.

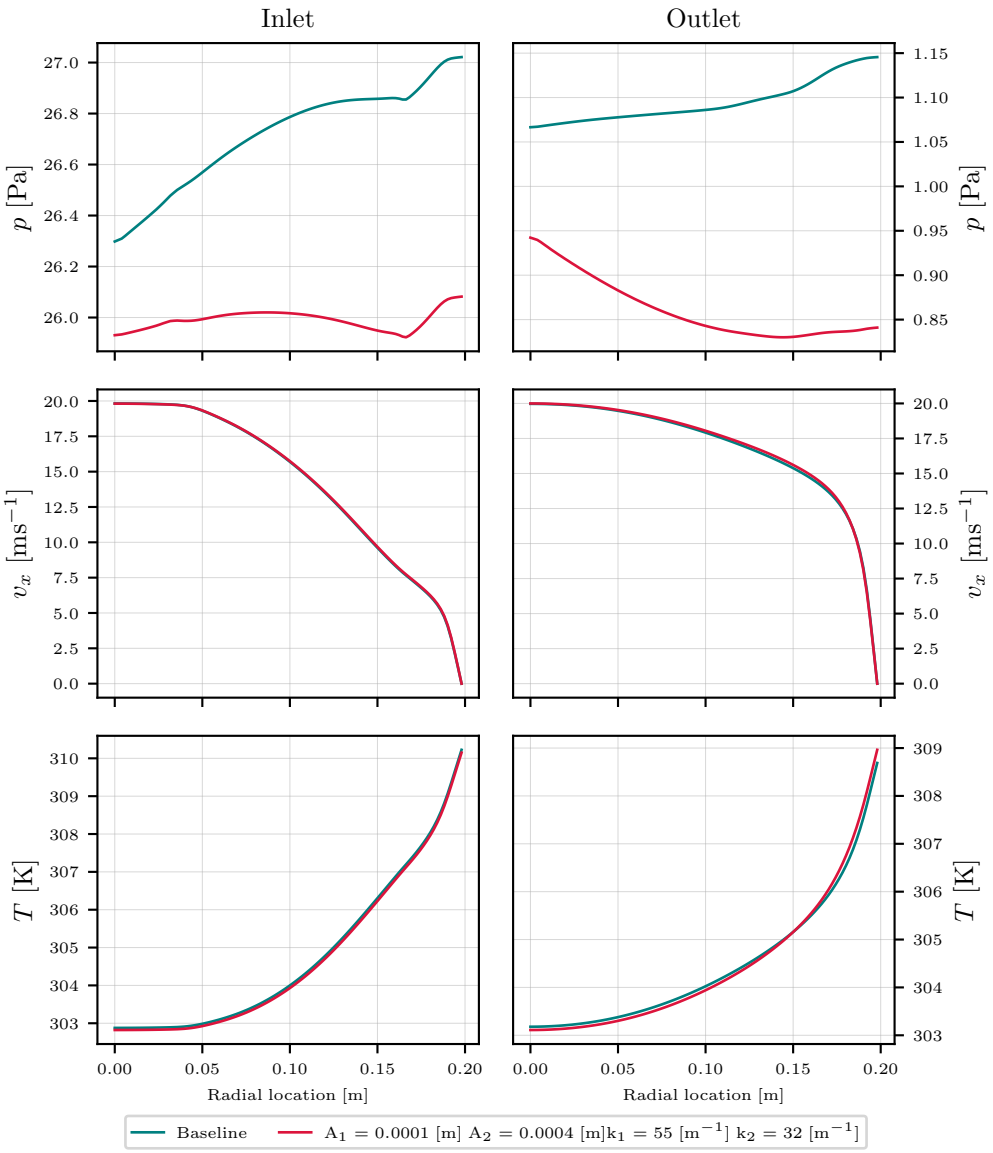


Figure 5.9: Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a micro-structured surface plate with highest $\dot{Q}_m$ vs baseline $\dot{Q}_0$ predicted by the surrogate model. This plot is not available for models trained for lumped data, which predict the THP directly and do not have the flow variable information. This plot requires `-s` to obtain the surrogate model optimised shape parameters.

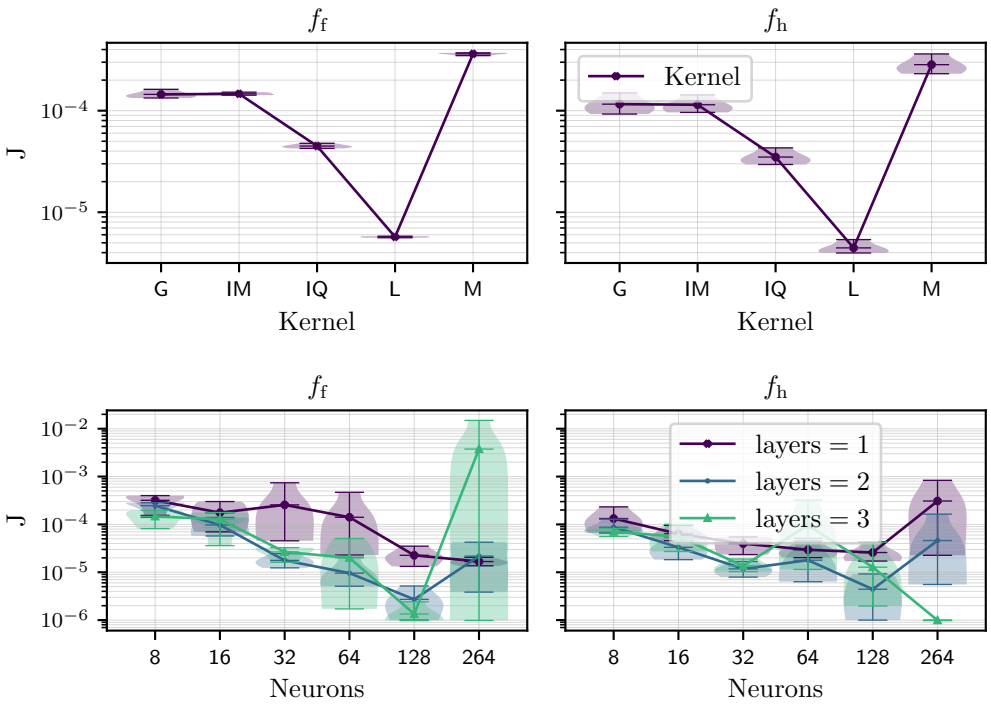


Figure 5.10: Architecture influence study on the relative norm of L^2 error of the surrogate models. The example shown is of the Lumped output –top– RBF interpolation and –bottom– NNs, where the advected heat flux (2.54) is on the left column, and the dissipation rate (2.52) is on the right column. These figures can be generated for both spatial an modal outputs where the variables displayed are the inlet and outlet pressure p, temperature T and velocity v_x, for any surrogate model –RBF, NN or GP–.

5.7 Conclusion

This chapter has presented the framework of the software Hammerhead, designed and develop as part of this work for the MOO problem of heat transfer and fluid flow in pipe systems. The software is comprised of modules that manage the production of either high fidelity or surrogate models, as well as the processing of data prior, in between and after the assembly of database for analysis and compatibility. The high-fidelity modules work with OpenFOAM's computational framework presented in previous chapters, while maintaining connectivity with the SM modules, which work within Python's libraries. The compatibility between Python and OpenFOAM, ill-posedness recognition and optimised ML implementation, as well as the underlying automation of the design process, make this framework a robust representation towards ML enhanced engineering development, as demonstrated by the examples in the next chapters.

CHAPTER 6

High fidelity model: benchmark and verification

"Sharks have molded evolution for 450 million years. All fish that are prey to the sharks have had their behaviour, their speed, their camouflage, their defense mechanisms molded by the shark."

— Paul Watson

6.1 Introduction

This chapter provides the numerical verification of the high-fidelity model based on the governing equations of Chapter 2 discretised through the methods shown in Chapter 3. OpenFOAM v2212 robustness and accuracy in discretising fluid flow through Finite Volume Methods (FVM) methodologies are assessed in the following examples: first, an $h-$refinement study is performed on the baseline and extreme scenarios of fluid flow in pipe systems in Section 6.2. These include the domains of 2D flow over a plate with (1) smooth surface and (2) parametrised geometry surface for laminar and turbulent flow regimes. Secondly, Section 6.3 shows the Thermo-Hydraulic Performance optimisation problem, to prove that a maximum value of net heat flux $\dot{Q}$ is found within the computed high-fidelity data, and that this value corresponds to a micro-structured surface with better THP behaviour than the smooth surface case, our baseline. The dissipation rate in this example plays a role of visual threshold verification, showing the cases that have losses equal or lower than the baseline.

6.2 Mesh independence study for 2D cross section

Let us first explore the solution a two-dimensional cross section a pipe for each of the flow variable fields of interest, reviewed in laminar and turbulent conditions for parametrised geometries over the surface in contact with the flow. Then again, an error measurement is needed to quantify the accuracy and convergence of the numerical solution. Consequently, an $h-$refinement study is conducted to obtain an *ideal* mesh size that can be shared across solutions to accurately solve the problem with a lower computational cost. The objectives of this example are:

Obj VI.I. Assessment of the proper selection of suitable OpenFOAM v2212 numerical schemes presented in Section 3.3.2 within the `chtMultiRegionSimpleFoam` solver for different Reynolds numbers regimes and interface geometries in conjugate heat transfer.

Obj VI.II. Demonstrate the mesh-independent $q-$order of accuracy of the numerical schemes presented in Section 3.3.2 via an $h-$refinement study of a structured mesh.

The discretisation of a domain defined by a two-dimensional cross section of a pipe with solid Ω_s in contact with fluid flow Ω_f on one side $-\Omega = \Omega_f \bigcup \Omega_s-$ is considered symmetric at the *pipe-like* geometry stream-wise centre line, examined in the standard Eucledian basis. With a length of $L = 3$ m and a thickness of $t = 0.06$ m, the cross section has single-surface contact with flow. These are separated by a domain of half width $r = 0.2$ m. The baseline pipe domain is shown in Figure 6.1–right–, and the relevant physical and thermal properties of both fluid and solid regions are listed in Table 6.1. The surface in contact with the fluid Γ_{sf} is parametrised using a simplified harmonic function representing the shark skin inspired biomimetic structure described in Section 2.5.2

$$y\left(x\right) = r + \sum_{i=1}^{\mathrm{N_h}} \mathrm{A}_i \left(\cos\left(\pi \mathrm{k}_i x\right) + 1\right) , \tag{6.1}$$

where $\mathrm{N_h}$, A_i and k_i are the number of harmonics to implement, and the amplitude and

Parameter	Value	Parameter	Value
ρ	1^* [kg/m^2]	T_f/T_s	0.6667
Re	{500, 8000}	k_f/k_s	0.2414
Pr	0.2414	$\varsigma_{p,f}/\varsigma_{p,s}$	5230

Table 6.1: Flow over *smooth surface* and *micro-structured surface* - Physical parameters of flow over *smooth surface* and *micro-structured surfaces*. *The density is considered constant for an incompressible problem.

wavenumber of the harmonic, respectively. This treatment results in a semi-complex geometry as shown in Figure 6.1–right–, which is briefly described in Section 5.3. The set of meshes generated with OpenFOAM's `blockMesh` tool are fitted to the surface shape computed by (6.1) within $x = [0.5,\ 2.5]$ m. The surface is left *smooth* at the entrance and exit of the flow up to $x = 0.5$ m and from $x = 2.5$ m, respectively. The smooth entrance and exit will serve as *probe* regions, to avoid local fluctuation areas affecting the post-processing computations of system-scale THP in the shape optimisation problem from Section 6.3. The geometry values surveyed for this problem are $H = 1$, $\mathrm{A}_1 = \{0.001,\ 0.01\}$ m and $\mathrm{k}_1 = 30$ m^{-1}. Both the *smooth surface* and the *micro-structured surface* are inspected in these examples and are shown along with their boundary conditions in Figure 6.1. The boundary value problem for domain Ω is defined in Table 6.2, described for the boundaries from Figure 6.1. The laminar velocity profile gradient is applied as a Dirichlet boundary condition $\boldsymbol{v} = v_x\,(y)\,\boldsymbol{e}_x$ on Γ_f^{in}, at the inlet boundary of the fluid domain

$$\boldsymbol{v} = v_x\,(y)\,\boldsymbol{e}_x = v_{x,max}\left[1 - \left(\frac{y}{r}\right)^2\right]\boldsymbol{e}_x\,, \tag{6.2}$$

where $v_{x,max}$ is computed from the corresponding Re. For $Re > 2000$ values, turbulence is considered, and the $k-\omega$ SST [134, 135] model is used due to its optimised formulation for the viscous sublayer in wall-bounded flows while keeping a low sensitivity to free-stream turbulence

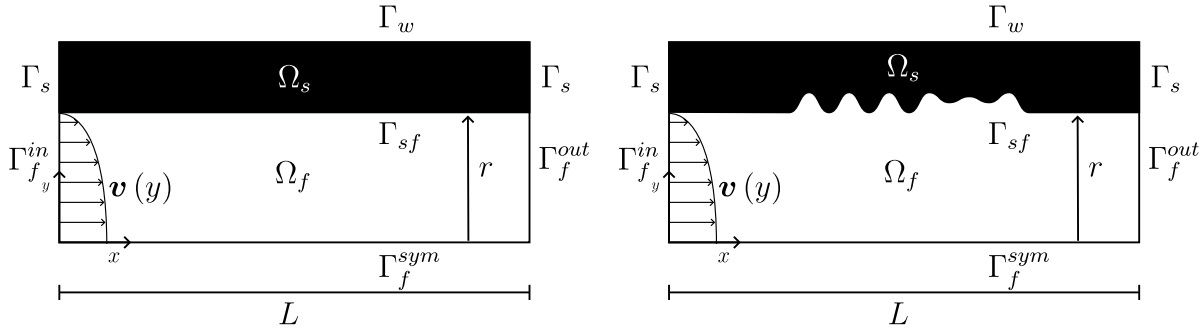


Figure 6.1: Flow over *smooth surface* and *micro-structured surface* - Domains of internal flow in a pipe represented by 2D flow over –left– a *smooth* surface plate and over –right– a *micro-structured* surface plate. The micro-structured surface is computed with (6.1), resulting in a semi-complex geometry described briefly in Section 5.3.

Boundary	$\boldsymbol{v}$ [m s^{-1}]	T [K]	p [Pa]
Ω_f initial	$v_{x,max}$	300	0
Ω_s initial	–	500	–
Γ_f^{in}	$v_x\,(y)\,\boldsymbol{e}_x$	$\nabla T \cdot \boldsymbol{n} = 0$	$\nabla p \cdot \boldsymbol{n} = p_o$
Γ_f^{out}	$\nabla \boldsymbol{v} \cdot \boldsymbol{n} = 0$	$\nabla T \cdot \boldsymbol{n} = 0$	0
Γ_f^{sym}	$\nabla \boldsymbol{v} \cdot \boldsymbol{n} = 0$	$\nabla T \cdot \boldsymbol{n} = 0$	$\nabla p \cdot \boldsymbol{n} = 0$
Γ_{sf}	0	$T_f = T_s$	$\nabla p \cdot \boldsymbol{n} = 0$
Γ_s^w	–	500	–
Γ_s	–	$\nabla T \cdot \boldsymbol{n} = 0$	–

Table 6.2: Boundary and initial conditions for the domain Ω of internal flow Ω_f in a pipe represented by a 2D flow over a surface plate Ω_s –$\Omega = \Omega_f \bigcup \Omega_s$–. No velocity and temperature values are needed for the solid region.

properties. It is reported that flow separation and adverse pressure gradients behave better than other RANS models, even though it still produces large turbulence levels in regions of large normal strain [45, 136]. This model is described in Appendix C.1, and requires a $y^+ \leq 1$ to appropriately compute the turbulent boundary layer. The turbulent kinetic energy fields resulting from the use of the $k-\omega$ SST turbulence model in the simulations at $Re = 8000$ are shown in Figure 6.2. The values obtained demonstrate the need for turbulence modelling, justifying their use in this study. The residuals achieved for the turbulence model variables k and ω are $[9.94e^{-8},\ 9.97e^{-8}]$ and $[9.20e^{-8},\ 9.81e^{-8}]$, respectively.

An $h-$refinement study is adopted to analyse the numerical solution sensitivity to grid cell size. A variation in grid spacing value h for different mesh levels is computed from the definition $\mathrm{h} = \Delta y/r$, where Δy is the $y-$direction element size. The study is applied to a relative L^2-norm RMSE Er_ϕ from (6.3), measured in Ω_f to evaluate the accuracy and convergence of the numerical solution ϕ against a reference mesh $\overline{\phi}$ of overestimated grid refinement. For three h grid spacing meshes with growth ratio of (r = 2), the error must be proportional to the order of accuracy q in the way h. When this relationship is met, the meshes are considered in the *asymptotic convergence region*. Consider the error Er_ϕ as

$$\mathrm{Er}_\phi = \frac{||\varepsilon_\phi||_{L^2(\Omega)}}{||\overline{\boldsymbol{x}}||_{L^2(\Omega)}}, \tag{6.3}$$

where $||\varepsilon_\phi\,(\boldsymbol{x})\,||_{L^2(\Omega)}$ is the L^2 norm of the error $\varepsilon_\phi\,(\boldsymbol{x})$ and $\overline{\boldsymbol{x}}$ is the reference solution. In general, $\mathrm{Er}_\phi = b\mathrm{h}^\mathrm{q}$, hence $\log(\mathrm{Er}_\phi) = \log(b) + \mathrm{q}\log(\mathrm{h})$. The slope of convergence can be

computed as

$$\begin{aligned} \mathrm{q} &= \frac{\log\left(\mathrm{Er}_{\phi_2}\right) - \log\left(\mathrm{Er}_{\phi_1}\right)}{\log\left(\mathrm{h}_2\right) - \log\left(\mathrm{h}_1\right)} = \frac{\log\left(\frac{\mathrm{Er}_{\phi_2}}{\mathrm{Er}_{\phi_1}}\right)}{\log\left(\frac{\mathrm{h}_2}{\mathrm{h}_1}\right)} \\ &= \frac{\log\left(\frac{||\varepsilon_{\phi_2}||_{L^2(\Omega)}}{||\varepsilon_{\phi_1}||_{L^2(\Omega)}}\right)}{\log\left(\frac{\mathrm{h}_2}{\mathrm{h}_1}\right)} = \frac{\log\left(\frac{\left(\int_\Omega \left(\phi_2 - \overline{\phi}\right)^2 d\Omega\right)^{1/2}}{\left(\int_\Omega \left(\phi_1 - \overline{\phi}\right)^2 d\Omega\right)^{1/2}}\right)}{\log\left(\frac{\mathrm{h}_2}{\mathrm{h}_1}\right)} . \end{aligned} \tag{6.4}$$

Then, the error values $||\varepsilon_\phi||_{L^2(\Omega)}$ against the grid spacing h are expected to have a behaviour of

$$||\varepsilon_r||_{L^2} \leq C\mathrm{h}^{\mathrm{p}} , \tag{6.5}$$

where C is an unknown constant independent of the grid spacing h and order of accuracy p. It is inferred from (6.5) that the logarithmic gradient for the error-spacing $||\varepsilon_r||_{L^2} - \mathrm{h}$ rate of convergence is expected to be or approach the order of accuracy p of the numerical solution.

The Reynolds $Re = \{500,\ 8000\}$ from Table 6.1 will be used for the laminar and turbulent conditions, respectively. Grid levels are shown in Figure 6.3. The grid parameters for each $l-$level mesh and quality are listed in Tables 6.3 shared across the 2D plate examples. It is worth noting that $\mathrm{r_h} = 2$, and all grids lie well within acceptable maximum non-orthogonal values –An orthogonality of around 40° 60° is generally accepted–. The error values $||\varepsilon_r||_{L^2}$

$Re = 8000$, $k - \omega$ SST turbulence model

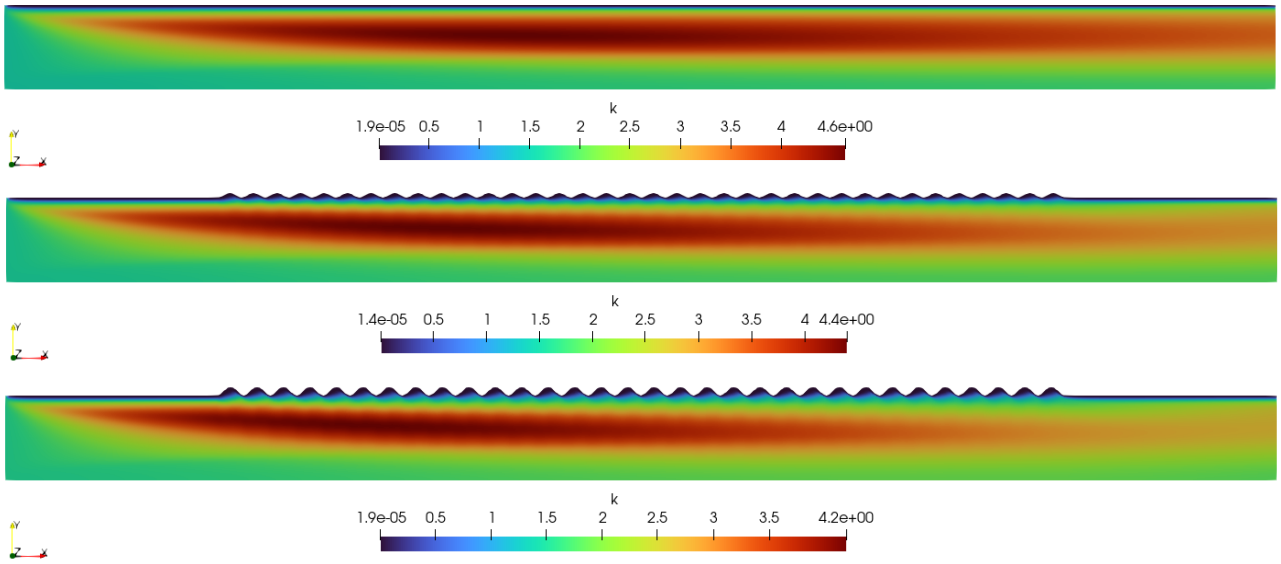


Figure 6.2: Flow over *smooth surface* and *micro-structured surface* - Turbulent kinetic energy field k results at $Re = 8000$ for the mesh with $1-$level of refinement and turbulence model $k - \omega$ SST.

$l-$level	h	Δy	$y-$cell count	$x-$cell count	$y-$expansion ratio
3	0.04	$5.53e^{-3}$	25	275	1.02
2	0.02	$2.19e^{-3}$	50	550	1.02
1	0.01	$7.14e^{-4}$	100	1100	1.02
0	0.005	$2.25e^{-4}$	200	2200	1.01

	Maximum non-orthogonality [°]		
$l-$level	$A_1 = 0$ m	$A_1 = 0.001$ m	$A_1 = 0.01$ m
3	0	28.1	43.6
2	0	28.9	44.6
1	0	29.3	44.9
0	0	29.3	45.1

Table 6.3: Flow over 2D *smooth* and *micro-structured* surfaces - Mesh refinement parameters and maximum non orthogonality for the 2D cross section flow over *smooth surface* and *micro-structured* examples.

computed for each mesh are presented against h in Figure 6.4, while the *measured* rates of convergence of all flow variable fields for the tested shape parameters are provided in Table 6.4. From the relation in (6.5) it is observed that most of the variables in the numerical results have a rate of convergence between 1 and 3, a behaviour expected from 1st and 2nd order of accuracy methods. This is not the case for the temperature in laminar regimes, which is lower than the expected minimum of 1 for all geometry examples. The observed enthalpy residual is of $4.38e^{-6}$ to $4.86e^{-6}$, where it is maintained for over 3000 iterations. The rate of convergence also decreases slowly when increasing non-orthogonality in the mesh as the geometry grows sharp turns, but does not go below 1, except for pressure at $A_1 = 0.01$m in the laminar regime where the value is 0.895. The reached pressure residual in this geometry is $9.26e^{-8}$ compared to $9.98e^{-8}$ in the smooth surface. A slightly improved convergence rate is seen for the temperature and pressure variables in the turbulent regime over the laminar, where turbulent residuals range from $9.1e^{-7}$ to $1.62e^{-6}$ for temperature and $9.53e^{-8}$ to $9.85e^{-8}$ for pressure.

6.2.1 Discussion and findings

From the observed rates of convergence, particularly the comparison between temperature rates in laminar and turbulent regimes, it seems like the results from increased mesh refinement do not reach the expected field values at the moment of simulation break compared to coarser meshes regardless of residual values. The validation assessment of the smooth surface against analytical data from the Hagen-Poiseuille law is presented in Appendix C.2. However, the rate of convergence is close to the value expected from a scheme of 1st order of accuracy, and the

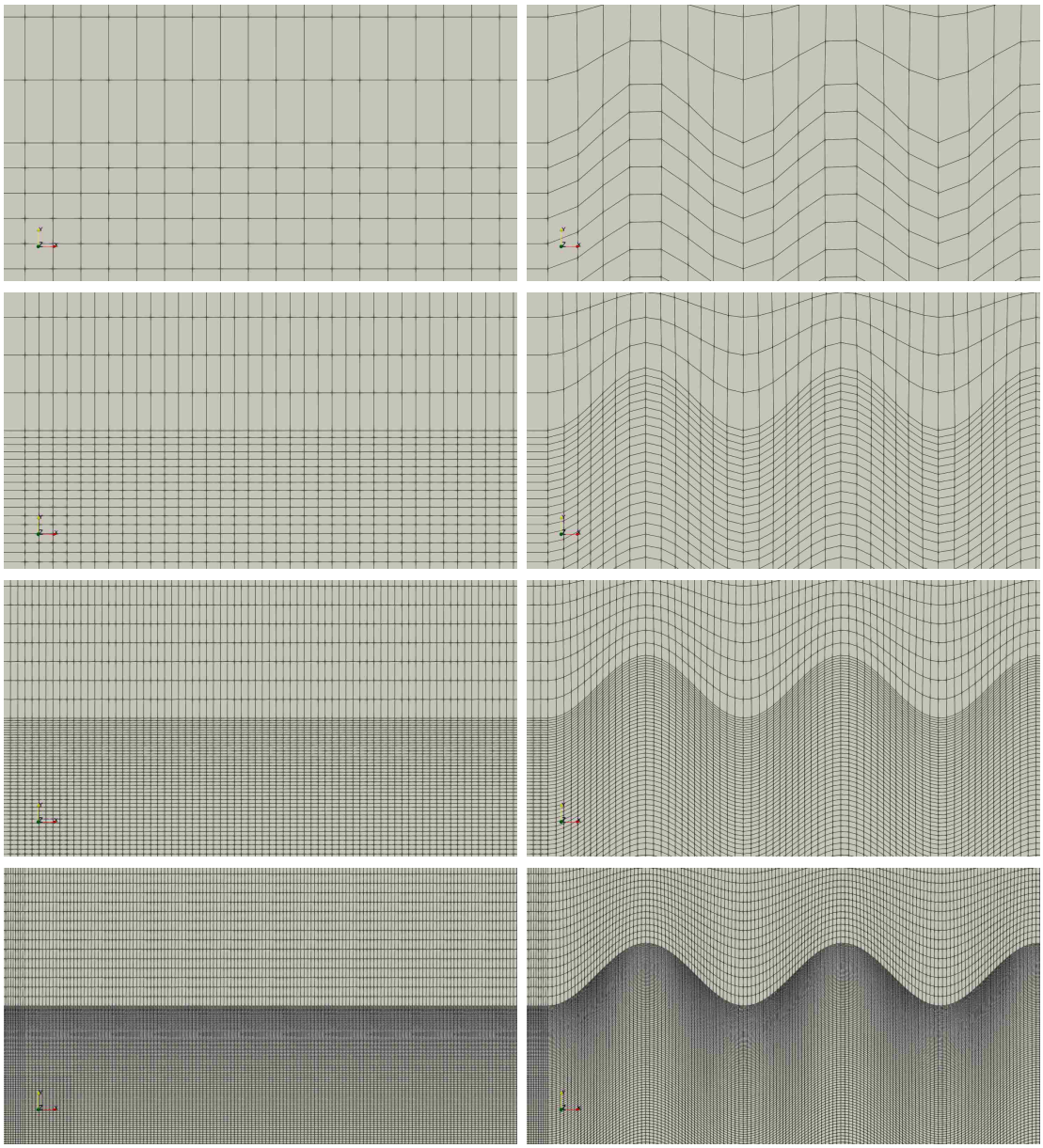


Figure 6.3: Flow over 2D *smooth* and *micro-structured* surfaces - Near-wall close up of the domain discretisation of a 2D cross section flow representing internal flow in a pipe for the grid spacing of –from top to bottom– h = 0.04, h = 0.02, h = 0.01 and h = 0.005 for –left to right– $A_1 = 0$ and $A_1 = 0.01$.

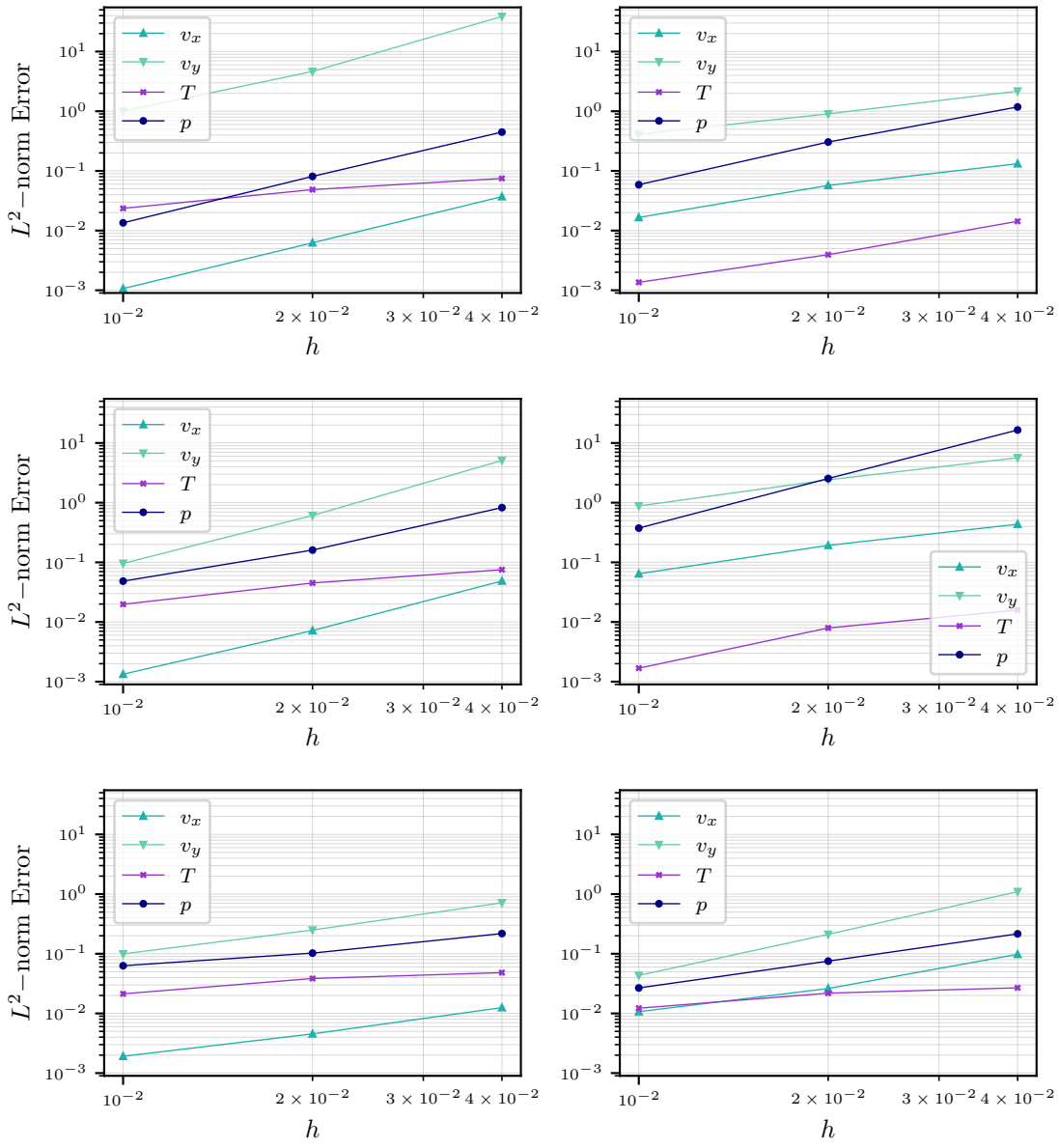


Figure 6.4: Flow over 2D *smooth* and *micro-structured* surfaces - Relative norm L^2 error $||\varepsilon_r||_{L^2}$ against h for the surface geometries of –from top to bottom– $A_1 = 0.0$ m, $A_1 = 0.001$ m, and $A_1 = 0.01$ m in –left to right– laminar and turbulent regimes.

Amplitude A_1 [m]	q_{v_x}	q_{v_y}	q_T	q_p
0.0	2.56	2.635	0.83	2.525
0.001	2.44	2.78	0.93	1.765
0.01	1.335	1.425	0.59	0.895
Amplitude A_1 [m]	q_{v_x}	q_{v_y}	q_T	q_p
0.0	1.495	1.2	1.7	2.18
0.001	1.38	1.34	1.62	2.73
0.01	1.595	2.335	0.57	1.5

Table 6.4: Flow over 2D *smooth* and *micro-structured* surfaces - Average measured rates of convergence for the numerical solution for each flow variable field within –from top to bottom– laminar and turbulent regimes.

local Nusselt number Nu[1] evaluation from Appendix C.2 show that the thermal behaviour is within what is expected in a fully developed thermal layer. From the analysis of thermally developed laminar flow in a cylindrical pipe with constant wall temperature, the increase in mean temperature has an exponential decay and is time-dependent, reducing the convergence of temperature in steady-state analysis. Additionally, the stagnant behaviour of residuals lead us to conclude the convergence is acceptable for this case of study. The remaining flow variables, as well as temperature in turbulent regimes, behave well within the expected rates of convergence for 1st and 2nd order of accuracy schemes, with some occasionally surpassing 2nd order. The increased rate when using turbulence modelling might be related to the turbulent Nu results from the validation in Appendix C.2, showing the thermal exchange is convection dominant [2], speeding the decay of mean temperature increase.

Careful consideration has to be taken when fitting the grid to more complex geometries, since the rate of convergence decreases slowly when increasing non-orthogonality, as was contemplated beforehand. There is an odd increase in the convergence rate for the flow variable v_y when discretising more complex geometries in turbulent regimes. This behaviour is concluded to be related to the direction of the cells close to the wall, in which an atypical alignment to the $y-$direction is forming, making it *easier* to the numerical solutions to solve for v_y accurately. This also might be propagating into pressure convergence rate due to the use of correction equations with the SIMPLE algorithm, which updated pressure from the velocity field in previous iterations.

In relation to Objective VI.I, the numerical schemes used in these examples show an approximate level of accuracy of 1st and 2nd order methods, and the rates of convergence suggest these are kept mesh independent, as regarded in Objective VI.II. Consequently,

[1] Ratio of total heat transfer to conductive heat transfer at the interface between flow and a solid. This dimensionless number is used to understand convection in fully developed boundary layers and the temperature profile development. [137, 138]

[2] $Nu = 1$ is pure conduction, $Nu \geqq 1$ is convection dominant, and turbulence increases Nu.

Objectives VI.I and VI.II are considered achieved for the level of accuracy required within this study with respect to the computational costs resulting from using the 1−level refinement grid –[2, 6] h CPU execution time per simulation–. This is to prepare the mesh as a *modular* base grid within *Hammerhead*'s software basis to account for both low and high *Re* regimes along with capturing smaller-scale features when increasing the complexity of the geometry.

6.3 Benchmark for the high fidelity model based THP optimisation problem

Let us consider the multi-objective optimisation problem of THP in pipe flows within laminar and turbulent conditions when in contact with parametrised geometries in the wetted surface. The objectives of this example are:

Obj VI.III. Demonstrate the influence of geometry variation on the thermo-hydraulic behaviour of the flow by implementing the shape parametrisation problem described in Section 2.5.2 on the high fidelity model domains.

Obj VI.IV. Analyse the thermal and hydraulic behaviour of flow over surface roughness and identify the underlying mechanisms resulting from the flow-roughness interaction that cause such behaviour.

Obj VI.V. Expose the shortcomings of using high fidelity models to approach the shape parameter optimisation problem, as well as the limitations in convergence and body-fitting meshes when considering more complex geometries.

The domain and boundary conditions are inherited from the examples in Section 6.2. To study the influence of micro-structure geometries in interaction with the flow on the overall THP behaviour, the surface in contact with the fluid Γ_{sf} is parametrised using a simplified harmonic function representing the shark skin inspired biomimetic structure described in Section 2.5.2. The changes in these parameters form a set of cases or *snapshots* $\mathrm{N_s}$. The relevant physical and thermal properties of both fluid and solid regions, along with the shape parametrisation features for the surface, are listed in Table 6.5.
The meshes generated with OpenFOAM's blockMesh tool are fitted to the surface shape computed by (6.1) within $x = [0.5,\ 2.5]$ m. The surface is left *smooth* at the entrance and exit of the flow up to $x = 0.5$ m and from $x = 2.5$ m, respectively. The smooth entrance and exit will serve as *probe* regions as shown in Figure 6.5, to avoid local fluctuation areas affecting the post-processing computations of system-scale THP based on the definitions of dissipation rate and advected heat flux from (2.52) and (2.54), respectively, in Section 2.4.
The normal contribution of velocity $\boldsymbol{v} \cdot \boldsymbol{n}$ is null in the surface and symmetry boundaries Γ_{sf}, Γ_f^{sym}, which leads to the THP evaluation performed by extracting the inlet/outlet boundary Γ_f^{in}, Γ_f^{out} information of the three relevant flow fields, velocity, temperature and pressure. It is visible in (2.54), as stated in Section 2.4.2, that the net thermal power output $\dot{Q}$ relates both advected heat flux and dissipation rate, yet as a value it will not detail if the pressure loss increases in reference with the smooth surface case - our *baseline*. The laminar case baseline has been validated in Appendix C.2. In the interest of enhancing heat transfer without increasing

Parameter	Value	Parameter	Value
ρ	1* [kg/m^2]	T_f/T_s	0.6667
Re	[500, 8000]	$k_{d,f}/k_{d,s}$	0.2414
Pr	0.2414	$\varsigma_{p,f}/\varsigma_{p,s}$	5230
N_h	1, 2		
A_i	[0.002, 0.01] m	k_i	[12, 60] m^{-1}

Table 6.5: THP optimisation benchmark of the high fidelity model - Physical parameters of flow in the THP evaluation problem. The Re for the 2D plates example are referred to in discrete intervals, with a step size of 500. Similarly, the geometry parameters A_i and k_i are discrete intervals, with 5 values in each parameter range for the 2D plates example. *The density is considered constant for an incompressible problem.

dissipation rate, this baseline acts as a visual threshold, showing the micro-structured cases that have losses equal or lower than the smooth surface case for each Reynolds number considered for the study. Granting that $\dot{Q}_0$ is the *baseline* and $\dot{Q}_m$ any micro-structured surface case, the observed results are portrayed as the normalised net thermal power output $\overline{\dot{Q}}$

$$\overline{\dot{Q}} = \frac{\dot{Q}_m}{\dot{Q}_0}\,. \tag{6.6}$$

The study of THP optimisation in a 2D cross section flow, as seen in Figure 6.1 top models, includes almost 10 000 combinations of shape and flow parameters, presented in Table 6.5, with an average CPU time of 4 h per simulation. Table 6.6 exhibits Re where optimisation was recognised, the THP baseline $\dot{Q}_0$, the optimum shape parameters and their respective nor-

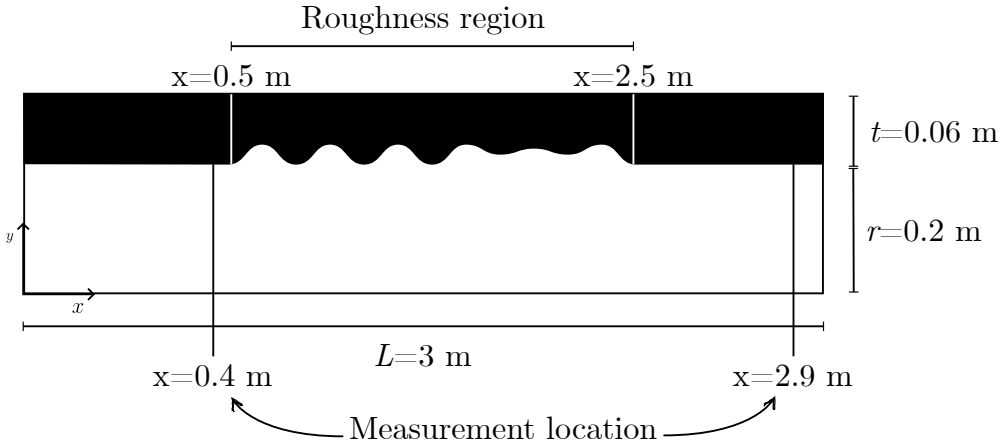


Figure 6.5: THP optimisation benchmark of the high fidelity model - Rough geometry region and flow variable measurement locations in the simulation domain.

malised power output $\overline{\dot{Q}}$ and normalised dissipation rate $\overline{f_f}$. In laminar regimes we observe extreme cases of optimisation, having an increase in the computed THP of over 7 times above the baseline. This behaviour changes when crossing to turbulent regimes. Beyond laminar regime, at $Re = 2000$, and up to $Re = 6500$ no THP improvement was found within the shape parameter space selected for the study. However, at $Re = 7000$ and above there is a modest improvement, although these were found in parametric regions with an increase in dissipation rate –$\overline{f_f} > 1$–.

Table 6.7 shows the information of rough surface cases with optimum $\overline{f_f}$ and worst $\overline{\dot{Q}}$. In laminar regimes, the optimum $\overline{f_f}$ was found within THP optimisation geometries, however, the highest optimisation ratio was found within higher Re values in turbulent regimes. The roughness parameters in Tables 6.7 and 6.7 indicate that for laminar regimes, higher amplitudes A_i and lower wavenumbers k_i benefit both $\overline{\dot{Q}}$ and $\overline{f_f}$. This changes for turbulence, where $\overline{f_f}$ benefits from high amplitudes and wavenumbers, but lower amplitudes and higher wavenumbers aid in the gain of $\overline{\dot{Q}}$. However, the minimum power output was found in the same geometry across all Re values at the highest A_i and k_i, with varying dissipation rate optimisation. In turbulent regimes the normalised dissipation rate is negative –$\overline{f_f} < 0$– but close to 0 except for $Re = 2000$, while in laminar they are positive and over 1. The negative values come from f_f of rough surface cases having a different sign from the baseline, and where negative values of f_f indicate turbulent kinetic energy generation and positive values are dissipation. Additionally, in laminar regimes the $\overline{\dot{Q}}$ values are negative, something not present in turbulent regimes. Recalling (2.54), this can only happen when the dissipation rate is larger than the advected heat flux or if the heat flux is negative. Even though all the cases where this happens have higher dissipation rate, the cause is loss in advected heat.

The influence of roughness parameters in the flow and how this changes in laminar and turbulent regimes can be observed better in Figure 6.6, which presents a surface-like data cloud of points, representing a *cross-section* on $\mathrm{A}_2 = 0.0$ m and $\mathrm{k}_2 = 0.0$ m^{-1} of the parameter space, showing $\overline{\dot{Q}}$ results. This further demonstrates that $\overline{\dot{Q}}$ optimisation is found in laminar regimes –top to bottom, first three plots– by the lower right corner of the plots, at high amplitudes with low wavenumbers and in turbulence regimes –top to bottom, last four plots– by the upper left corner at low amplitudes with high wavenumbers. However, the minimum $\overline{\dot{Q}}$ is always found by the upper right corner, at high amplitudes and wavenumbers.

The $\overline{\dot{Q}}$ values for all cases are shown in Figure 6.7 as a cloud of points, one per parameter combination case. Each plot contains only the data relevant to a single Re. The cases that did not increase the dissipation rate compared to the baseline –$\overline{f_f}$– are highlighted in cyan, while the cases with worse dissipation rate remain in black. It is clear that in laminar regimes –top to bottom, first four plots–, most shapes with a decrease in dissipation rate have an improved THP. Particularly in $Re = \{1000,\ 1500\}$, it looks like in cases with no $\overline{\dot{Q}}$ enhancement, no dissipation optimisation is possible either. This changes rapidly and abruptly when moving towards $Re = 2000$ and higher, where transition into turbulence takes place. In turbulent regimes –top to bottom, last four plots–, increasing Re values leads to the increment of rough surface cases with higher dissipation rate, while no improvement is seen in $\overline{\dot{Q}}$ compared to the baseline until reaching $Re = 7000$.

The variable profiles of the baseline case are shown against geometries with optimum power

Optimum $\overline{\dot{Q}}$							
Re	A_1 [m]	A_2 [m]	k_1 [m^{-1}]	k_2 [m^{-1}]	$\dot{Q}_0$ [W/m^2]	$\overline{f_f}$	$\overline{\dot{Q}}$
500	$1e^{-2}$	$1e^{-2}$	12	12	$3.230e^5$	0.9468	5.2145
1000	$1e^{-2}$	$1e^{-2}$	12	24	$1.395e^6$	0.8085	4.7945
1500	$1e^{-2}$	$1e^{-2}$	12	36	$2.280e^6$	0.4816	7.418
7000	$2e^{-3}$	-	60	-	$3.675e^8$	1.0856	1.0013
8000	$2e^{-3}$	-	60	-	$4.131e^8$	1.082	1.0005

Table 6.6: THP optimisation benchmark of the high fidelity model - Net power output $\dot{Q}_0$ of baseline surface in different Re regimes and shape parameters of the micro-structured surface plate with optimum $\overline{\dot{Q}}$ in the corresponding flow conditions.

Optimum $\overline{f_f}$						
Re	A_1 [m]	A_2 [m]	k_1 [m^{-1}]	k_2 [m^{-1}]	$\overline{f_f}$	$\overline{\dot{Q}}$
500	$1e^{-2}$	$1e^{-2}$	12	24	0.9412	4.2132
1000	$1e^{-2}$	$1e^{-2}$	12	24	0.8085	4.7944
1500	$1e^{-2}$	$8e^{-3}$	12	36	0.4759	7.3095
2000	$1e^{-2}$	$1e^{-2}$	36	48	0.0095	0.6265
7000	$8e^{-3}$	$1e^{-2}$	60	60	−0.1627	0.4207
8000	$8e^{-3}$	$1e^{-2}$	60	60	0.0357	0.4628
Minimum $\overline{\dot{Q}}$						
Re	A_1 [m]	A_2 [m]	k_1 [m^{-1}]	k_2 [m^{-1}]	$\overline{f_f}$	$\overline{\dot{Q}}$
500	$1e^{-2}$	$1e^{-2}$	60	60	1.3449	−12.7311
1000	$1e^{-2}$	$1e^{-2}$	60	60	1.7423	−6.8609
1500	$1e^{-2}$	$1e^{-2}$	60	60	2.3031	−7.3178
2000	$1e^{-2}$	$1e^{-2}$	60	60	−2.6805	0.1046
7000	$1e^{-2}$	$1e^{-2}$	60	60	−0.2192	0.3682
8000	$1e^{-2}$	$1e^{-2}$	60	60	−0.1668	0.3794

Table 6.7: THP optimisation benchmark of the high fidelity model - Shape parameters of the micro-structured surface plate with optimum $\overline{f_f}$ and minimum $\overline{\dot{Q}}$ in the corresponding Re regimes.

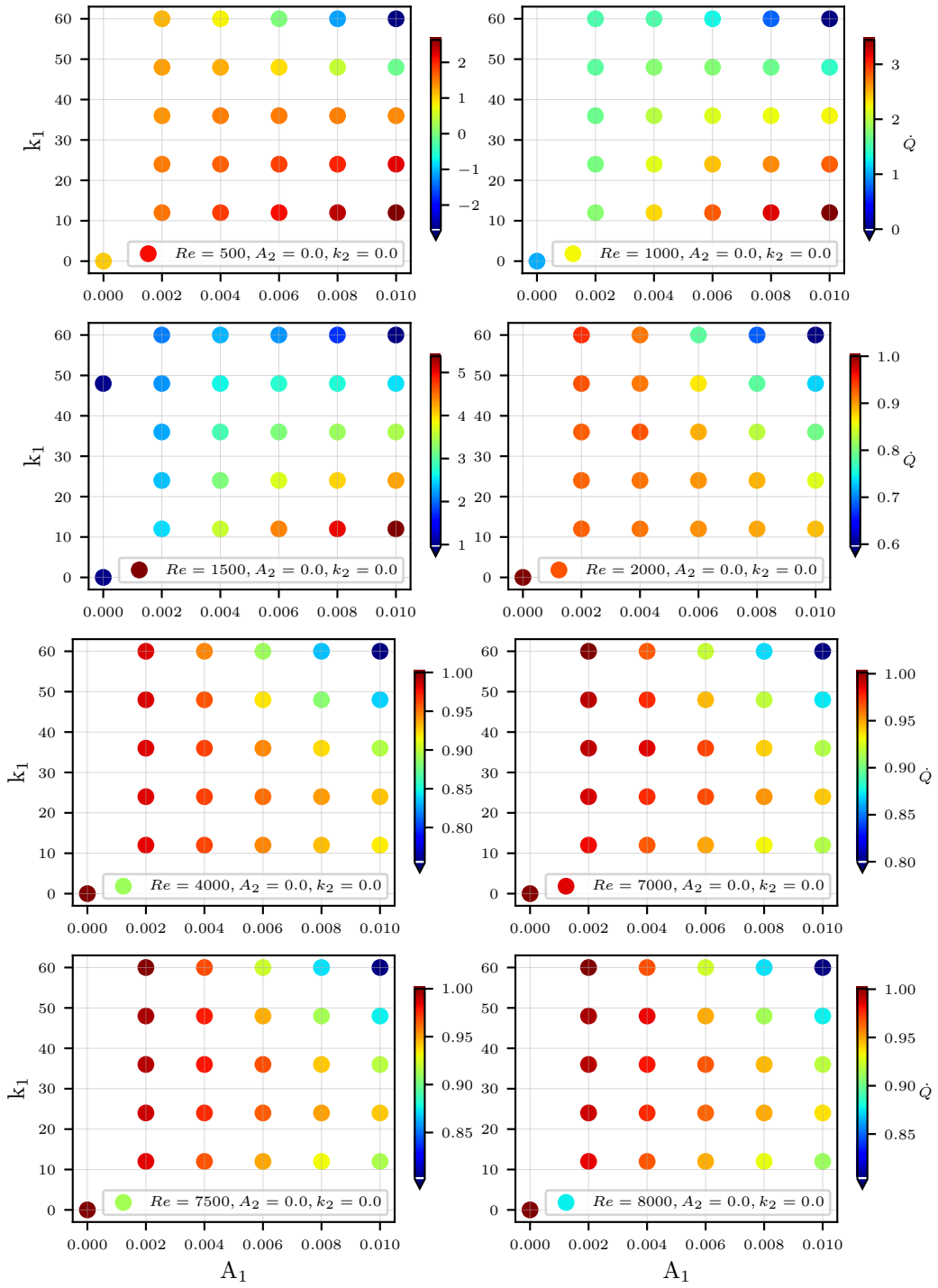


Figure 6.6: THP optimisation benchmark of the high fidelity model - Cloud of $\overline{\dot{Q}}$ points from the parametrised high fidelity models.

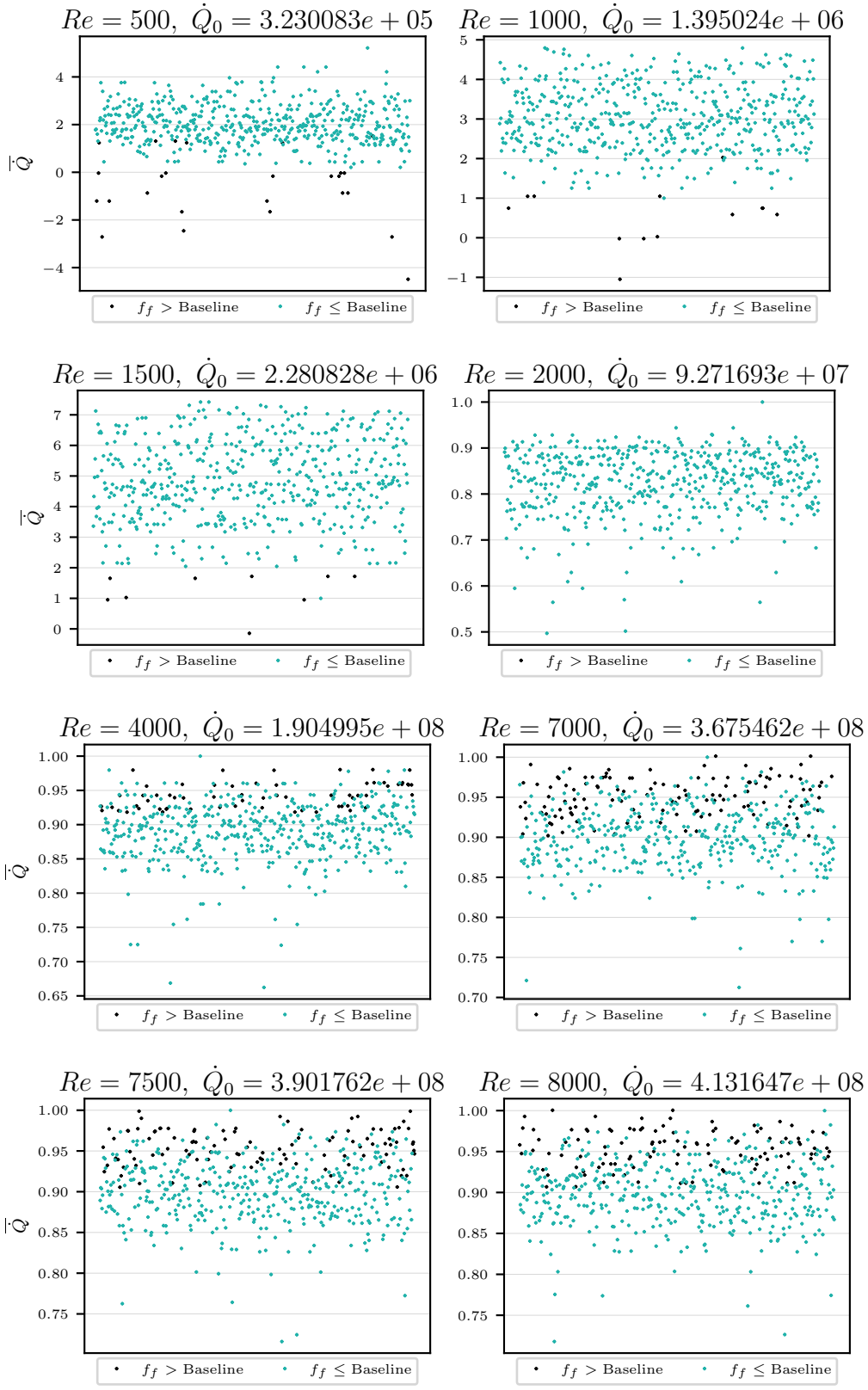


Figure 6.7: THP optimisation benchmark of the high fidelity model - Cloud of $\overline{\dot{Q}} = \dot{Q}_m/\dot{Q}_0$ points from the parametrised high fidelity models. The points with values under the baseline dissipation rate as the maximum threshold are shown in blue.

output $\overline{\dot{Q}}$, minimum laminar power output $\overline{\dot{Q}}$ and optimum turbulent dissipation rate $\overline{f_f}$ in Figures 6.8 to 6.19. When observing optimisation within laminar regimes –Figures 6.8, 6.10 and 6.12–, the temperature gradient –overall temperature in $Re = 500$– is greatly enhanced by introducing the right roughness parameters, but this rise seems inversely proportional to Re numbers. The pressure wall-normal gradient increases with higher Re, and has a profile that decreases towards the wall. The velocity profiles are shifted slightly towards the wall, reducing the thickness of the hydraulic boundary layer. From the variable profiles at negative $\overline{\dot{Q}}$ in Figures 6.9, 6.11 and 6.13, we see loss in advected heat $-\overline{f_h} < 0$, in this definition of f_h, $\dot{D}$ is not considered in the equation–, a significant increase in pressure values at the inlet, and a shift in the velocity profile away from the wall, not towards it. Additionally at $Re = 500$, the rough surface lowers the temperature profile at the inlet and outlet. This is contrasted with $Re = 2000$ –Figure 6.15–, where normalised advected heat $\overline{f_h}$ is positive, but normalised dissipation rate $\overline{f_f}$ is negative.
Moving to optimisation in turbulent regimes –Figures 6.16 and 6.18–, we observe an increase in advected heat $-\overline{f_h} > 1$–, the thermal and hydraulic boundary layers' thickness increases compared to the baseline case, and the buffer layer is visible at the inlet for all cases, a sign of underdeveloped turbulent flow. Pressure values of both baseline and rough cases in turbulence are significantly higher than in the laminar regime, indicating a severe pressure loss when increasing Re values. The wall-normal pressure gradient of the baseline case shows a profile that increases towards the wall in both boundaries. This indicates flow moving away from the wall, a behaviour shared by rough surfaces at the inlet and inverted at the outlet, indicating flow moving towards the wall. Additionally, the inlet wall-normal pressure gradient visibly changes for both baseline and optimum geometry at the buffer layer, which is smoothed when fully developed flow is observed at the outlet. The profiles undergo similar changes when this same geometry is studied under $Re = 2000$ flows, as shown in Figure 6.14, but with no overall THP optimisation. It is worth noting that there is optimisation in dissipation rate $-\overline{f_f} < 1$– in this scenario, but advected heat is lower than the baseline $-\overline{f_h} < 1$–, opposite to what is seen in $Re = \{7000,\ 8000\}$.

6.3.1 Discussion and findings

From the previous observations and in line with Objective VI.III, it is clear that surface roughness has an impact on the flow, and that the influence varies depending on the roughness parameters, amplitude and wavenumber in the case of this study. Note that turbulent regimes are dissipation dominated due to the generation and dissipation of turbulent kinetic energy [23, 88]. In turbulence, the average velocity increases because the profile shifts towards the wall compared to laminar regimes, as seen in Figures 6.8 to 6.19, where the baseline cases have a logarithmic curve instead of a parabolic profile. This increase in kinetic energy happens in the transition from imposed inlet parabolic profile to a logarithmic profile, and appears at the cost of pressure, where higher pressure losses can be seen with increasing values of Re. This also intensifies the shear stress due to the development of a smaller viscous sublayer with a large velocity gradient. The kinetic energy generation rate is recognised as a negative value in the dissipation rate equation, f_f.

All baseline cases in turbulent regimes yielded negative dissipation rate, which indicates generation of turbulent kinetic energy. When introducing surface roughness, depending on the geometric parameters, the dissipation values obtained were both negative and positive. A transition to positive dissipation, as seen in Figures 6.15 and 6.17, indicates that the flow is no longer generating turbulent kinetic energy, the viscous and log sublayers are fully disrupted and the logarithmic shape shifts towards a parabolic profile, reducing the shear stress and average velocity. In cases of negative f_f with extreme optimisation $\overline{f_f} \lll 1$ as seen in Figure 6.19 the velocity decays in a similar manner without transitioning to positive values. The collapse of the sublayer is associated in literature with *fully rough flow regimes* [92].
The turbulent mixing initiated by roughness induced dissipation increases the temperature gradient and thickens the thermal boundary layer, which gives the impression of better performance. This was observed by Stalio and Nobile [29], Benhalilou and Kasagi [104] and Jin [102], who report either no enhancement at low Pr or enhancement with drag penalty at high Pr values, contradicting the findings of Wiesche [103]. However, both thermal and hydraulic mechanisms are driven mostly by the velocity profile, since its decay in the outlet boundary reduces the pressure and temperature outlet values in (2.52) and (2.54), respectively, reducing the positive effects these might have in the overall performance. The roughness parameters that yielded these results are dense and have severe elevation changes, associated with the disruption in the boundary layer.
On the other hand, a dense roughness with moderate elevation changes has a lower impact on the logarithmic layer. In lower Re values this influence is contained within the viscous sublayer, increasing the thickness and reducing the shear stress and average velocity, as shown in Figure 6.14. However, at higher Re values –Figures 6.16 and 6.18–, the influence propagates inwards in the flow, increasing the average velocity and therefore, the generation rate.
The gentle disruption causes a similar behaviour in the thermal mechanism: The viscous sublayer thickness is increased in either Re values, but at higher Re th impact propagates inwards in the flow, and the profile has lower temperature values in the pipe radius range where there is an increase in velocity. However, in these Re, the thermal performance was enhanced compared to the baseline, further sustaining that these mechanisms are driven mostly by the velocity profile in turbulent regimes. Additionally, the temperature at the wall rises in all roughness cases at all turbulent Re values, potentially due to the increase in wetted surface area. This gain in temperature combined with a cautious disruption from dense, low elevation surface roughness is what leads to thermal optimisation in turbulence regimes. It is worth noting that, since this study focuses on a 2D domain, no cross-stream instabilities were analysed, and the lack of these in the simulation potentially reduces heat transfer induced by cross-stream turbulent mixing.
Laminar flows have no turbulent kinetic energy generation –$f_f > 0$ in all cases–, the pressure drop is extremely low compared to turbulent regimes, and the velocity has a parabolic profile. The interaction with rough surfaces generally induces a slightly higher pressure drop than the baseline, and causes little changes in the velocity profile. In optimum $\overline{\dot{Q}}$ cases from Figures 6.8, 6.10 and 6.12, the average velocity increases, which indicates the profile is inclining into a logarithmic shape. The changes in pressure drop and velocity profile are similarly reported in

literature [92, 94, 115, 116][3]. Opposite to what was found in turbulent regimes, the roughness parameters with optimum results are sparse and have severe elevation changes. Interestingly though, when increasing the roughness density with severe elevation changes, similarly to turbulent regimes, the flow is associated with higher dissipation rates and reduced average velocity –Figures 6.9, 6.11 and 6.13–. This yields loss in $\overline{\dot{Q}}$ performance due to the velocity decay in the outlet.

There is particular interest in $Re = 500$, where the surface roughness is reducing the overall temperature in the outlet and causing severe negative $\overline{\dot{Q}}$. The mechanism behind this significant impact in the thermal boundary layer are the eddies trapped in the roughness valleys shown in Figure 6.20, observed in previous studies [92, 94, 113]. When the feature parameters reach a critical ratio between amplitudes and wavenumbers, presented in Table 6.8[4], the eddies act as insulation rather than enhancing the thermal interaction. However, the formation of eddies is seen in the optimum $\overline{\dot{Q}}$, where the interaction enhances thermal gain. The eddies in the optimum $\overline{\dot{Q}}$ remain close to the wall, while in the minimum $\overline{\dot{Q}}$ the formation of multiple eddies with flow separation cause an insulation layer [23], which reduces the thermal exchange. It is worth noting that regardless of roughness parameter ratio, when these eddies form in the valleys they have minimum interaction with the flow, and because of this the insulation is outweighted by the turbulent mixing heat transfer above the roughness when in turbulent regimes.

Going back to Table 6.8, the critical geometric ratios for $\overline{f_f} > 1$ show a trend: the parametric region where dissipation losses are higher with flow over a rough surface compared to the baseline seems to decrease in the transition from laminar to turbulent regimes, to then rise consistently with increase in Re. This suggests that the relation between parametric region and Re is non-linear in laminar-turbulence transition, but could be linear considering only higher turbulent regimes. Finding this relation is a good use case for Machine Learning algorithms, adjacent to Objective VI.V about the limitations of high fidelity modelling. Additionally, we can observe that in flow regimes where there is no thermal enhancement –no ratios found for $f_h > 1$–, there is no enhancement in $\dot{Q}$ either, meaning that THP optimisation is predominantly influenced by enhancing f_h.

Overall, after satisfying Objective VI.III, the analysis of flow for Objective VI.IV resulted in the identification of parametric regions that allow optimisation with surface roughness. The optimisation of $\overline{\dot{Q}}$ in both laminar and turbulent regimes is mainly driven by the incremental average velocity in addition to the increase in wetted surface area for enhanced heat transfer. In turbulent regimes this is achieved with low recess, high density roughness, where there is no eddy structure formation in the valleys and the flow remains mostly attached or has early reattachment to the wall. This is to avoid disrupting the logarithmic layer, since doing so yields a decrease in average velocity. In laminar regimes the disturbance in the flow generated by deep structures enhances average velocity and thermal interaction, but the density of roughness is kept low to ensure that the trapped eddies remain in contact with the wall and the flow above

[3]Early transition due to surface roughness has not been achieved in these cases, however, further analysis in laminar regimes with turbulence modelling to account for transition must be performed in the future.

[4]Note that there is overlap between $\overline{\dot{Q}} < 0$ and $\overline{\dot{Q}} > 1$. The higher limit of $\overline{\dot{Q}} > 1$ is found in composite ratios, where $A_{1,2} \neq 0$ and $k_{1,2} \neq 0$. This ratio serves as a preliminary reference, further work on real ratios is needed to resolve the geometrical relation to flow behaviour.

Re	$\overline{f_f} > 1$	$\overline{f_h} > 1$	$\overline{\dot{Q}} > 1$	$\overline{\dot{Q}} < 0$
500	[0.2361, 1.2107]	[0.023, 0.9586]	[0.023, 0.9586]	[0.3632, 1.2107]
1000	[0.4843, 1.2107]	[0.023, 1.2084]	[0.023, 1.2084]	[0.6053, 1.2107]
1500	[0.4843, 1.2107]	[0.023, 1.2084]	[0.023, 1.2084]	[0.7264, 1.2107]
2000	[0.8475, 1.2107]	-	-	-
2500	[0.0964, 1.2107]	-	-	-
3000	[0.0964, 1.2107]	-	-	-
3500	[0.0472, 0.3264]	-	-	-
4000	[0.0472, 0.3834]	-	-	-
5000	[0.0472, 0.4833]	-	-	-
5500	[0.0472, 0.4833]	-	-	-
6500	[0.0471, 0.4833]	-	-	-
7000	[0.0467, 0.5637]	[, 0.121]	[, 0.121]	-
7500	[0.0467, 0.4833]	-	-	-
8000	[0.0467, 0.4833]	[, 0.121]	[, 0.121]	-

Table 6.8: THP optimisation benchmark of the high fidelity model - Range of critical ratios of roughness geometric parameters for scenarios of significant loss as $\overline{f_f} > 1$, $\overline{\dot{Q}} < 0$, and optimisation as $\overline{f_h} > 1$, $\overline{\dot{Q}} > 1$. The ratio is computed with the average of the corresponding parameters as $\overline{\mathrm{A}} \cdot \overline{\mathrm{k}}$, where $\overline{\mathrm{A}} = \mathrm{ave}(\mathrm{A}_1, \mathrm{A}_2)$, $\overline{\mathrm{k}} = \mathrm{ave}(\mathrm{k}_1, \mathrm{k}_2)$.

with minimum dissipation in the main flow.

In relation to Objective VI.V, finding a geometry for $\overline{\dot{Q}}$ optimisation was not possible for all Re within the parametric space studied. Each simulation has a CPU execution time in the range [2, 6] h, which resulted in an average CPU time of 40 500 h for over 10 000 simulations. This rapidly becomes unfeasible for a three-dimensional domain, necessary to study cross-stream turbulent structures and their effect on heat transfer as well as their interaction with surface roughness. The axi-symmetric domain example described in Appendix C.3 has a CPU execution time in the range of [50, 72] h, which is more than 15 times the average in two dimensions. Additionally, the cases with more aggressive elevation changes remain a challenge for the high fidelity model mesh with the current refinement level, which visibly coarsens the valleys in Figure 6.20. Further refinement on the mesh to properly model these geometries and potentially ones with higher complexity will increase the computational cost of future optimisation studies.

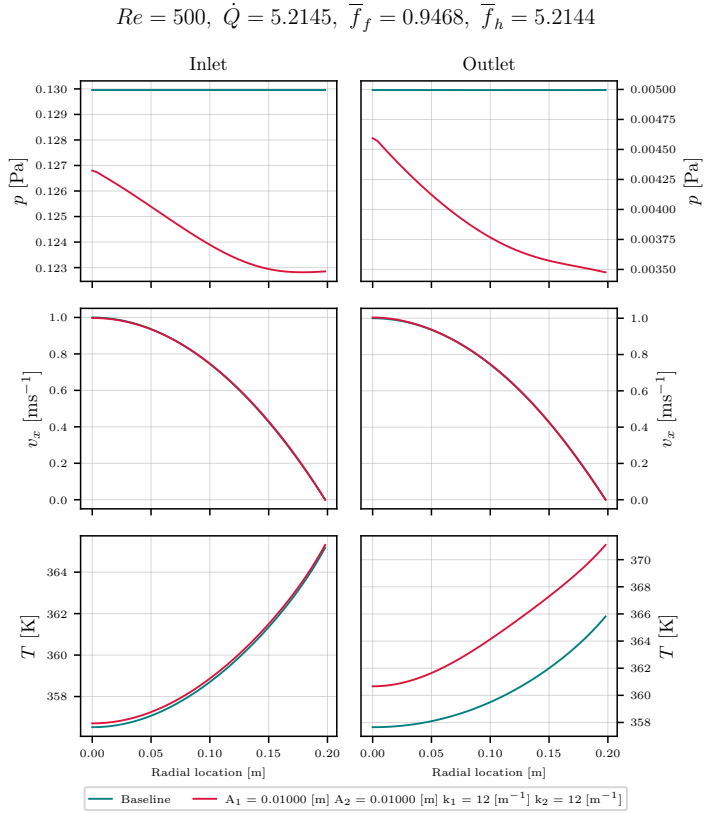


Figure 6.8: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 500$ with optimum $\overline{\dot{Q}}$.

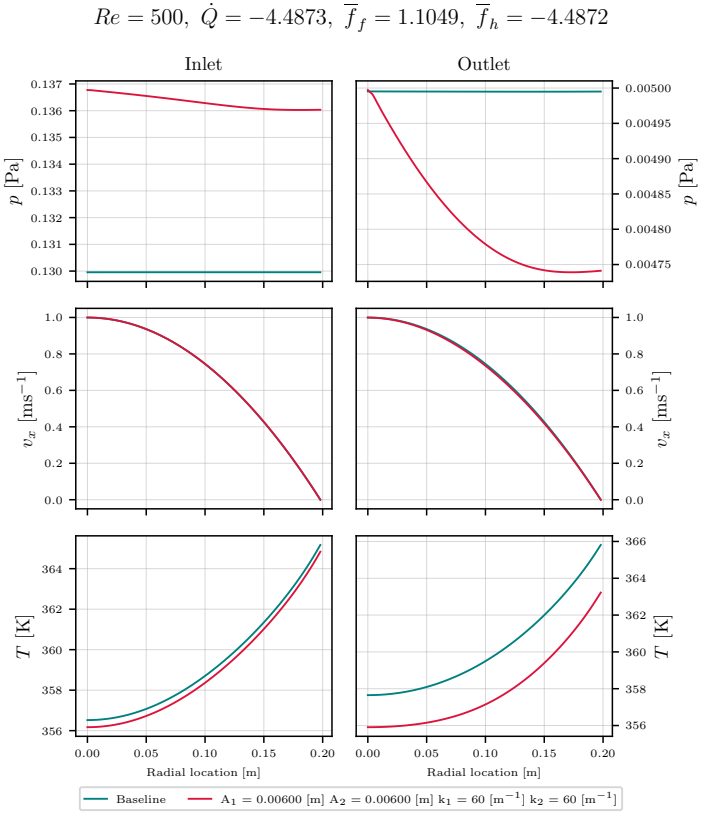


Figure 6.9: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 500$ with minimum $\overline{\dot{Q}}$.

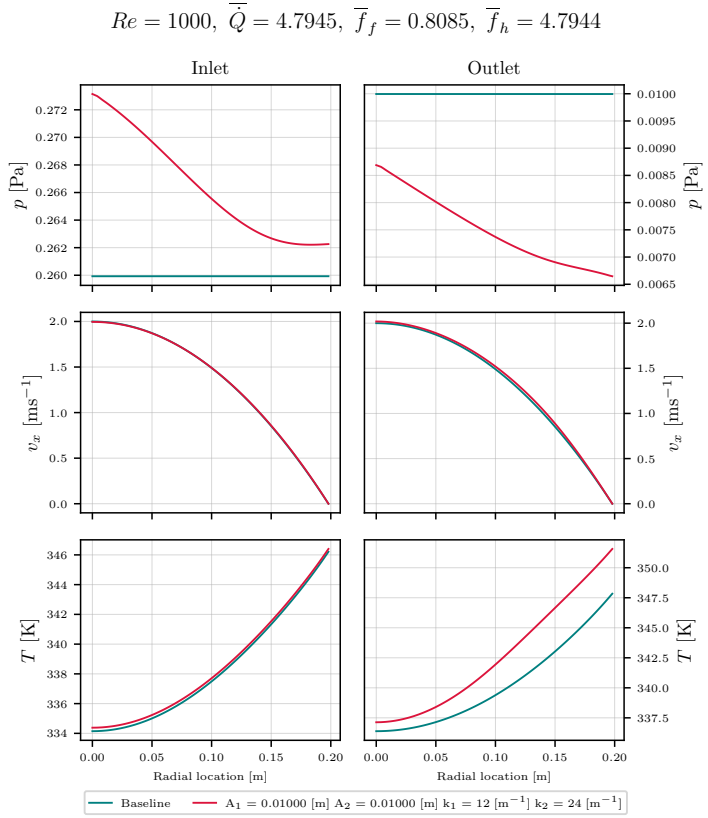


Figure 6.10: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 1000$ with optimum $\overline{\dot{Q}}$.

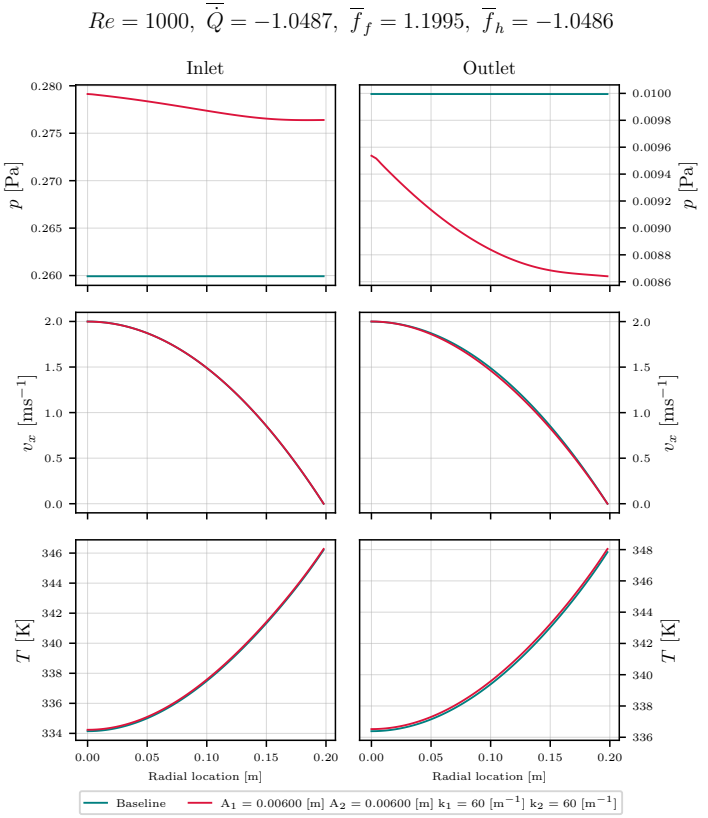


Figure 6.11: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 1000$ with minimum $\overline{\dot{Q}}$.

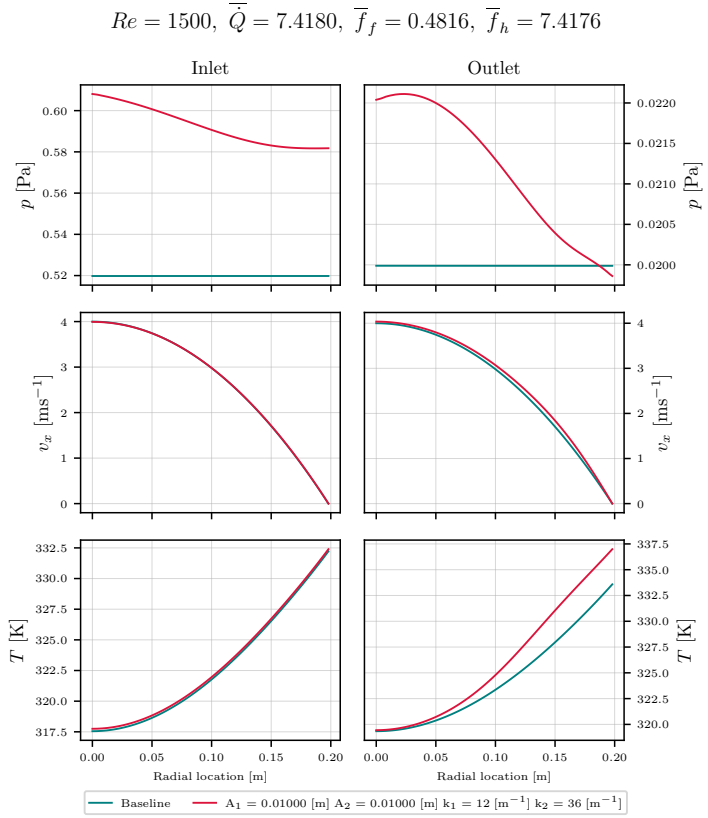


Figure 6.12: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 1500$ with optimum $\overline{\dot{Q}}$.

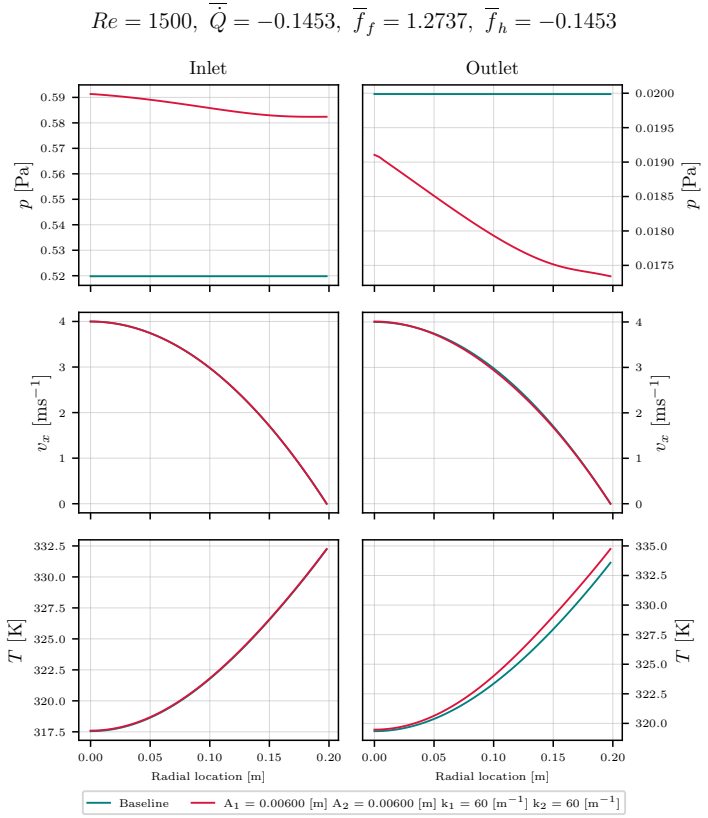


Figure 6.13: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 1500$ with minimum $\overline{\dot{Q}}$.

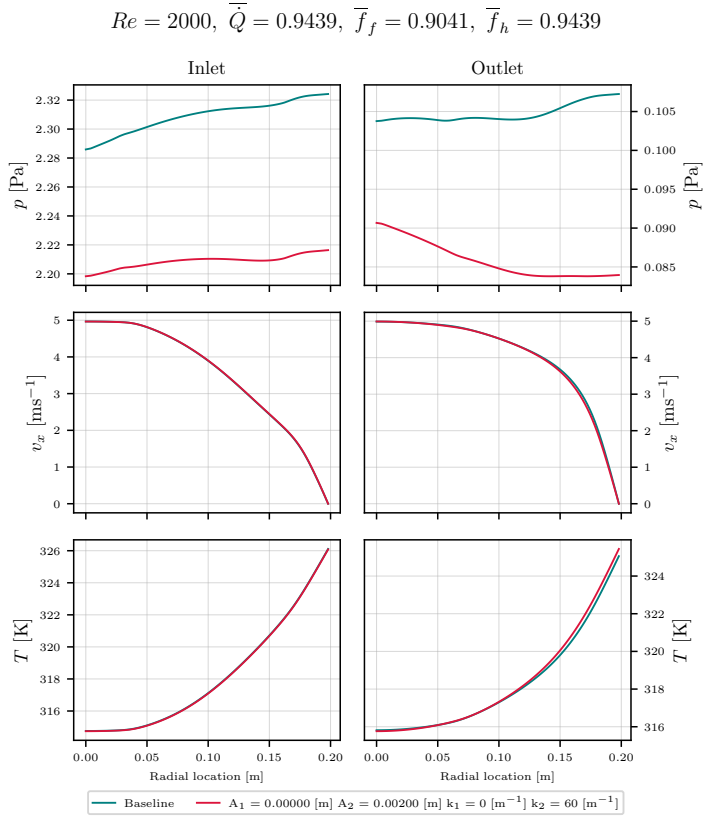


Figure 6.14: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 2000$ with near optimum $\overline{\dot{Q}}$.

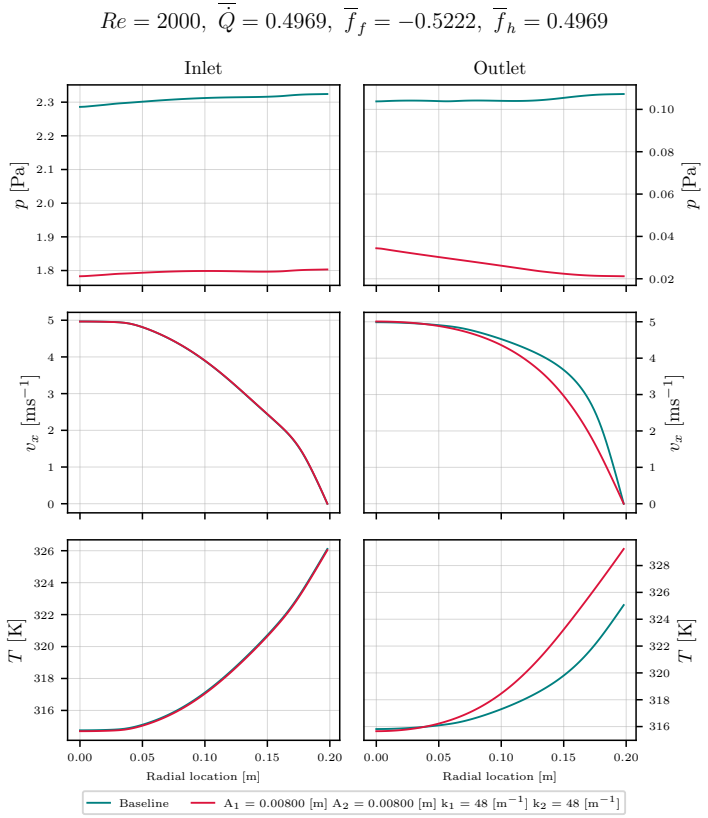


Figure 6.15: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 2000$ with minimum $\overline{\dot{Q}}$.

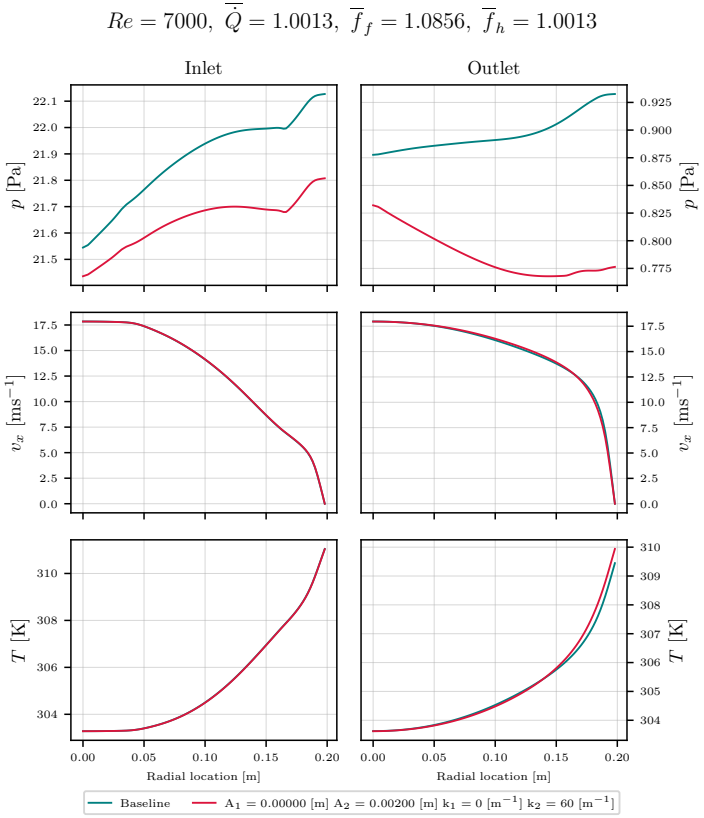


Figure 6.16: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 8000$ with optimum $\overline{\dot{Q}}$.

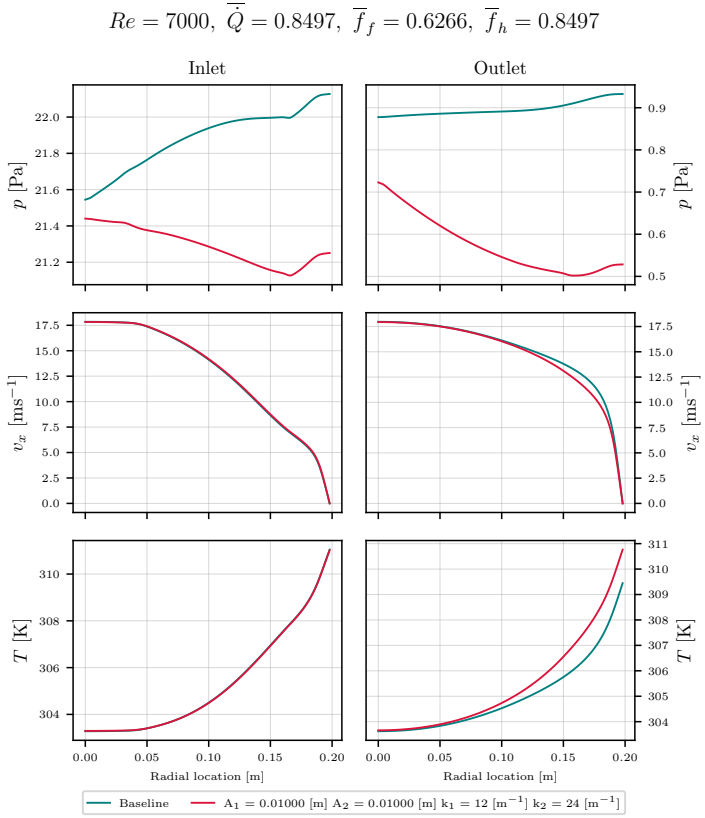


Figure 6.17: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 7000$ with optimum $\overline{f_f}$.

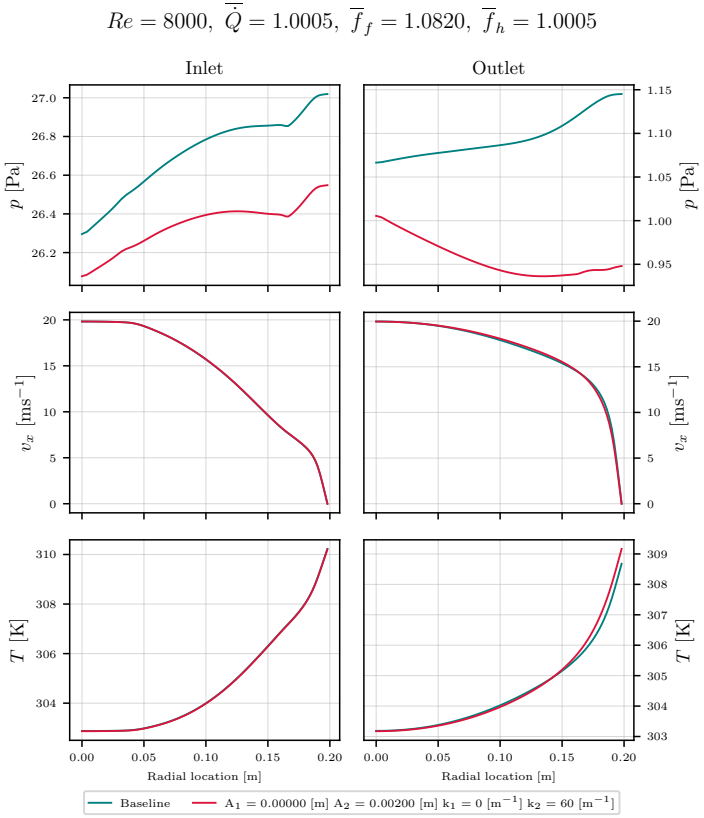


Figure 6.18: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 8000$ with optimum $\overline{\dot{Q}}$.

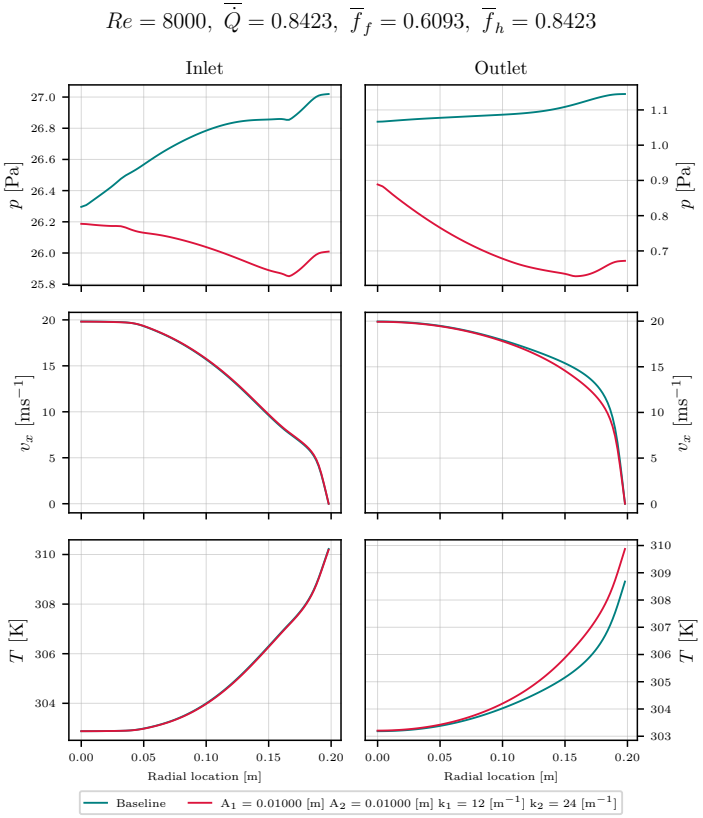


Figure 6.19: THP optimisation benchmark of the high fidelity model - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields at $Re = 8000$ with optimum $\overline{f_f}$.

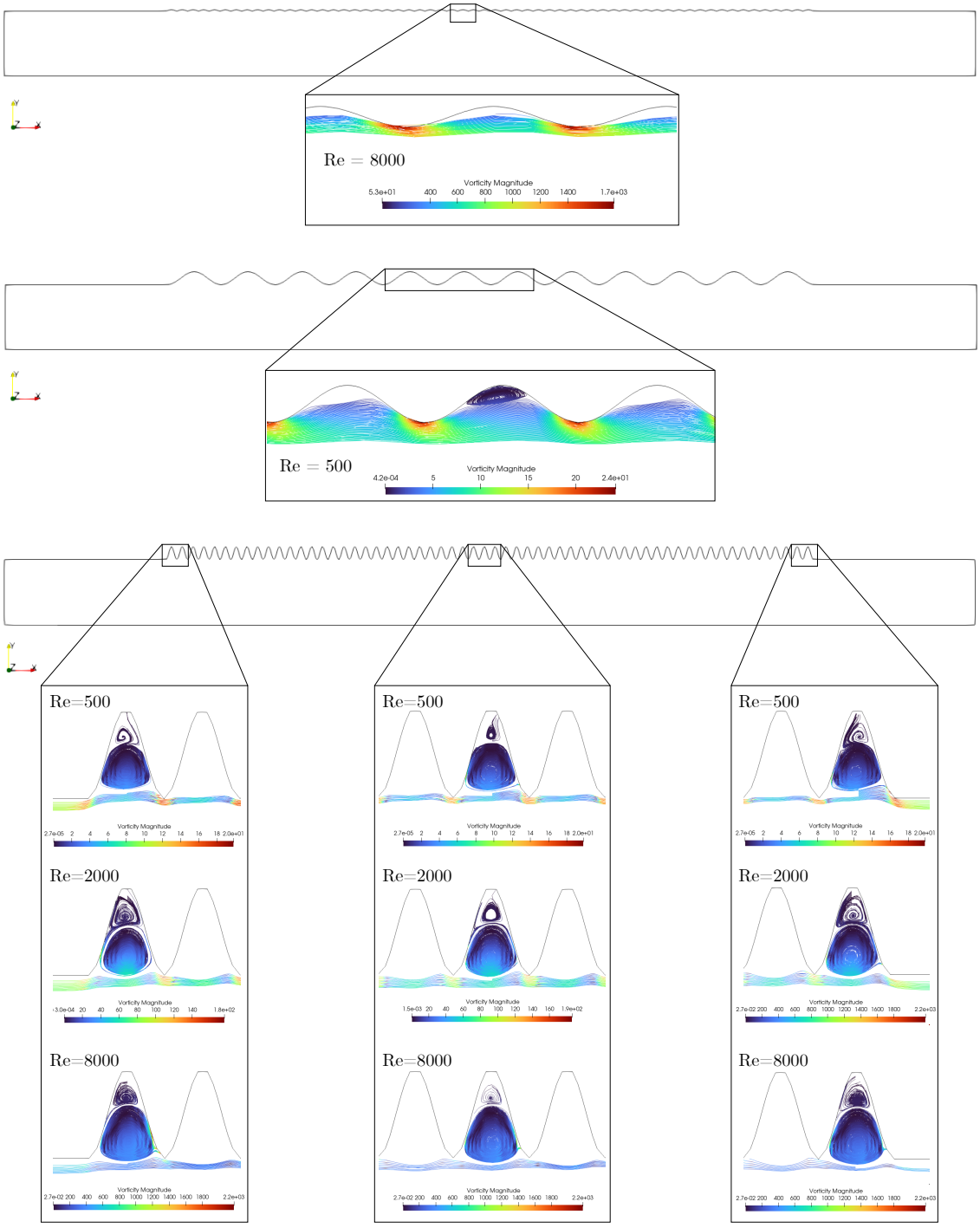


Figure 6.20: THP optimisation benchmark of the high fidelity model - Eddy formation at increasing Re for various surface roughness –top to bottom– optimum $\overline{\dot{Q}}$ at $Re = 8000$ with $A_1 = 0.002\text{m}$, $A_2 = 0.0\text{m}$, $k_1 = 60\text{m}^{-1}$, $k_2 = 0.0\text{m}^{-1}$; optimum $\overline{\dot{Q}}$ at $Re = 500$ with $A_1 = 0.01\text{m}$, $A_2 = 0.01\text{m}$, $k_1 = 12\text{m}^{-1}$, $k_2 = 12\text{m}^{-1}$; minimum $\overline{\dot{Q}}$ with $A_1 = 0.01\text{m}$, $A_2 = 0.01\text{m}$, $k_1 = 60\text{m}^{-1}$, $k_2 = 60\text{m}^{-1}$.

6.4 Summary and conclusion

This chapter presented the numerical verification and shape optimisation study of the high fidelity model representing the solution of flow in a 2D cross-section of a pipe in laminar and turbulent regimes with varying parametrised geometries over the wetted surface. In relation to Objective VI.I, Section 6.2 presents the rates of convergence of L^2 norm error of mesh with r= 2 refinement ratio for $Re = \{500,\ 8000\}$ flows. Most variables give a rate between 1 and 3, generally expected for 1st and 2nd order schemes. Laminar temperature, at rates under 1, is regarded as having enough accuracy from the validation in Appendix C.2, featuring the behaviour of thermally developed laminar layers. In line with Objective VI.II, the overall rates suggest results are mesh independent, and the 1−level refinement grid achieves the required level of accuracy for this study with respect to the computational costs. The mesh has been selected as a base for both low and high Re regimes to properly represent the smaller-scale features of the roughness.
The analysis of flow over rough surfaces satisfies Objectives VI.III and VI.IV, where it was found that increasing the wetted surface area and average velocity resulted in the optimisation of $\overline{\dot{Q}}$ in both laminar and turbulent regimes. Laminar $\overline{\dot{Q}}$ is enhanced by low density deep structures that induce eddy formation that keep flow-wall contact and low interaction with the flow above, resulting in high heat transfer and low dissipation. Turbulent optimisation of $\overline{\dot{Q}}$ is accomplished by high density roughness with shallow structures that induce small separation after each peak with early reattachment before the next to avoid high disruption to the logarithmic layer, resulting in high heat transfer and turbulent kinetic energy generation. However, optimisation in turbulent regimes was possible for $Re = \{7000, 8000\}$ only.
Further exploration of other Re to analyse $\overline{\dot{Q}}$ optimisation requires the expansion of the parametric space. This increases the computational expense of high fidelity modelling, with a current average CPU time of 40 500 h for over 10 000 simulations, a problem proposed in Objective VI.V. Additionally, element count and mesh refinement escalate with geometric complexity and the use of a three-dimensional domain to study cross-stream turbulent structures, further increasing the computational cost of the exploratory research.
The outcomes from this chapter are

Out VI.I. A robust and verified OpenFOAM mesh for a flow regime range of $Re = \{500,\ 8000\}$ to be used within the Hammerhead software as a modular base file for the high fidelity modelling approach.

Out VI.II. Validated fluid flow and heat transfer models in a pipe, represented by a 2D cross-section, to use as basis for the high fidelity modelling of flow over rough surfaces and therefore, the exploratory research of thermal and hydraulic optimisation in pipe flows.

Out VI.III. In depth analysis of the influence that surface roughness has in fluid flow and heat transfer within various Re regimes over different parametrised geometries, gathering statistics that inform about the best parametric regions for optimisation in each regime and the mechanisms behind enhancement.

Out VI.IV. An extensive database of high fidelity simulations of fluid flow over parametrised geometries to feed the machine learning algorithms in a simplified and informed manner.

CHAPTER 7

Surrogate models: benchmark and approximations

"Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful."

— George P. E. Box, 1987

7.1 Introduction

This chapter covers the design and mathematical verification of surogate models using the Machine Learning (ML) algorithms introduced in Chapter 4 to estimate the Thermo-Hydraulic Performance (THP) described in Section 2.4. This includes observations on the models' approximations across different architectural designs, as well as an evaluation of their ability to predict optimised THP feature parameters based on these approximations. The POD dimensionality reduction formulations presented in Section 4.2 and the aforementioned algorithms described in Sections 4.3, 4.4 and 4.5 are assessed in the following examples: first, the low-rank matrix approximation is generated through Singular Value Decomposition (SVD) from high-dimensional data. This method provides detailed visual truncation to determine the chosen matrix rank as described in Section 7.2. The high-dimensional matrix is computed from the two-dimensional high-fidelity data presented in Section 6.3.
Section 7.3 presents a benchmark assessment for all the proposed models with different dimensional approaches in regards of input features, output approximations and hidden architecture formulations, demonstrating the influence of matrix dimension on the robustness and accuracy of the surrogate models.
Lastly, Section 7.4 explores the use of surrogate models to predict a set of input features that maximise THP using `SciPy optimize.differential_evolution`. These results are compared with the available high fidelity data to evaluate the surrogate models' accuracy. The parameters are used to compute new simulations to verify the models' fitness at obtaining interpolated data.

7.2 Low-rank matrix representation of data

Let us explore lower order models to approximate the problem of THP evaluation for pipe flows within laminar and turbulent conditions when in contact with parametrised geometries in the wetted surface. When the optimisation problem is approached through surrogate modelling, the complexity and computational cost of the models scale up when the dimensionality of the problem is increased. To address this drawback, the Singular Value Decomposition (SVD) presented in Section 4.2 is assessed in this section as a feature extraction method, proposed to approximate lower-rank matrices from the numerical solution examples in Section 6.3. The objectives of this example are:

Obj VII.I. Description of the dimension reduction techniques presented in Section 4.2 applied to the flow variable profile matrices generated from high fidelity model data for the thermo-hydraulic behaviour study.

Obj VII.II. Assessment of the efficiency of dimension reduction techniques to formulate a low-rank matrix representation of the data, the correct implementation of said techniques and appropriate selection of σ_{N} modes.

This *parametric problem* encompasses a set of *snapshots* $\mathrm{N_s}$, the cases available from the high fidelity data, dependent on a set of input features $\boldsymbol{\omega}_{\mathrm{N}_i}$ based on the shape parameters, amplitude A_i and wavenumber k_i, and flow parameter Re. The variables of interest from

Parameter	Values	Parameter	Values
Re	$[500,\ 8000]$	$\mathrm{N_h}$	1, 2
A_i	$[0,\ 0.01]$ m	k_i	$[0, 60]$ m^{-1}

Table 7.1: Low-rank matrix representation of THP for pipe flows - Input features $\boldsymbol{\omega}$ for the $\mathbf{D}(\boldsymbol{\omega})$ function. The geometry parameters A and k with 6 values in each parameter range.

each case are considered in the set of outputs $\mathbf{D}(\boldsymbol{\omega}) \in \mathbb{R}^{\mathrm{N_s}} \times \mathbb{R}^{\mathrm{N}_x}$, which encompasses the numerical measurements at locations $\mathbf{x} \in \mathbb{R}^{\mathrm{N}_x}$. The data is arranged in matrix form such that the element D_i^j is the measurement in the $j-$th location taken with the $i-$th iteration of features $\boldsymbol{\omega}_{\mathrm{N}_i}$, where the features are considered the independent variables. There are R number of features $\boldsymbol{\omega}$ used to evaluate $\mathbf{D}$, which is varied to assess the correlation between features and the sensitivity of the matrix to each feature. The range of values adopted for each feature are listed in Table 7.1, and the sets of features evaluated in this study are

$$\boldsymbol{\omega}_2 = [\mathrm{A}_1, \mathrm{k}_1]\,, \qquad \text{when } Re = \textit{constant},\ \mathrm{N_h} = 1\,, \tag{7.1a}$$
$$\boldsymbol{\omega}_3 = [\mathrm{A}_1, \mathrm{k}_1, Re]\,, \qquad \text{when } \mathrm{N_h} = 1\,, \tag{7.1b}$$
$$\boldsymbol{\omega}_4 = [\mathrm{A}_1, \mathrm{k}_1, \mathrm{A}_2, \mathrm{k}_2]\,, \qquad \text{when } Re = \textit{constant},\ \mathrm{N_h} = 2\,, \tag{7.1c}$$
$$\boldsymbol{\omega}_5 = [\mathrm{A}_1, \mathrm{k}_1, \mathrm{A}_2, \mathrm{k}_2, Re]\,, \qquad \text{when } \mathrm{N_h} = 2\,. \tag{7.1d}$$

By applying SVD –described in Section 4.2– we obtain the low-rank matrix $\mathbf{P}_{\mathrm{N}}$ that represents the original matrix $\mathbf{D}$ in a lower dimension $\mathbf{x} \in \mathbb{R}^k$ where $k \lll \mathrm{N}_x$. The singular values of $\boldsymbol{\Sigma} = [\sigma_1, \sigma_2, ..., \sigma_R]$ for $R = 50$, arranged naturally in decreasing order, are shown in Figures 7.1 to 7.6. The tolerance cut-off for the low-rank matrix is the visual valley of σ where it exists. Datasets $\boldsymbol{\omega}_2$ and $\boldsymbol{\omega}_4$ have the tolerance cut-off between σ_5 to σ_{15}, where higher Re matrices have an outlet pressure p without a visible valley and an outlet temperature T visibly outside of said range. When including Re for datasets $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$, the average tolerance cut-off lands around σ_{10} and σ_{20}, with similar behaviour in outlet p and T. The relative error for the reconstruction data is computed as

$$\mathrm{Er}_\phi = \frac{\mathrm{abs}\left(\phi - \overline{\phi}\right)}{\overline{\phi}}\,, \tag{7.2}$$

where ϕ and $\overline{\phi}$ are the reconstructed and high-fidelity variables, respectively. The profiles with $\max(\mathrm{Er}_\phi)$ from datasets $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$ are presented in Figures 7.7 to 7.10. Considering the observed difference and tolerance, the selected amount of σ_{N} to construct low-rank matrices for this study is 5 for $\boldsymbol{\omega}_2$ and $\boldsymbol{\omega}_4$, and 20 for $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$.

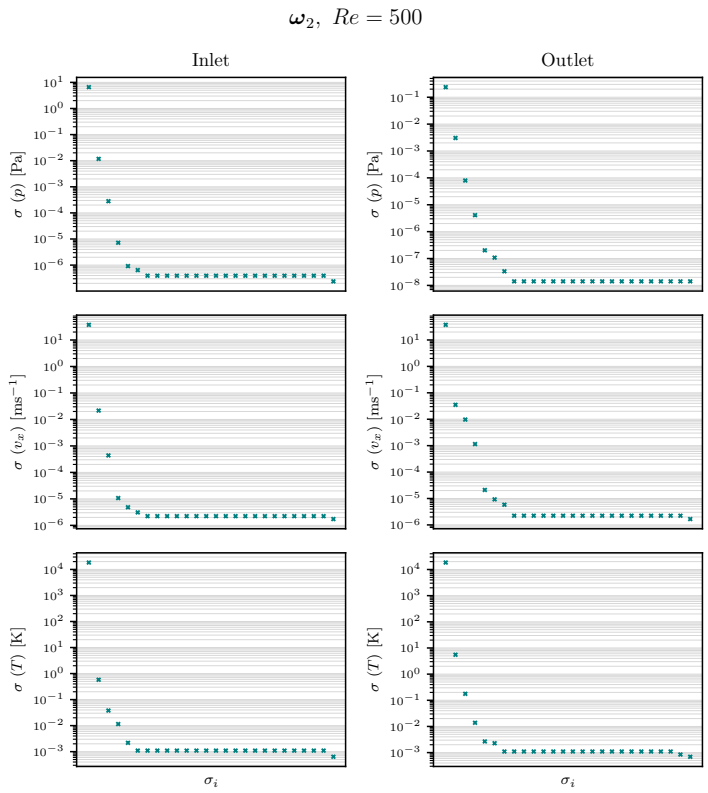


Figure 7.1: Low-rank matrix representation of THP for pipe flows - Non-zero singular values σ_i of the different matrices of –left to right, top to bottom– inlet and outlet pressure $\mathbf{D}^{p}(\boldsymbol{\omega})$, inlet and outlet velocity $\mathbf{D}^{v_x}(\boldsymbol{\omega})$, inlet and outlet temperature $\mathbf{D}^{T}(\boldsymbol{\omega})$ based on dataset $\boldsymbol{\omega}_2$.

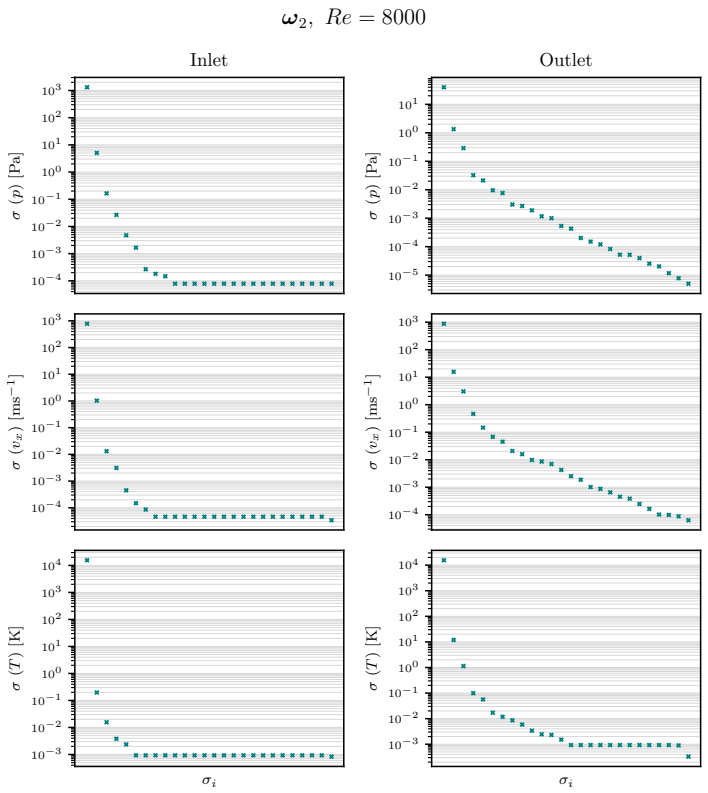


Figure 7.2: Low-rank matrix representation of THP for pipe flows - Non-zero singular values σ_i of the different matrices of –left to right, top to bottom– inlet and outlet pressure $\mathbf{D}^{p}(\boldsymbol{\omega})$, inlet and outlet velocity $\mathbf{D}^{v_x}(\boldsymbol{\omega})$, inlet and outlet temperature $\mathbf{D}^{T}(\boldsymbol{\omega})$ based on dataset $\boldsymbol{\omega}_2$.

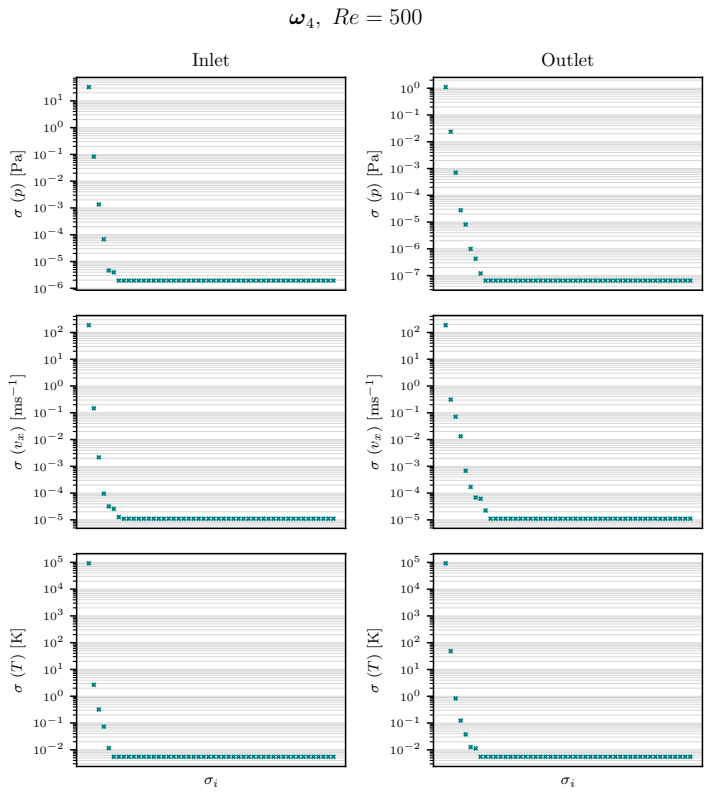


Figure 7.3: Low-rank matrix representation of THP for pipe flows - Non-zero singular values σ_i of the different matrices of –left to right, top to bottom– inlet and outlet pressure $\mathbf{D}^p(\boldsymbol{\omega})$, inlet and outlet velocity $\mathbf{D}^{v_x}(\boldsymbol{\omega})$, inlet and outlet temperature $\mathbf{D}^T(\boldsymbol{\omega})$ based on dataset $\boldsymbol{\omega}_4$.

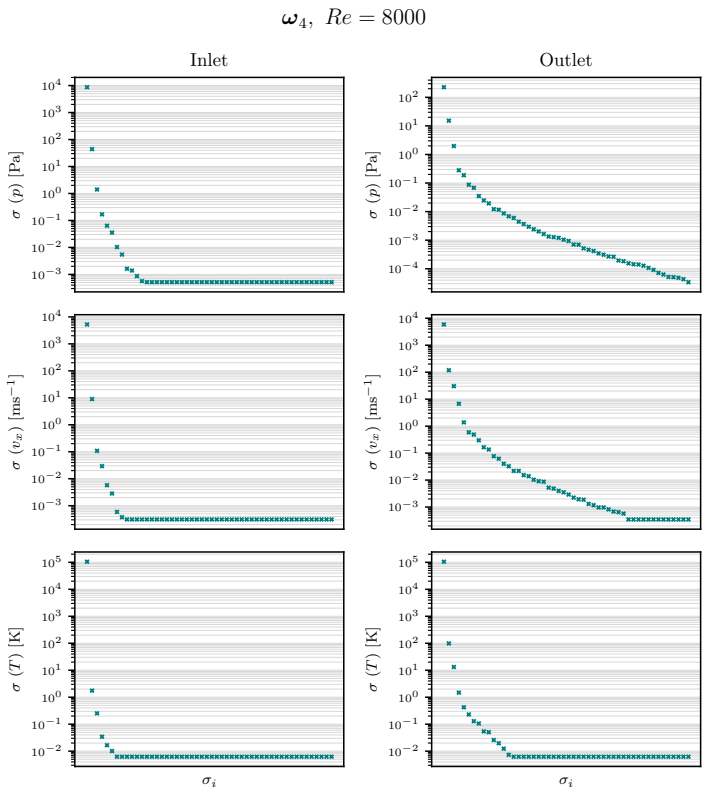


Figure 7.4: Low-rank matrix representation of THP for pipe flows - Non-zero singular values σ_i of the different matrices of –left to right, top to bottom– inlet and outlet pressure $\mathbf{D}^p(\boldsymbol{\omega})$, inlet and outlet velocity $\mathbf{D}^{v_x}(\boldsymbol{\omega})$, inlet and outlet temperature $\mathbf{D}^T(\boldsymbol{\omega})$ based on dataset $\boldsymbol{\omega}_4$.

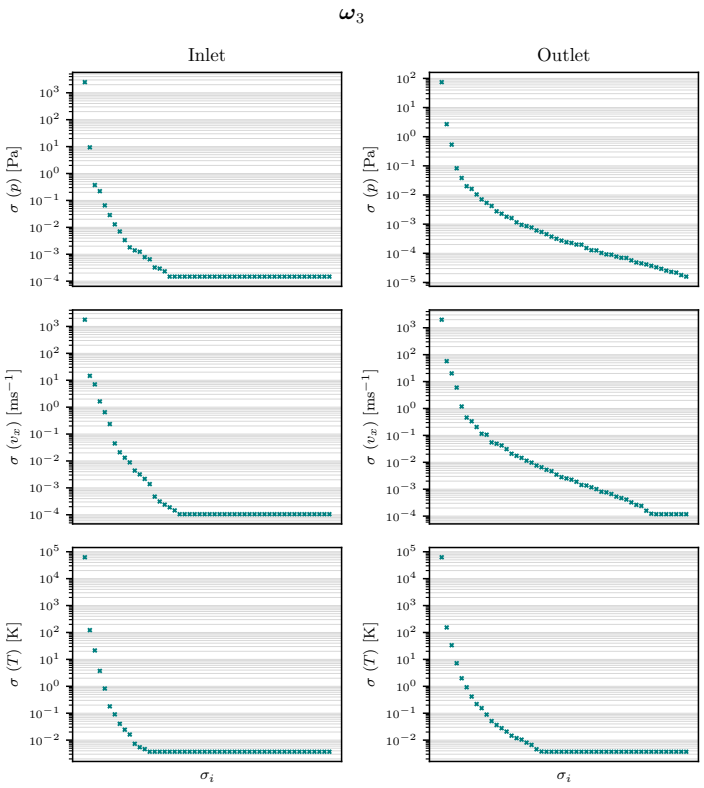


Figure 7.5: Low-rank matrix representation of THP for pipe flows - Non-zero singular values σ_i of the different matrices of –left to right, top to bottom– inlet and outlet pressure $\mathbf{D}^{p}(\boldsymbol{\omega})$, inlet and outlet velocity $\mathbf{D}^{v_x}(\boldsymbol{\omega})$, inlet and outlet temperature $\mathbf{D}^{T}(\boldsymbol{\omega})$ based on dataset $\boldsymbol{\omega}_3$.

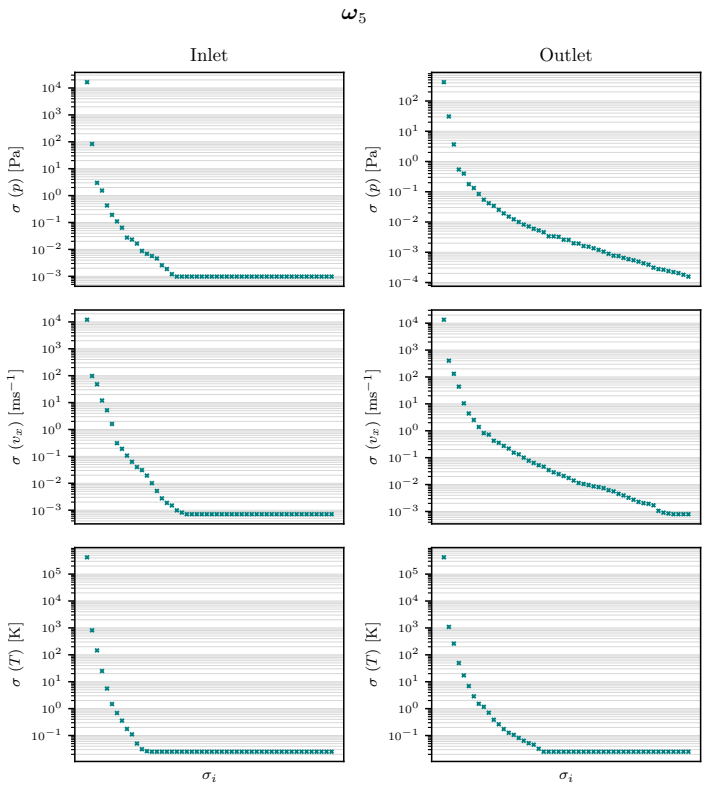


Figure 7.6: Low-rank matrix representation of THP for pipe flows - Non-zero singular values σ_i of the different matrices of –left to right, top to bottom– inlet and outlet pressure $\mathbf{D}^{p}(\boldsymbol{\omega})$, inlet and outlet velocity $\mathbf{D}^{v_x}(\boldsymbol{\omega})$, inlet and outlet temperature $\mathbf{D}^{T}(\boldsymbol{\omega})$ based on dataset $\boldsymbol{\omega}_5$.

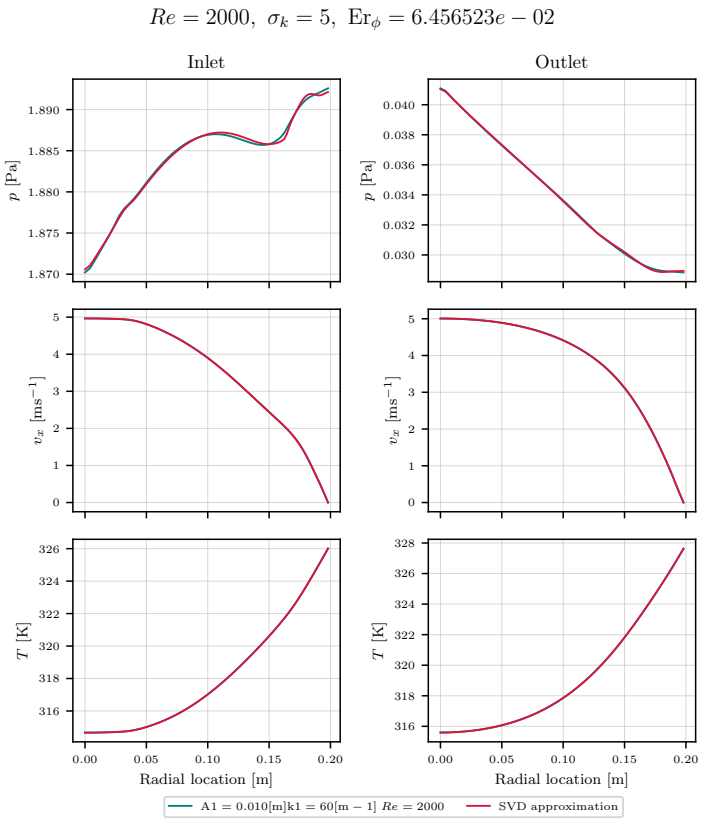


Figure 7.7: Low-rank matrix representation of THP for pipe flows - Low-rank matrix reconstruction of –left to right, top to bottom– inlet and outlet pressure $\mathbf{D}^{p}(\boldsymbol{\omega})$, inlet and outlet velocity $\mathbf{D}^{v_x}(\boldsymbol{\omega})$, inlet and outlet temperature $\mathbf{D}^{T}(\boldsymbol{\omega})$ from the matrix $\sigma_k = 5$ of dataset $\boldsymbol{\omega}_3$. The profiles shown were found to have the highest relative error $\max(\mathrm{Er}_\phi)$ among the snapshots within the evaluated tensor.

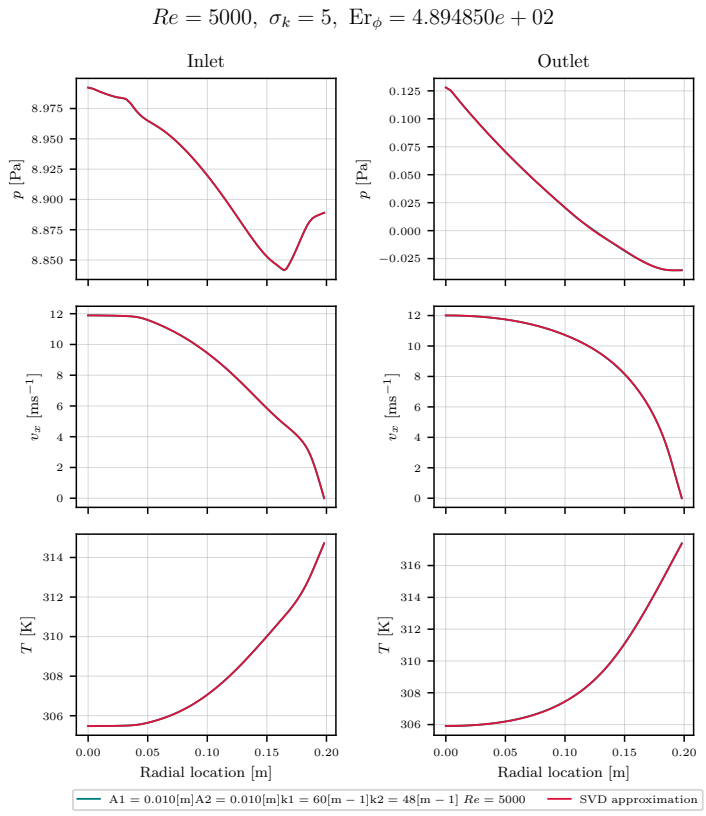


Figure 7.8: Low-rank matrix representation of THP for pipe flows - Low-rank matrix reconstruction of –left to right, top to bottom– inlet and outlet pressure $\mathbf{D}^{p}(\boldsymbol{\omega})$, inlet and outlet velocity $\mathbf{D}^{v_x}(\boldsymbol{\omega})$, inlet and outlet temperature $\mathbf{D}^{T}(\boldsymbol{\omega})$ from the matrix $\sigma_k = 5$ of dataset $\boldsymbol{\omega}_5$. The profiles shown were found to have the highest relative error $\max(\mathrm{Er}_\phi)$ among the snapshots within the evaluated tensor.

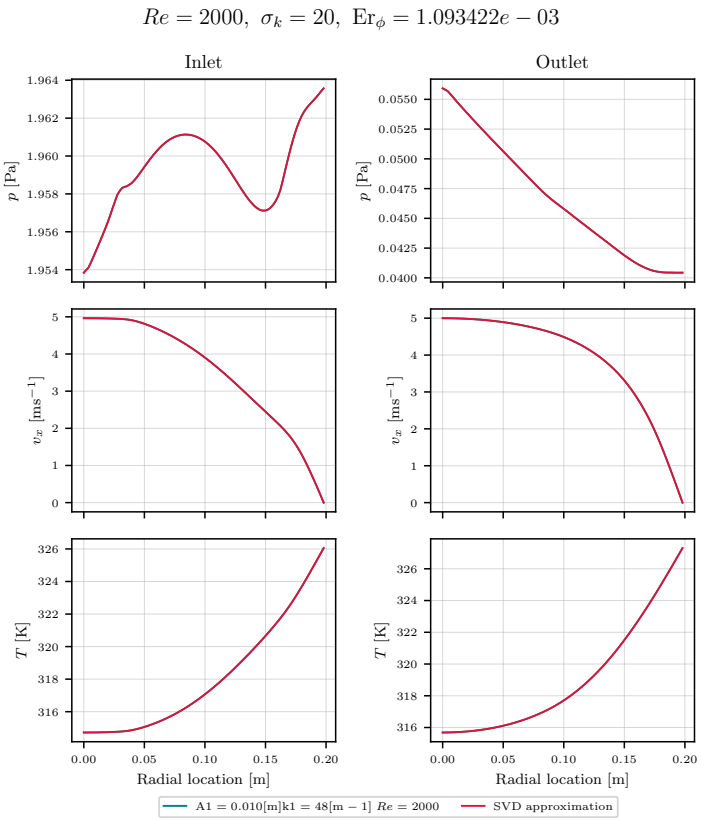


Figure 7.9: Low-rank matrix representation of THP for pipe flows - Low-rank matrix reconstruction of –left to right, top to bottom– inlet and outlet pressure $\mathbf{D}^{p}(\boldsymbol{\omega})$, inlet and outlet velocity $\mathbf{D}^{v_x}(\boldsymbol{\omega})$, inlet and outlet temperature $\mathbf{D}^{T}(\boldsymbol{\omega})$ from the matrix $\sigma_k = 20$ of dataset $\boldsymbol{\omega}_3$. The profiles shown were found to have the highest relative error $\max(\mathrm{Er}_\phi)$ among the snapshots within the evaluated tensor.

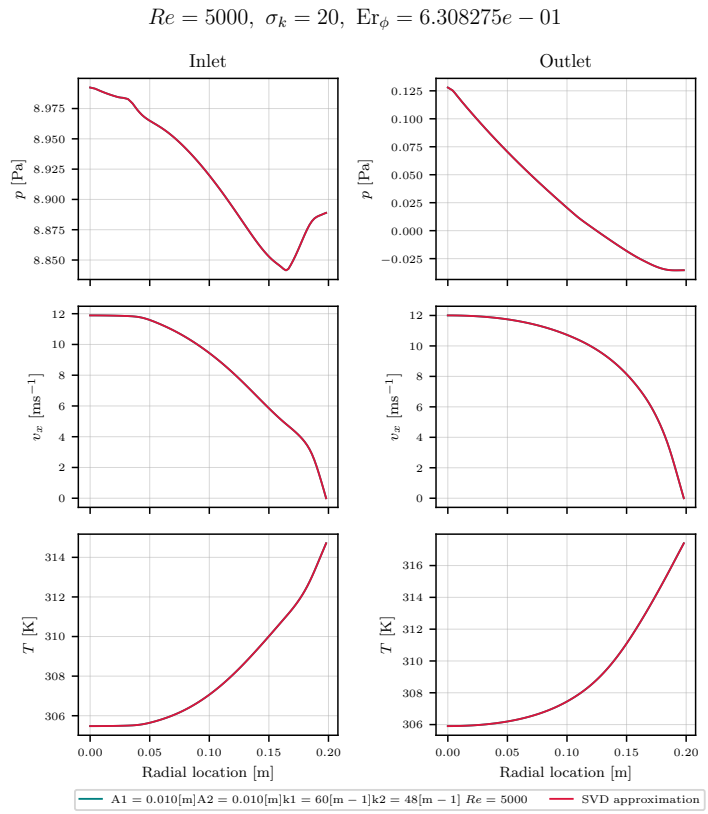


Figure 7.10: Low-rank matrix representation of THP for pipe flows - Low-rank matrix reconstruction of –left to right, top to bottom– inlet and outlet pressure $\mathbf{D}^{p}(\boldsymbol{\omega})$, inlet and outlet velocity $\mathbf{D}^{v_x}(\boldsymbol{\omega})$, inlet and outlet temperature $\mathbf{D}^{T}(\boldsymbol{\omega})$ from the matrix $\sigma_k = 20$ of dataset $\boldsymbol{\omega}_5$. The profiles shown were found to have the highest relative error $\max(\mathrm{Er}_\phi)$ among the snapshots within the evaluated tensor.

7.2.1 Discussion and findings

In line with Objective VII.I, this section presented the use of SVD methods to obtain low-rank matrices that represent the datasets of flow variables from different rough surface geometries interacting with the flow in a lower dimension. The low-dimension matrices are adopted to obtain values at new, unseen rough surface cases, and the new low-dimensional data is used to reconstruct the flow variables of the geometry with no high fidelity data required. The advantage of these methods is to reduce the computational cost of interpolation and other surrogate modelling algorithms that perform the calculations to obtain the data at new locations.

The performance of the dimensionality reduction method for flow variables $\mathbf{D}\,(\mathrm{A}_1, \mathrm{k}_1, \mathrm{A}_2, \mathrm{k}_2, Re)$ depending on shape and flow features $\boldsymbol{\omega}$ was verified through an eigen-analysis, varying the amount of independent features –$\boldsymbol{\omega}_2$ to $\boldsymbol{\omega}_5$–, where only either one or two sets of shape parameters were considered –$\boldsymbol{\omega}_2 = [\mathrm{A}_1, \mathrm{k}_1]$ or $\boldsymbol{\omega}_4 = [\mathrm{A}_1, \mathrm{k}_1, \mathrm{A}_2, \mathrm{k}_2]$–, or where the flow parameter is included –$\boldsymbol{\omega}_3 = [\mathrm{A}_1, \mathrm{k}_1, Re]$ or $\boldsymbol{\omega}_5 = [\mathrm{A}_1, \mathrm{k}_1, \mathrm{A}_2, \mathrm{k}_2, Re]$–. It is determined that the data has high correlation between shape parameters, amplitude and wavenumber, even when evaluating $\boldsymbol{\omega}_4$, where the tolerance cut-off was similar to $\boldsymbol{\omega}_2$ at σ_5 to σ_{15}. The correlation of these parameters with Re is low, resulting in a tolerance cut-off between σ_{10} and σ_{20} for both datasets that included Re, $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$, to reduce the error at the reconstruction stage. The dimensionality of the matrices with $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$ features is still reduced, which satisfies Objective VII.II, but the rank of the new matrix is 4 times higher than that of matrices with $\boldsymbol{\omega}_2$ and $\boldsymbol{\omega}_4$ features to produce reliable reconstructions. This means that the rank of the new matrices scales up faster when there is low correlation between features, lowering the efficiency of the method. However, the rank is still significantly lower than the original dimensions of the data, which is $\mathrm{N}_x = 100$, fulfilling Objective VII.II.

7.3 Benchmark for the surrogate approximation problem

Let us explore lower order models to approximate the problem of THP evaluation for pipe flows within laminar and turbulent conditions when in contact with parametrised geometries in the wetted surface. The data-driven surrogate models assessed in this section are based on the algorithms presented in Chapter 4, proposed to approximate the optimisation problem from the numerical solution examples in Section 6.3.

The algorithms include Radial Basis Function (RBF) interpolation, Neural Networks (NN) and Gaussian Processes (GP), each with their advantages and drawbacks. RBF is a kernel-based, *linear* interpolation method, and a low cost, effective tool used to approximate high dimensionality problems. However, this method can require a large dataset for non-linear problems. NNs have deep architectures that allow them to model non-linear problems with relatively limited datasets. Finally, GP is a kernel-based method that models the probability distribution fitting a given dataset, a useful feature to understand the model's *confidence* in the parametric space and the regions where more data is required to improve accuracy.

The complexity of each model scales differently for increasing levels of non-linearity and dimensionality in the problem. While RBF has good performance for high dimensionality problems, if the correlation between the data is non-linear it hinders the accuracy of the model

for unseen cases. GP has a similar behaviour, with the addition of a cubic computational complexity because of the use of the covariance matrix, which limits the amount of features and dimensions it can model. However, the problem can be approached with the aid of the confidence region, refining the data sampling in parametric areas the model is struggling with. On the other hand, NNs are less sensitive to non-linearities and problem dimension compared to RBF and GP, but their architecture and computational cost of training scales up with these problem characteristics. Therefore, this section presents the models' performance assessment when training with varying amounts of features $\boldsymbol{\omega}$ and for highly dimensional problems compared to low and reduced dimension matrices computed in Section 7.2.
An error measurement is necessary to evaluate the models' sensitivity to these dimensions. A study on the architecture of each model is conducted to achieve the ideal complexity each surrogate model requires to produce an accurate solution to the problem while maintaining a relative simplicity. The objectives of this example are:

Obj VII.III. Description of the surrogate models' input and output dimensions based on the parameters and variables of interest, address the training method and architectures introduced in Chapter 4.

Obj VII.IV. Compare the advantages, limitations and suitability of the surrogate models for the prediction of flow variable profiles and thermo-hydraulic behaviour against high fidelity models and each other in terms of physical accuracy and computational cost.

Obj VII.V. Assessment of the efficiency of the model's architectures and correct implementation of kernels, activation functions, dimension reduction techniques and training batch size.

Parameter	**Values**	**Parameter**	**Values**
Re	$\{500,\ 8000\}$	$\mathrm{N_h}$	$1,\ 2$
A_i	$[0,\ 0.01]$ m	k_i	$[0,\ 60]$ m^{-1}

Table 7.2: Surrogate model's benchmark - Range of values of the input features $\boldsymbol{\omega}$ matching the available data from the matrix $\mathbf{D}(\boldsymbol{\omega}, \mathbf{x})$. The geometry parameters A_H and k_H are referred to in discrete intervals, with 6 values in each parameter range.

The *parametric problem* is comprised of a set of *snapshots* $\mathrm{N_s}$, referring to the set of cases available from the high fidelity data. Since these were computed with varying surface roughness and flow regimes, each case is dependent on a set of input features $\boldsymbol{\omega}_{\mathrm{N}_i}$ with $_{\mathrm{N}_i}$ parameters based on shape, amplitude A_i and wavenumber k_i, and flow parameter Re. The variables of interest from each case are considered in the set of outputs $\mathbf{D}(\boldsymbol{\omega})$, which encompasses the numerical measurements at locations $\mathbf{x} \in \mathbb{R}^{\mathrm{N}_x}$. The data is arranged such that the element D_i^j is the measurement in the $j-$th location taken with the $i-$th iteration of features $\boldsymbol{\omega}_{\mathrm{N}_i}$, where the features are considered the independent variable.
The amount of measurement locations depends on the dimensionality of the results N_x, based

on the variables of interest defined by the flow variables $\mathbf{x}^{\Gamma_f} \in \mathbb{R}^{100}$ measured at the mesh element centre, the THP evaluation $\mathbf{x}^{\dot{Q}} \in \mathbb{R}^1$, referring to the computed dissipation rate f_f and advected heat flux f_h –in this description f_h neglects $\dot{D}$– from (2.52) and (2.54), and finally a dimension reduction of the flow variables $\mathbf{x}^{\mathbf{P}} \in \mathbb{R}^{\sigma_k}$, computed through SVD and assessed in Section 7.2. The output matrices then take the form of

$$\mathbf{x}^{\dot{Q}} = [f_f, f_h]\,, \qquad \text{then } \mathrm{N}_x = 1\,, \tag{7.3a}$$

$$\mathbf{x}^{\Gamma_f} = \left[{v_x}^{in}, T^{in}, p^{in}, {v_x}^{out}, T^{out}, p^{out}\right], \quad \text{then } \mathrm{N}_x = 100\,, \tag{7.3b}$$

$$\mathbf{x}^{\mathbf{P}} = \left[\mathbf{P}^{in}_{v_x}, \mathbf{P}^{in}_{T}, \mathbf{P}^{in}_{p}, \mathbf{P}^{out}_{v_x}, \mathbf{P}^{out}_{T}, \mathbf{P}^{out}_{p}\right], \text{then } \mathrm{N}_x = 5, 20\,, \tag{7.3c}$$

where f_f is the dissipation rate, f_h is the advected heat flux, v_x is the stream-wise velocity, T is the temperature, p is the pressure, $\mathbf{P}$ are the POD modes for each flow variable, and the superscripts *in* and *out* refer to the variable profiles taken at the inlet and outlet, respectively. The proposed $\mathbf{x}^{\dot{Q}}$, referred to as *lumped* (L-), has the smallest dimension $\mathrm{N}_x = 1$ –two surrogate models result from this output, one for each THP evaluation, f_f and f_h–, introducing low cost and complexity for the surrogate models. However, the initial drawbacks of the lumped output resides in the absence of primary flow variables to approximate, which when the THP evaluation changes or the flow variables are required, these surrogate models are no longer applicable. The $\mathbf{x}^{\Gamma_f}$ matrix, referred to as *spatial* (S-), has a dimension of $\mathrm{N}_x = 100$ data points based on the flow variable's mesh inlet and outlet boundary values –six surrogate models result from this output, one for each variable profile–. This allows the surrogate models to be informed of the primary flow variables, yet the high dimensionality of data also rises the computational cost of regression in statistical learning and potentially the architecture complexity. The last output considered is $\mathbf{x}^{\mathbf{P}}$, referred to as *modal* (M-), computed through SVD in Section 7.2, which reduces the complexity of the spatial matrices by providing a compact form of the primary flow variable boundary profiles. These models are designed to predict for the most prominent *modes* of each flow variable, keeping a lower-rank, mesh independent matrix. This reduces the computational cost and complexity of the surrogate models compared to spatial matrices while maintaining the flow variable information, unlike the lumped output. The *modes* are then used to reconstruct the variable profiles and to compute the THP mechanisms in the post-processing. Furthermore, we observed in Section 7.2 that the correlation between features has an impact on the non-linearity of the output matrices, and therefore, the overall problem. Likewise, the amount of features and their correlation affects the architecture and computational cost of the surrogate models. The range of values adopted for each feature are listed in Table 7.2, and the sets of features used to evaluate the performance of the surrogate models in this study are

$$\boldsymbol{\omega}_2 = [\mathrm{A}_1, \mathrm{k}_1]\,, \qquad \text{when } Re = constant,\ \mathrm{N_h} = 1\,, \tag{7.4a}$$

$$\boldsymbol{\omega}_3 = [\mathrm{A}_1, \mathrm{k}_1, Re]\,, \qquad \text{when } \mathrm{N_h} = 1\,, \tag{7.4b}$$

$$\boldsymbol{\omega}_4 = [\mathrm{A}_1, \mathrm{k}_1, \mathrm{A}_2, \mathrm{k}_2]\,, \qquad \text{when } Re = constant,\ \mathrm{N_h} = 2\,, \tag{7.4c}$$

$$\boldsymbol{\omega}_5 = [\mathrm{A}_1, \mathrm{k}_1, \mathrm{A}_2, \mathrm{k}_2, Re]\,, \qquad \text{when } \mathrm{N_h} = 2\,. \tag{7.4d}$$

For the solution approximation by means of surrogate modelling, each iteration of input-output sets $\mathbf{D}_i^{\mathrm{N}_x}(\boldsymbol{\omega}_{\mathrm{N}_i})$ is used to train the three methods of interest: RBF interpolation, NN and GP. NN and GP follow a similar process when training to produce a surrogate model, with the next general steps

1. Randomly select values for the unknown weight coefficients $\mathbf{W}_{\mathrm{N}_i}$,
2. Compute the prediction $\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}(\boldsymbol{\omega}_{\mathrm{N}_i}, \mathbf{W}_{\mathrm{N}_i})$ with the corresponding architecture,
3. Compare the prediction $\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}$ with the training data output $\mathbf{D}_{\mathrm{N}_i}^{\mathrm{N}_x}$ by computing the loss J if NN, and the log likelihood $\mathrm{L_J}$ if GP,
4. Compute the gradient estimate of J or $\mathrm{L_J}$ in terms of the weight coefficients $\mathbf{W}_{\mathrm{N}_i}$ as $\frac{\partial \mathrm{J}}{\partial \mathbf{W}_{\mathrm{N}_i}}$, $\frac{\partial \mathrm{L_J}}{\partial \mathbf{W}_{\mathrm{N}_i}}$,
5. Update the weight coefficients $\mathbf{W}_{\mathrm{N}_i}$ from the gradient estimate,
6. Repeat from step 2. until target loss has been achieved, or the current iteration reaches the maximum *epoch*.

RBF on the other hand, has a simple linear system of equations, and the only training step involves solving the linear system for $\mathbf{W}_{\mathrm{N}_i}$ when $\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x} = \mathbf{D}_{\mathrm{N}_i}^{\mathrm{N}_x}$. Finally, the performance of the models is assessed through 6 characteristics

- **Convergence.** Reach a target loss J computed from (4.14), or for GP the log likelihood $\mathrm{L_J}$ computed from (4.36).
- **Consistency.** Minimise local convergence with random initialisation of the weight parameters. Each architecture is trained with randomly initialised weight parameters 4 independent times to evaluate this aspect.
- **Over-fitting.** Minimising the relative error Er_r of the prediction against high fidelity data in cases used for training and validation –data ignored when training the model–.
- **Data cost and sensitivity.** Limit at which the convergence of the model does not improve with increasing training data and how the amount of training data affects the model characteristics. A set of *validation split* is examined where the validation percentage refers to the data from the matrix **ignored when training the model**. The selection of data by this percentage is random, however, the models always train with the boundary values of the parameter space to avoid extrapolation. Note that at higher $\boldsymbol{\omega}_{\mathrm{N}_i}$ with the same percentage, the amount of data available is larger, since the complete dataset for $\boldsymbol{\omega}_5$ has about 10 000 cases, and the datasets for lower features are taken from the available 10 000 cases.
- **Computational cost.** Scalability of training and prediction cost with increasing model complexity. Note that this analysis regards the CPU time of the model per variable, where $\mathbf{y}_{\mathrm{N}_i}^{\dot{Q}}$ has 2 variables, and $\mathbf{y}_{\mathrm{N}_i}^{\dot{\Gamma}_f}$ and $\mathbf{y}_{\mathrm{N}_i}^{\mathbf{P}}$ have 6 variables each.
- **Complexity sensitivity.** Scalability of the model architecture and their performance characteristics when increasing the input feature $\boldsymbol{\omega}$ and output $\mathbf{x}$ dimensions and non-linear correlation. This is addressed in the discussion of preliminary findings.

These characteristics provide a general understanding of the models capability to address the THP evaluation for pipe flows in contact with rough surfaces, and serve as a reference to outline the best $\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}\left(\boldsymbol{\omega}_{\mathrm{N}_i}, \mathbf{x}^{\mathrm{N}_x}\right)$, architecture and algorithm to fit the problem from the evaluated global parameters. $Re = \{500,\ 8000\}$ serve as the main examples in this section to compare the models' performance in laminar and turbulent regimes. Note that at higher $\boldsymbol{\omega}_{\mathrm{N}_i}$ with the same percentage, the amount of data available is larger, since the complete dataset for $\boldsymbol{\omega}_5$ has about 10 000 cases, and the datasets for lower features are taken from the available 10 000 cases.

7.3.1 Radial Basis Function interpolation

Parameter	
Kernels [127]	{Multiquadric (M), Inverse Multiquadric (IM), Inverse Quadratic (IQ), Linear (L), Gaussian (G)}
Validation/training split	{20%, 35%, 50%, 65%, 80% 95%}

Table 7.3: RBF benchmark - Numerical parameters for the radial basis function interpolation benchmark study.

The description of RBF interpolation is given in Section 4.3. The RBF kernel functions assessed in this study are shown in Table 7.3, and the models' *loss* J is computed from (4.14). The benchmark for $\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}(\boldsymbol{\omega}_{\mathrm{N}_i})$ with output dimensions $\mathrm{N}_x = \{\dot{Q},\ \Gamma_f,\ \mathbf{P}\}$ and input features $\mathrm{N}_i = [2,\ 5]$, trained with validation set sizes from Table 7.3 are shown in Figures 7.11 to 7.16 for lumped $\mathbf{y}_{\mathrm{N}_i}^{\dot{Q}}$, 7.21 to 7.28 for spatial $\mathbf{y}_{\mathrm{N}_i}^{\Gamma_f}$ and 7.33 to 7.40 for modal $\mathbf{y}_{\mathrm{N}_i}^{\mathbf{P}}$. The kernel convergence study is evaluated from models trained with 65% of the available high fidelity data –or 35% validation split as shown in Table 7.3–, while the data sensitivity study uses the Linear kernel. These figures show a *violin plot shade*, representing the range of values of the loss from 4 independent models trained with the same architecture but randomly initialised weight parameters. The varying thickness of this shade represents the density of values within the region, and the mean value connects the plot line. The relative error $\mathrm{Er}(\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x})$ of the best Linear kernel predictions against high fidelity data is shown in ascending order of $\overline{\dot{Q}}$ values in Figures 7.17 to 7.20 for lumped $\mathbf{y}_{\mathrm{N}_i}^{\dot{Q}}$, 7.29 to 7.32 for spatial $\mathbf{y}_{\mathrm{N}_i}^{\Gamma_f}$ and 7.41 to 7.44 for modal $\mathbf{y}_{\mathrm{N}_i}^{\mathbf{P}}$.

- **Convergence.** Focusing on the loss benchmark figures, most kernels when training on $\boldsymbol{\omega}_2$ and $\boldsymbol{\omega}_4$, reach and surpass the $\mathrm{J} = 10^{-6}$, which is the target loss imposed in the NNs, but the minimum J depends highly on the kernel it computes with, where the Linear kernel reaches the lowest values by several orders of magnitude –Linear achieved $\mathrm{J}(\mathbf{y}_{2,4}^{\mathrm{N}_x}) = 10^{-30}$ when Gaussian and Multiquadric reached $\mathrm{J}(\mathbf{y}_{2,4}^{\mathrm{N}_x}) = 10^{-28}$ as the lowest J, and Linear reached $\mathrm{J}(\mathbf{y}_{2,4}^{\mathrm{N}_x}) = 10^{-25}$ when Gaussian and Multiquadric reach $\mathrm{J}(\mathbf{y}_{2,4}^{\mathrm{N}_x}) = [10^{-24},\ 10^{-1}]$ as the largest gap between J–. However, when training on $\boldsymbol{\omega}_3$ or $\boldsymbol{\omega}_5$, adding the Re to the features, J increases drastically for all kernels –Gaussian and Multiquadric kernels sharing similar loss at $\mathrm{J}(\mathbf{y}_{3,5}^{\dot{Q}}) = [10^3,\ 10^4]$, $\mathrm{J}(\mathbf{y}_5^{\Gamma_f}) = [10^0,\ 10^3]$, $\mathrm{J}(\mathbf{y}_{3,5}^{\mathbf{P}}) = [10^0,\ 10^4]$– except for Linear with $\mathrm{J}(\mathbf{y}_{3,5}^{\mathrm{N}_x}) = 10^{-25}$.

- **Consistency.** From the loss benchmark figures, for a training set of 65% and over, most models reach a similar J regardless of the random initialisation, dimension $\mathbf{x}^{\mathrm{N}_x}$. This behaviour doesn't change when varying the validation split. However, specifically for feature count $\boldsymbol{\omega}_3$ some spatial and modal outputs have visible variation in the Gaussian and Multiquadric kernels with $\Delta\mathrm{J}(\mathbf{y}_3^{\Gamma_f}) = [10^0,\ 10^2]$, and $\Delta\mathrm{J}(\mathbf{y}_3^{\mathbf{P}}) = [10^0,\ 10^3]$, which are non-negligible inconsistency values.

- **Over-fitting.** When observing $\mathrm{Er}(\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x})$ for the Linear kernel, there are two types of behaviour in higher $\boldsymbol{\omega}_{\mathrm{N}_i}$ regardless of the output size $\mathbf{x}^{\mathrm{N}_x}$. There is a cluster of high density in $\boldsymbol{\omega}_4$ and two clusters of high density in $\boldsymbol{\omega}_5$, both with descending order of errors with increasing $\overline{\dot{Q}}$ values. The error density areas are in the range of $\mathrm{Er}(\mathbf{y}_4^{\mathrm{N}_x}) = [10^{-4},\ 10^1]$ for $Re = 500$, and $\mathrm{Er}(\mathbf{y}_4^{\mathrm{N}_x}) = [10^{-5},\ 10^{-1}]$ for $Re = 8000$. The high errors for lower Re indicate over-fitting, especially in the lower $\overline{\dot{Q}}$ –with negative $\overline{\dot{Q}}$ values–, while the model predicts higher Re with more accuracy. This is observed to be more extreme for $\boldsymbol{\omega}_5$ features, when Re is used for training, where the high density areas are found within $\mathrm{Er}(\mathbf{y}_5^{\dot{Q}}) = [10^0,\ 10^2]$, $\mathrm{Er}(\mathbf{y}_5^{\Gamma_f}) = [10^0,\ 10^3]$ and $\mathrm{Er}(\mathbf{y}_5^{\mathbf{P}}) = [10^1,\ 10^2]$ on $Re = 500$. The outliers in $\boldsymbol{\omega}_5$ have values over $\mathrm{Er}(\mathbf{y}_5^{\dot{Q}}) = 10^2$ and $\mathrm{Er}(\mathbf{y}_{\mathrm{N}_i}^{\Gamma_f,\mathbf{P}}) = 10^3$.

- **Data cost and sensitivity.** From the loss benchmark figures showing the validation split, it seems that increasing the training set does not have a significant impact on the loss of the Linear kernel, where $\mathrm{J}(\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}) = [10^{-24},\ 10^{-32}]$. High over-fitting is observed even with a training set of 65%, reducing the training set size potentially increases over-fitting errors.

- **Computational cost.** When training on $\boldsymbol{\omega}_2$, computational cost does not seem to be heavily affected by validation %, output size $\mathbf{x}^{\mathrm{N}_x}$ or kernel function, since all models had a CPU time $t(\mathbf{y}_2^{\mathrm{N}_x}) \lll 1$ s. This was observed for $\boldsymbol{\omega}_3$ at lower output dimensions $\mathbf{y}_3^{\dot{Q}}$ and $\mathbf{y}_3^{\mathbf{P}}$, but not for $\mathbf{y}_3^{\Gamma_f}$, where $t(\mathbf{y}_2^{\Gamma_f}) \approx 0.2$ s. For $\boldsymbol{\omega}_4$, $t(\mathbf{y}_4^{\dot{Q}}) \lll 1$ s is maintained, but higher output dimensions increase to $t(\mathbf{y}_4^{\Gamma_f}) \approx 0.5$ s, $t(\mathbf{y}_4^{\mathbf{P}}) \leq 0.1$ s. Validation % influences CPU time at $\boldsymbol{\omega}_2$, where with 65% on training set $t(\mathbf{y}_5^{\dot{Q}}) \approx 11$ s, $t(\mathbf{y}_5^{\Gamma_f}) \approx 15$ s, $t(\mathbf{y}_5^{\mathbf{P}}) \approx 13$ s; with 50% on training set $t(\mathbf{y}_5^{\dot{Q}}) \approx 10$ s, $t(\mathbf{y}_5^{\Gamma_f}) \approx 7$ s, $t(\mathbf{y}_5^{\mathbf{P}}) \approx 6$ s; and finally with 35% on training set $t(\mathbf{y}_5^{\dot{Q}}) \approx 3$ s, $t(\mathbf{y}_5^{\Gamma_f}) \approx 3$ s, $t(\mathbf{y}_5^{\mathbf{P}}) \approx 3$ s.

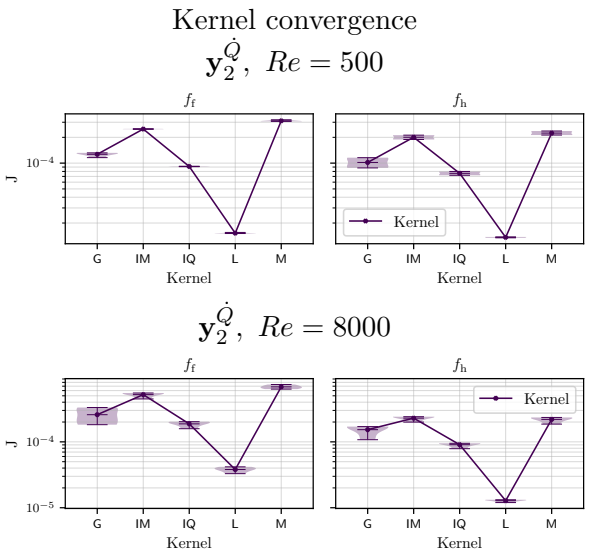


Figure 7.11: L-RBF benchmark - J($\mathbf{y}_2^{\dot{Q}}$) against kernels of dataset $\boldsymbol{\omega}_2$ feature RBF trained with a validation split of 35%. The Lumped output data the RBF is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

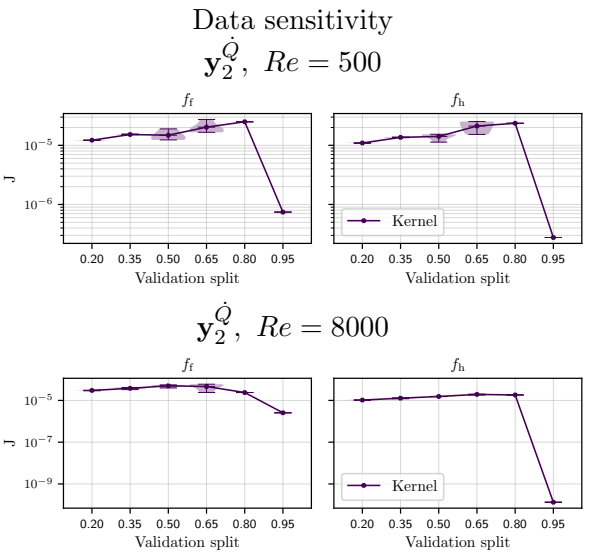


Figure 7.12: L-RBF benchmark - J($\mathbf{y}_2^{\dot{Q}}$) against validation Split of dataset $\boldsymbol{\omega}_2$ feature RBF trained with a Linear kernel. The Lumped output data the RBF is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

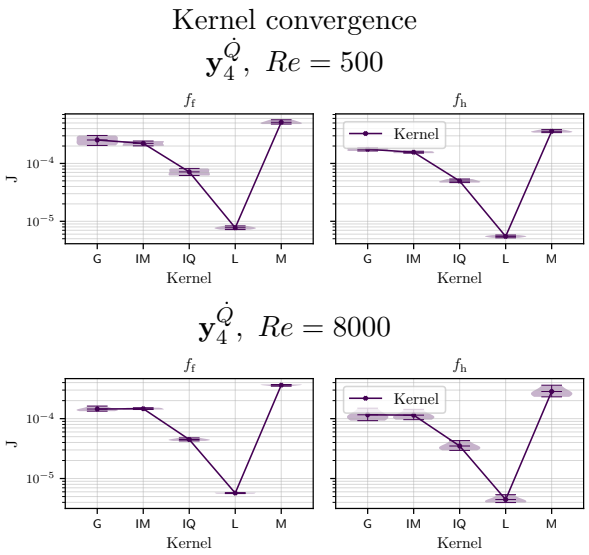


Figure 7.13: L-RBF benchmark - J($\mathbf{y}_4^{\dot{Q}}$) against kernels of dataset $\boldsymbol{\omega}_4$ feature RBF trained with a validation split of 35%. The Lumped output data the RBF is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

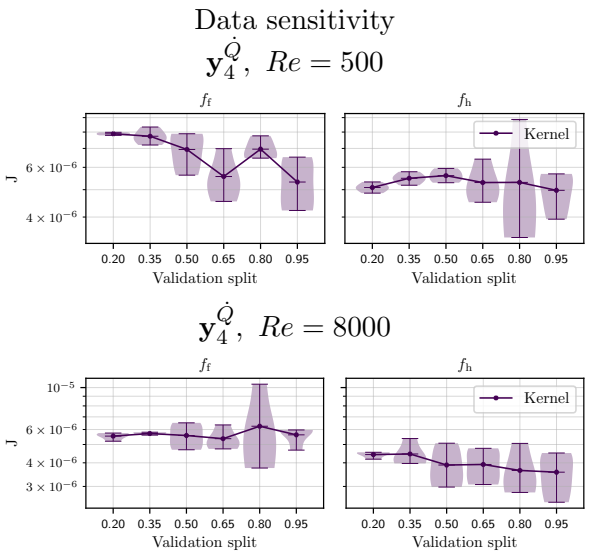


Figure 7.14: L-RBF benchmark - J($\mathbf{y}_4^{\dot{Q}}$) against validation Split of dataset $\boldsymbol{\omega}_4$ feature RBF trained with a Linear kernel. The Lumped output data the RBF is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

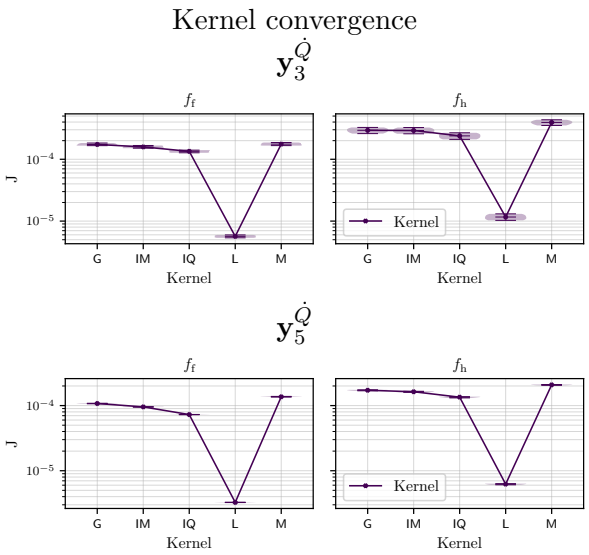


Figure 7.15: L-RBF benchmark - $\mathrm{J}(\mathbf{y}_{3,5}^{\dot{Q}})$ against kernels of datasets –top to bottom– $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Lumped output data the RBF is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

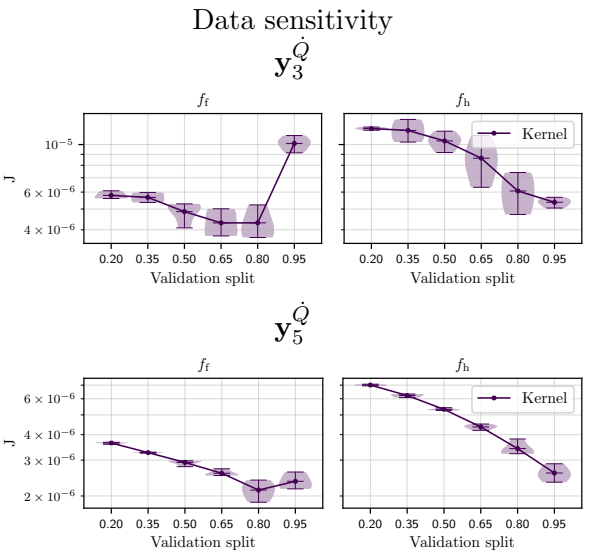


Figure 7.16: L-RBF benchmark - $\mathrm{J}(\mathbf{y}_{3,5}^{\dot{Q}})$ against validation Split of dataset –top to bottom– $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a a Linear kernel. The Lumped output data the RBF is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

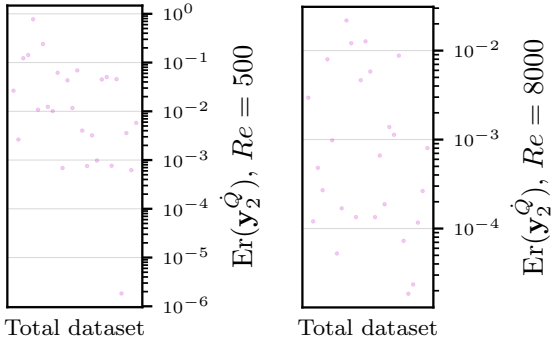


Figure 7.17: L-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_2^{\dot{Q}})$ of independent cases prediction from $\boldsymbol{\omega}_2$ feature RBF trained with a validation split of 35% vs high-fidelity data.

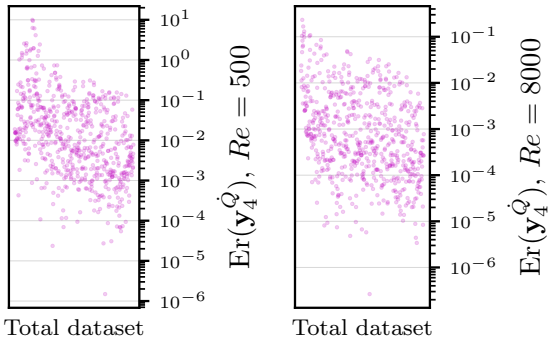


Figure 7.18: L-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_4^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature RBF trained with a validation split of 35% vs high-fidelity data.

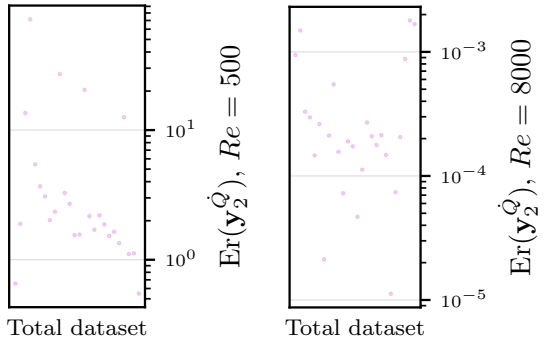


Figure 7.19: L-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_3^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_3$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

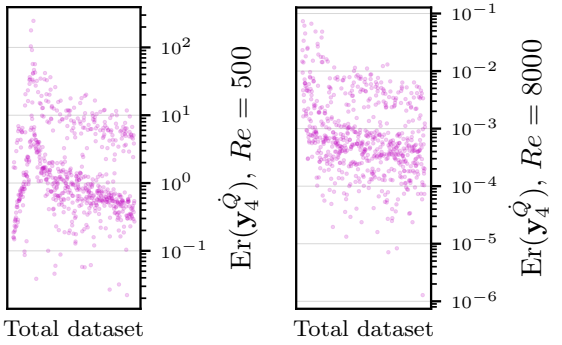


Figure 7.20: L-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_5^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_5$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

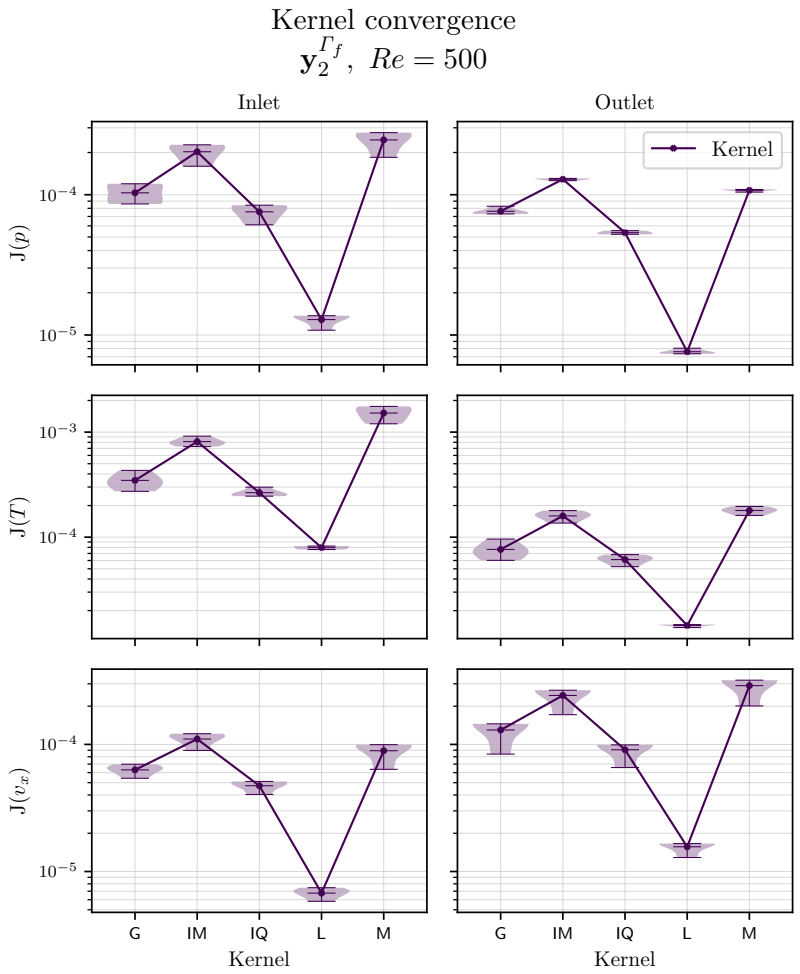


Figure 7.21: S-RBF benchmark - J($\mathbf{y}_2^{\Gamma_f}$) against kernels of dataset $\boldsymbol{\omega}_2$ feature RBF trained with a validation split of 35%. The Spatial output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

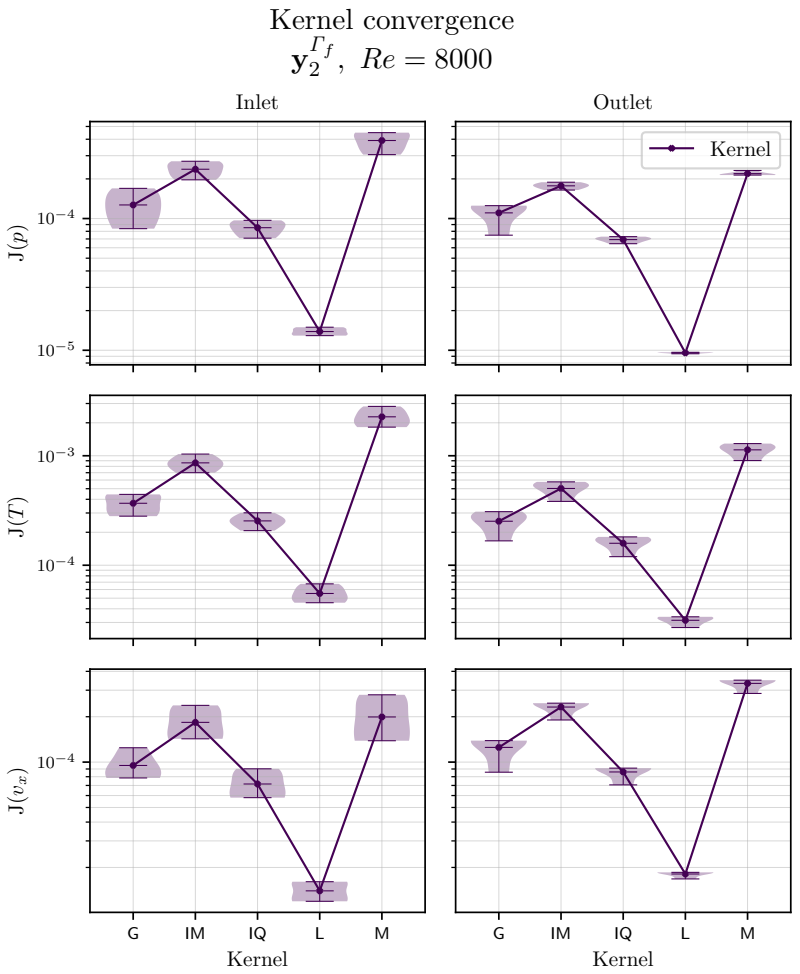


Figure 7.22: S-RBF benchmark - J($\mathbf{y}_2^{\Gamma_f}$) against kernels of dataset $\boldsymbol{\omega}_2$ feature RBF trained with a validation split of 35%. The Spatial output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

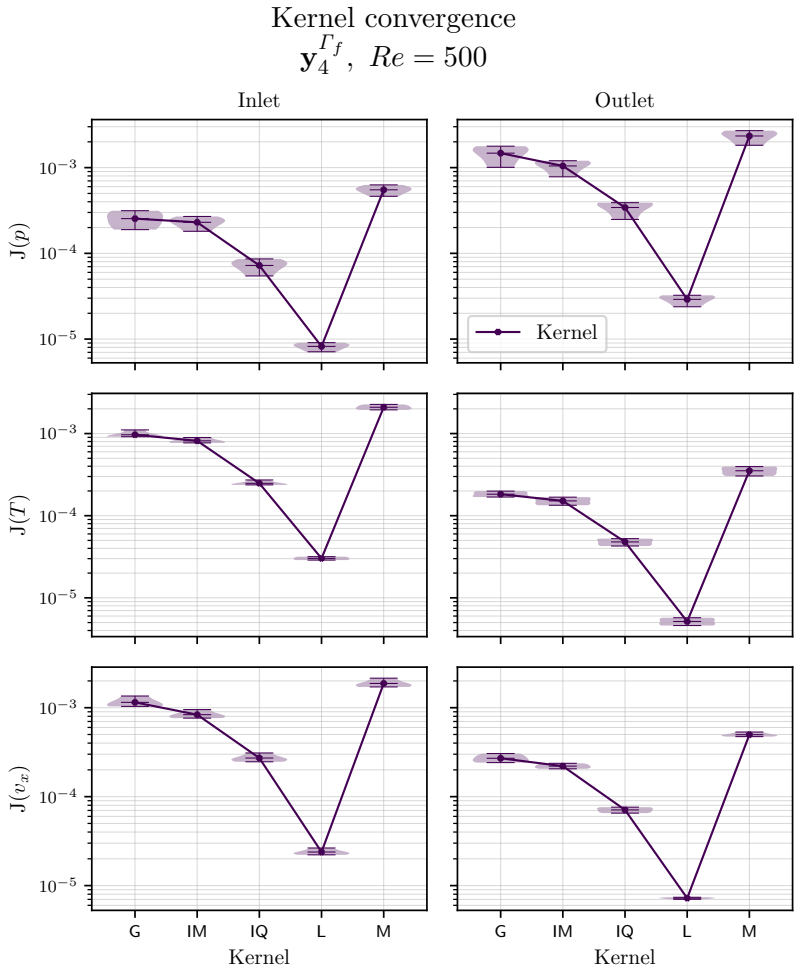


Figure 7.23: S-RBF benchmark - J($\mathbf{y}_4^{\Gamma_f}$) against kernels of dataset $\boldsymbol{\omega}_4$ feature RBF trained with a validation split of 35%. The Spatial output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

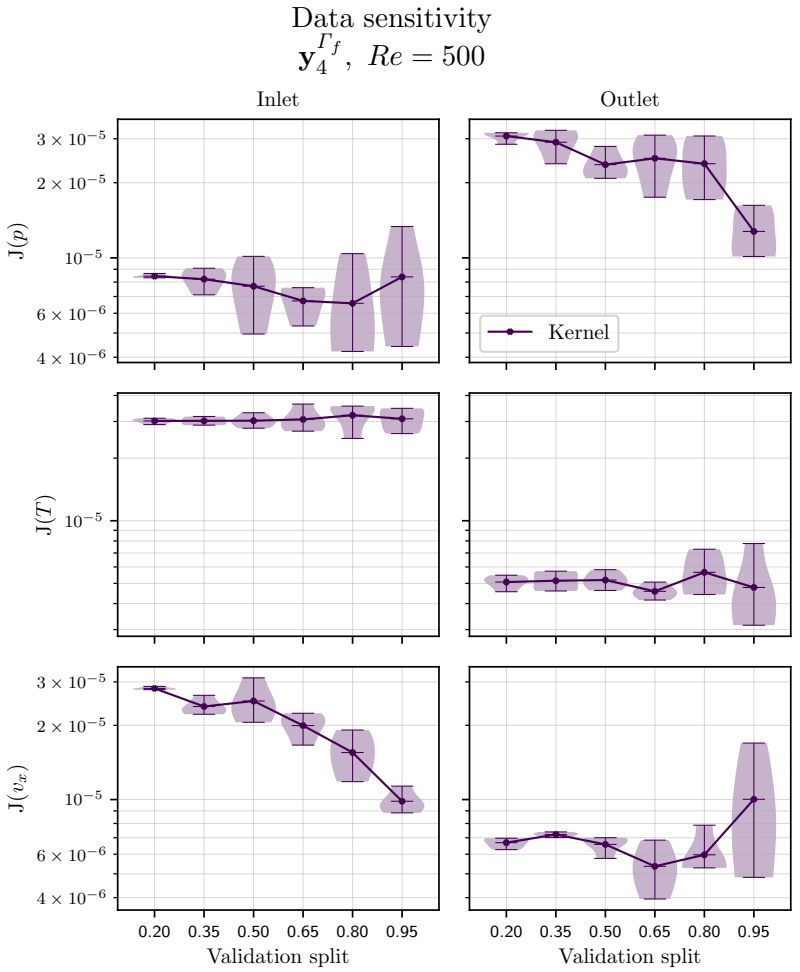


Figure 7.24: S-RBF benchmark - J($\mathbf{y}_4^{\Gamma_f}$) against validation Split of dataset $\boldsymbol{\omega}_4$ feature RBF trained with a Linear kernel. The Spatial output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

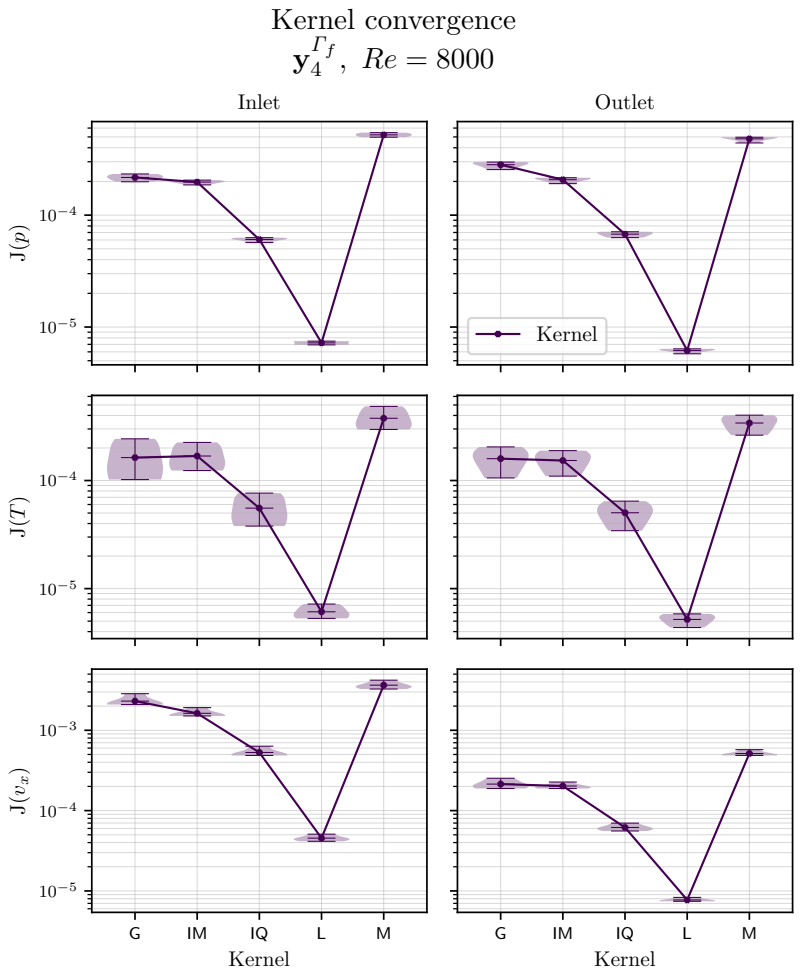


Figure 7.25: S-RBF benchmark - J($\mathbf{y}_4^{\Gamma_f}$) against kernels of dataset $\boldsymbol{\omega}_4$ feature RBF trained with a validation split of 35%. The Spatial output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

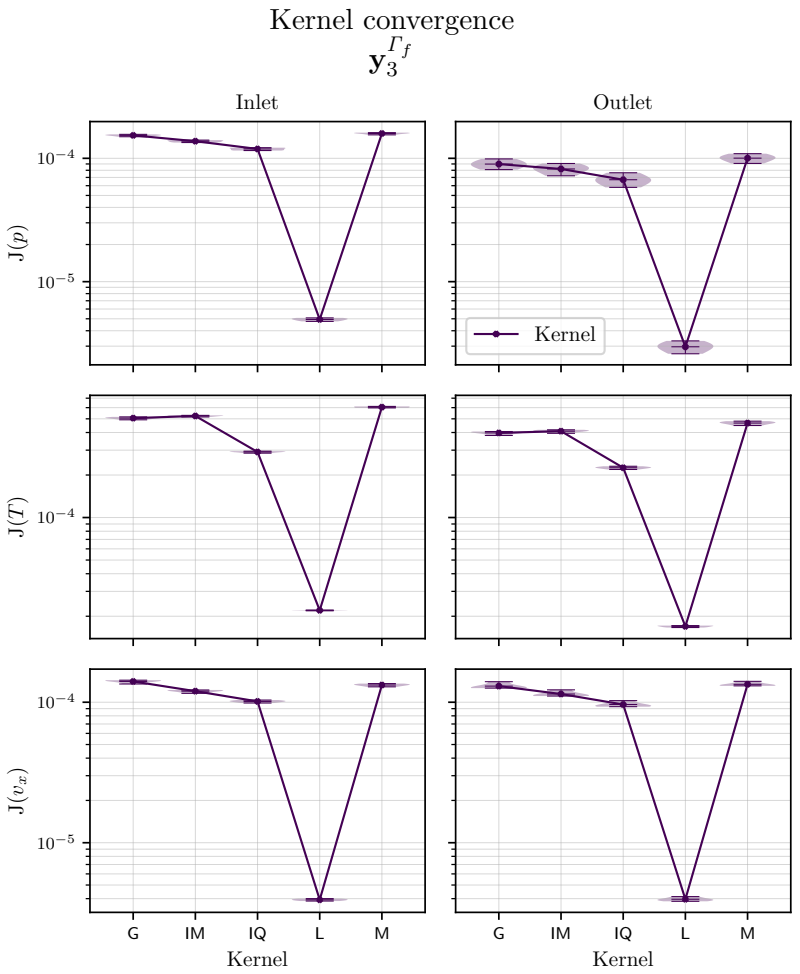


Figure 7.26: S-RBF benchmark - J($\mathbf{y}_3^{\Gamma_f}$) against kernels of dataset $\boldsymbol{\omega}_3$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Spatial output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

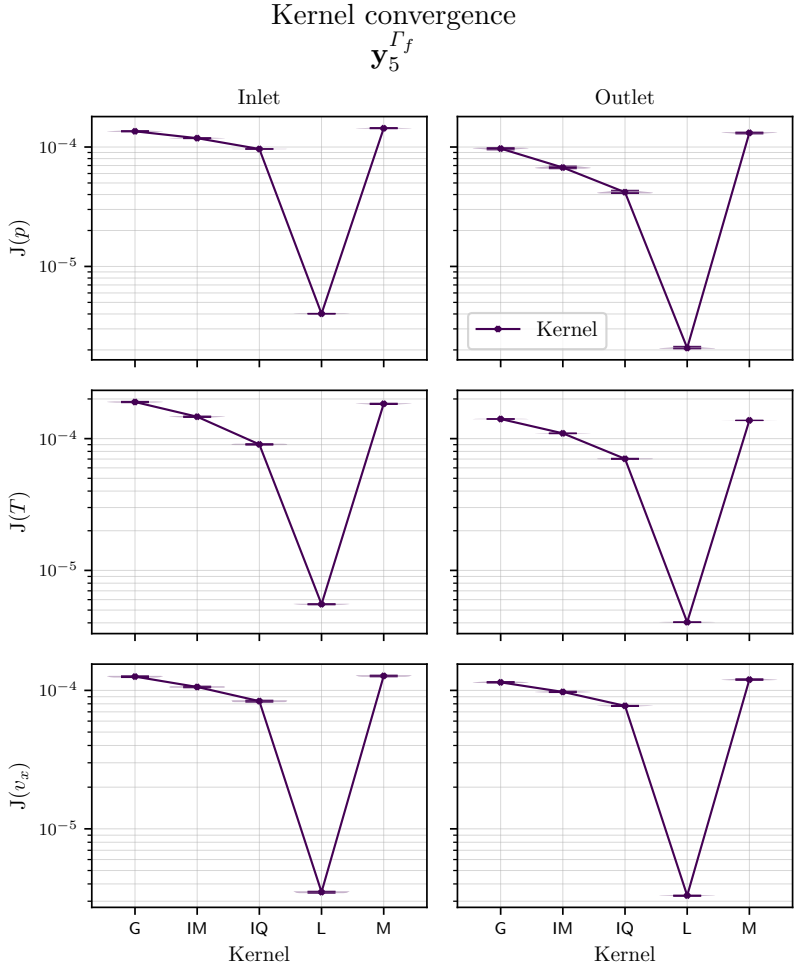


Figure 7.27: S-RBF benchmark - J($\mathbf{y}_5^{\Gamma_f}$) against kernels of dataset $\boldsymbol{\omega}_5$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Spatial output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

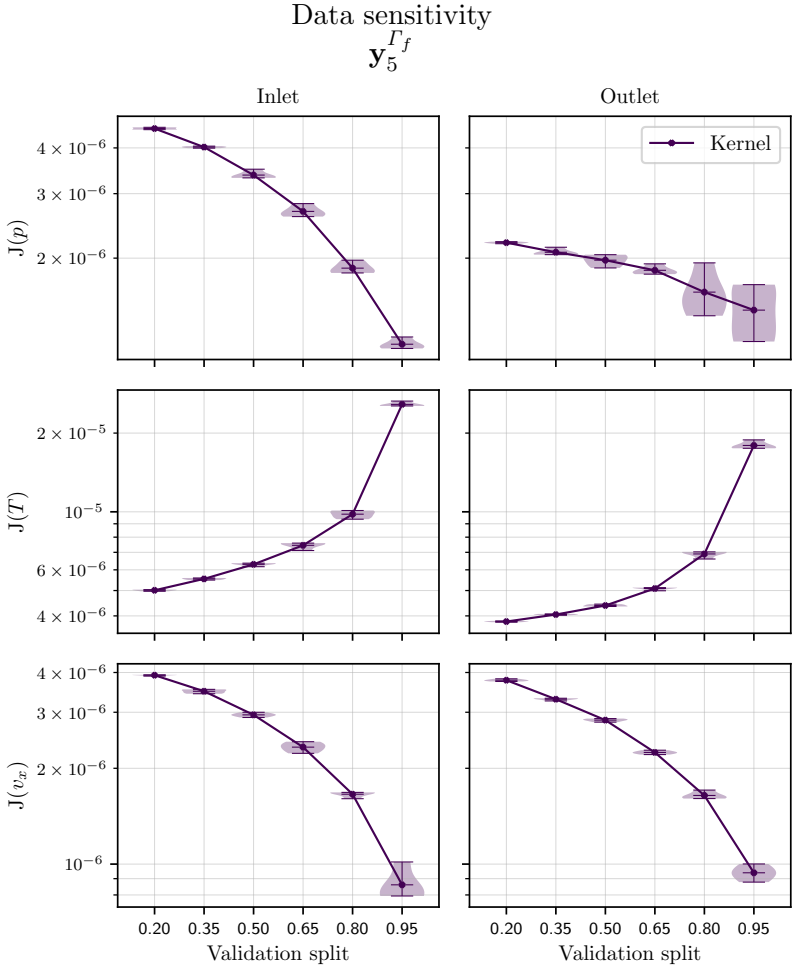


Figure 7.28: S-RBF benchmark - J($\mathbf{y}_5^{\Gamma_f}$) against validation Split of dataset $\boldsymbol{\omega}_5$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a Linear kernel. The Spatial output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

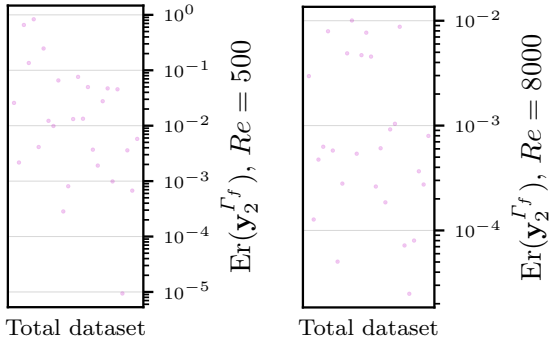


Figure 7.29: S-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_2^{\Gamma_f})$ of independent cases prediction from $\boldsymbol{\omega}_2$ feature RBF trained with a validation split of 35% vs high-fidelity data.

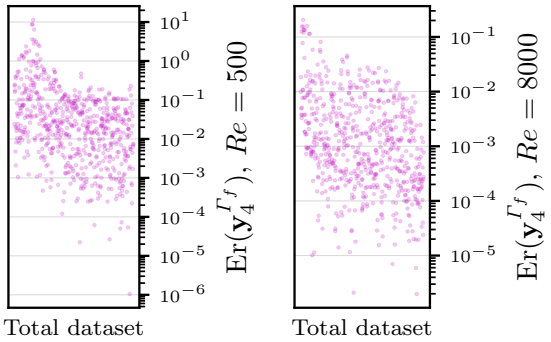


Figure 7.30: S-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_4^{\Gamma_f})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature RBF trained with a validation split of 35% vs high-fidelity data.

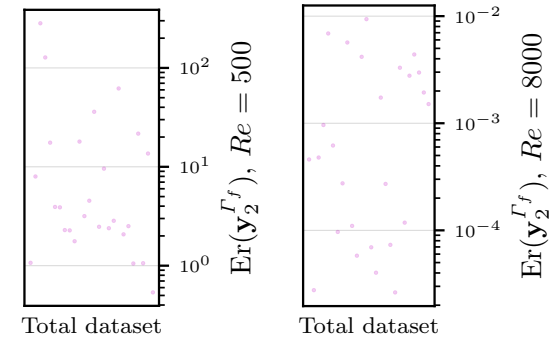


Figure 7.31: S-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_3^{\Gamma_f})$ of independent cases prediction from the $\boldsymbol{\omega}_3$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

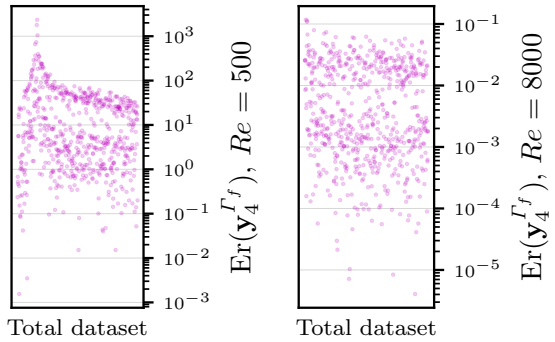


Figure 7.32: S-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_5^{\Gamma_f})$ of independent cases prediction from the $\boldsymbol{\omega}_5$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

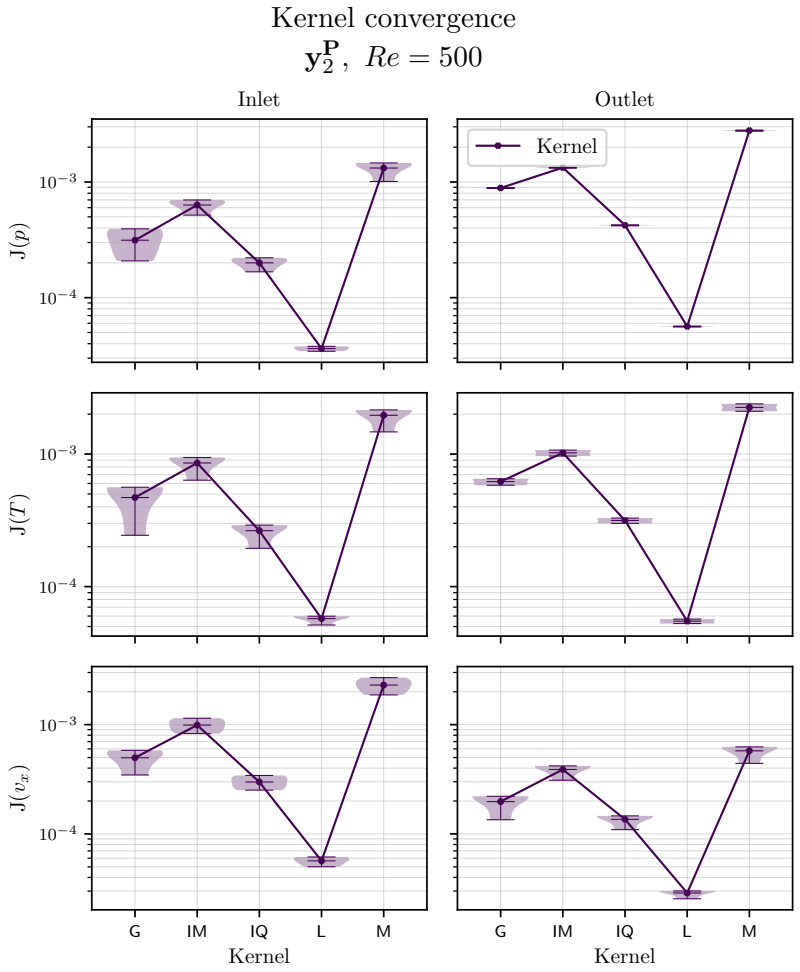


Figure 7.33: M-RBF benchmark - J($\mathbf{y}_2^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_2$ feature RBF trained with a validation split of 35%. The Modal output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

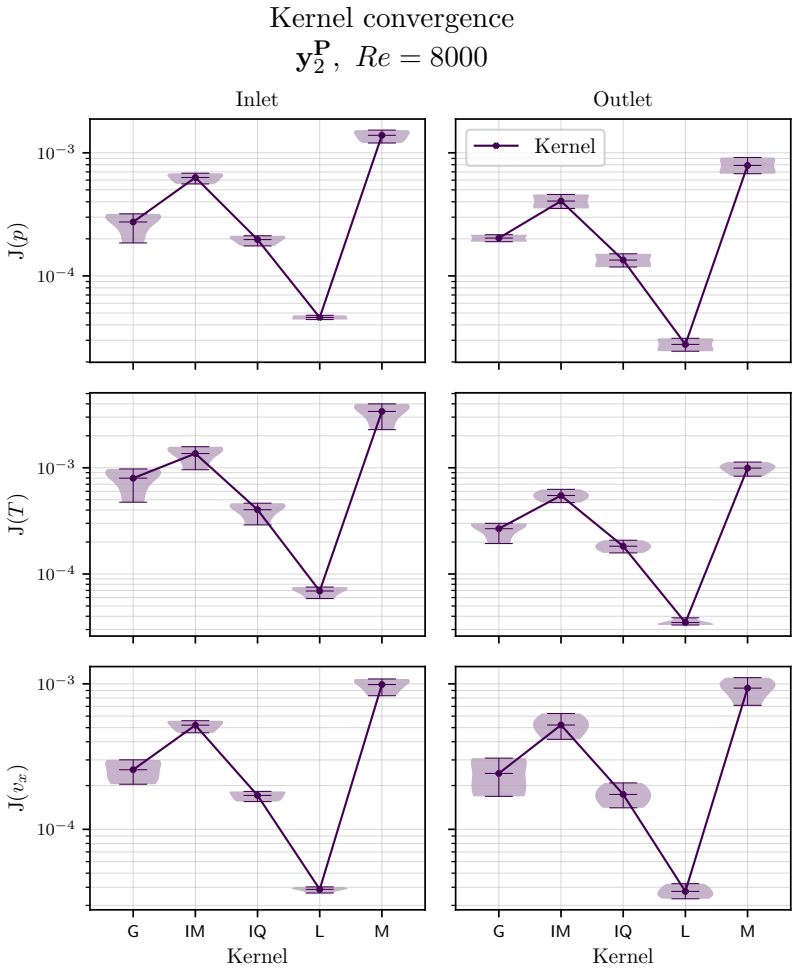


Figure 7.34: M-RBF benchmark - J($\mathbf{y}_2^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_2$ feature RBF trained with a validation split of 35%. The Modal output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

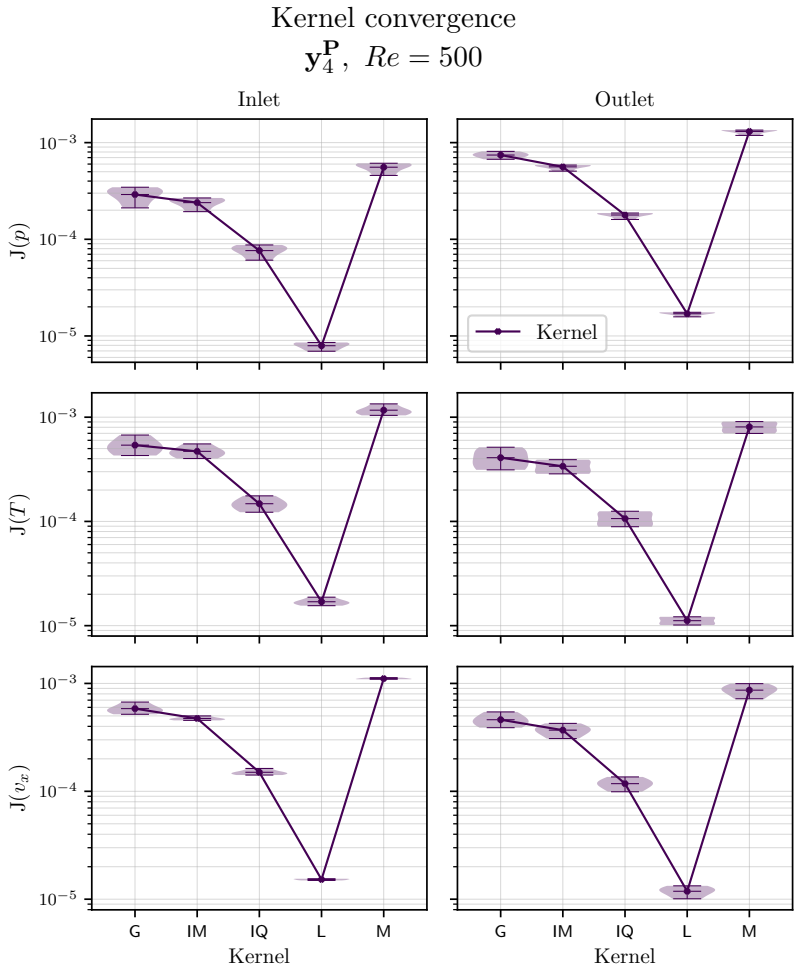


Figure 7.35: M-RBF benchmark - J($\mathbf{y}_4^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_4$ feature RBF trained with a validation split of 35%. The Modal output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

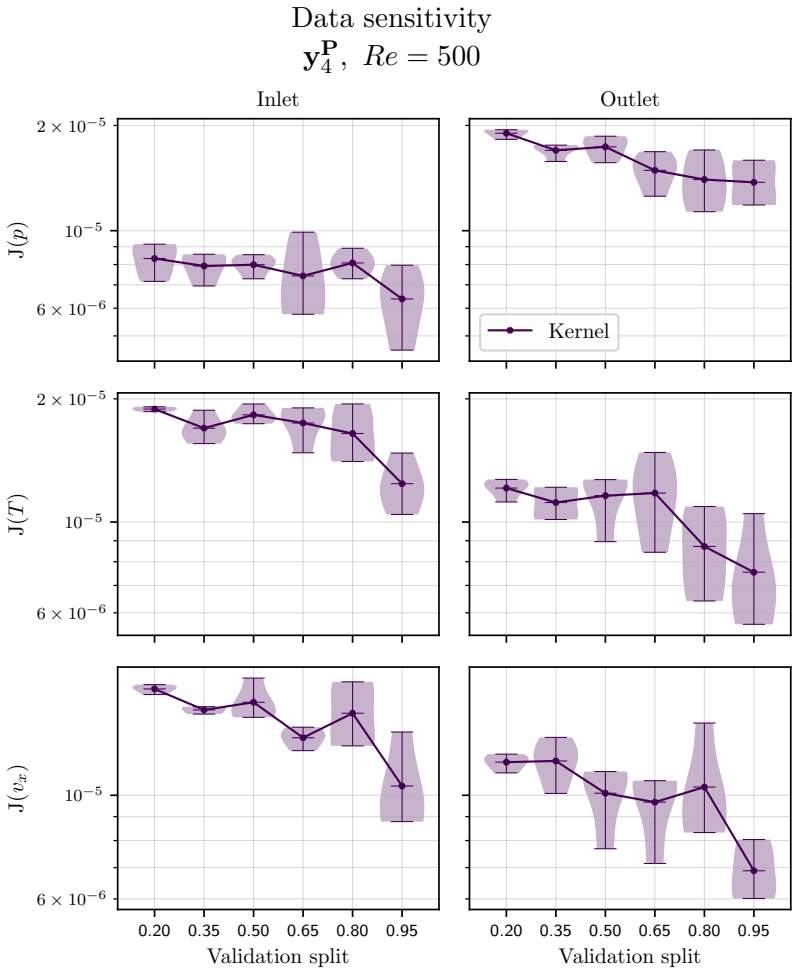


Figure 7.36: M-RBF benchmark - J($\mathbf{y}_4^{\mathbf{P}}$) against validation Split of dataset $\boldsymbol{\omega}_4$ feature RBF trained with a Linear kernel. The Modal output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

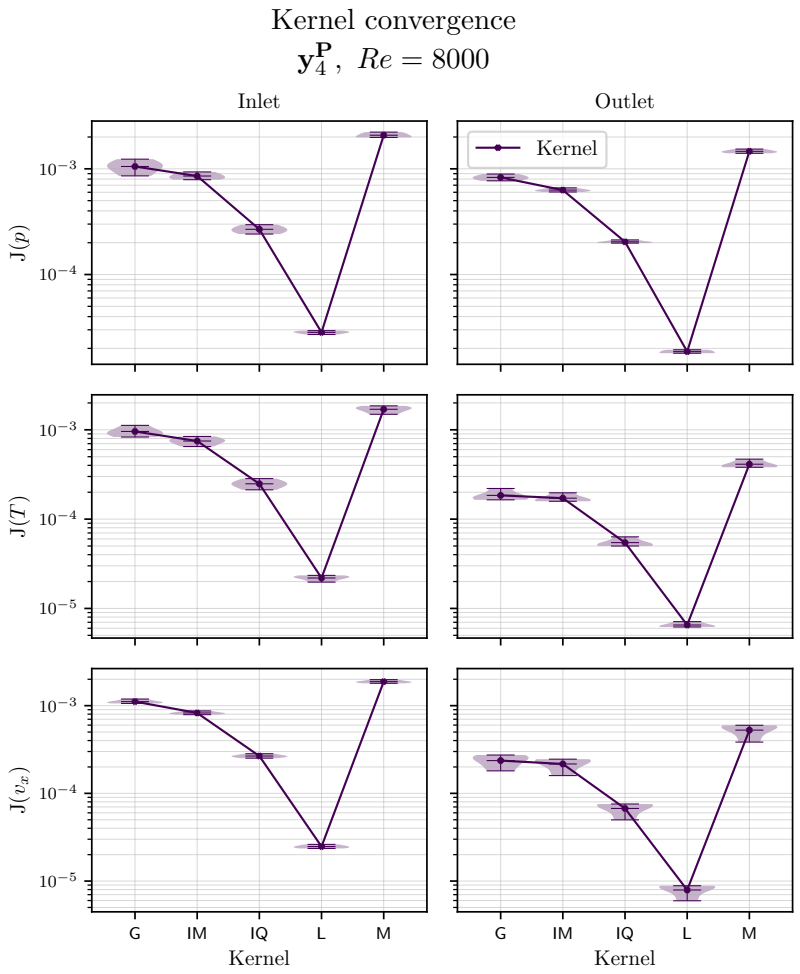


Figure 7.37: M-RBF benchmark - J($\mathbf{y}_4^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_4$ feature RBF trained with a validation split of 35%. The Modal output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

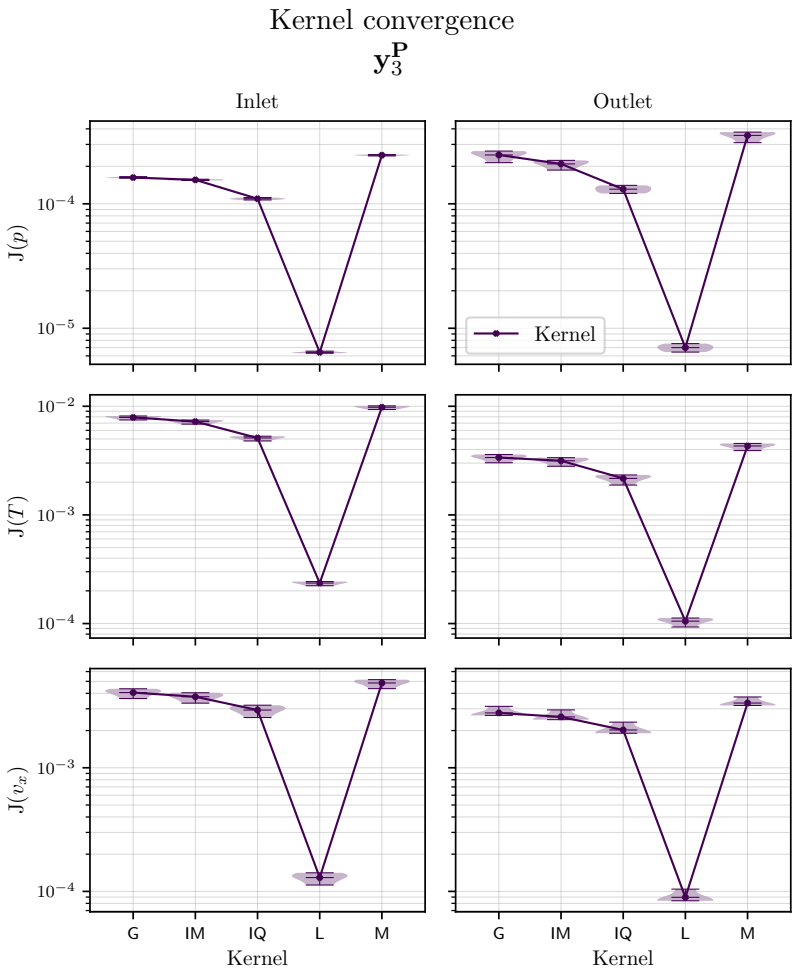


Figure 7.38: M-RBF benchmark - J($\mathbf{y}_3^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_3$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Modal output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

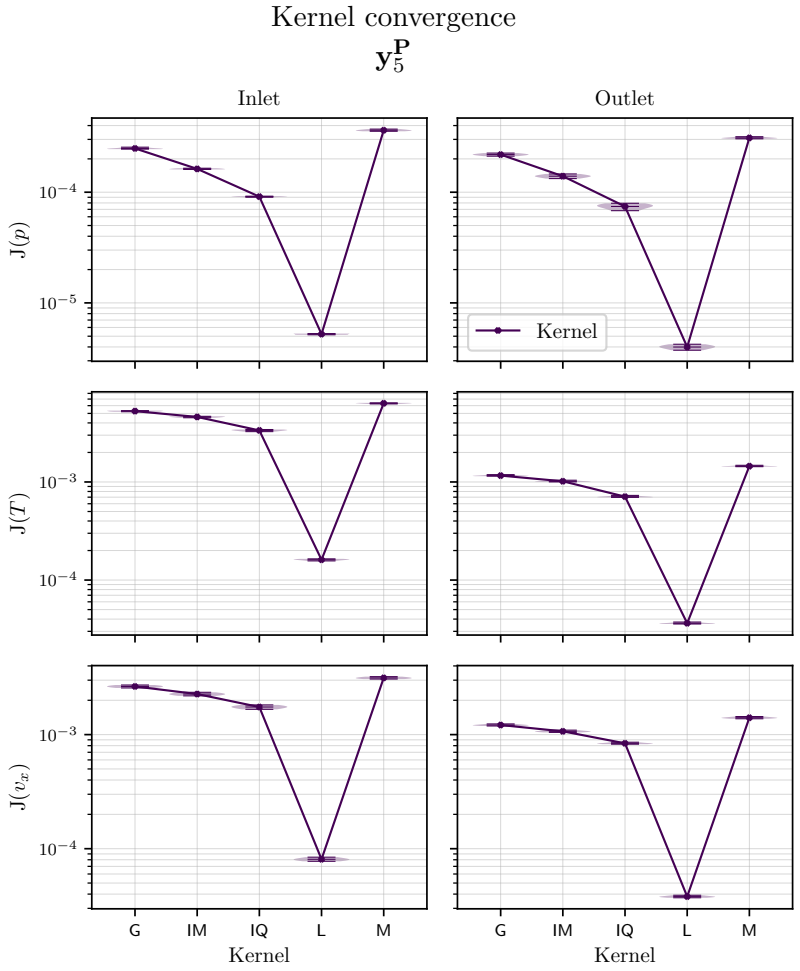


Figure 7.39: M-RBF benchmark - J($\mathbf{y}_5^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_5$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Modal output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

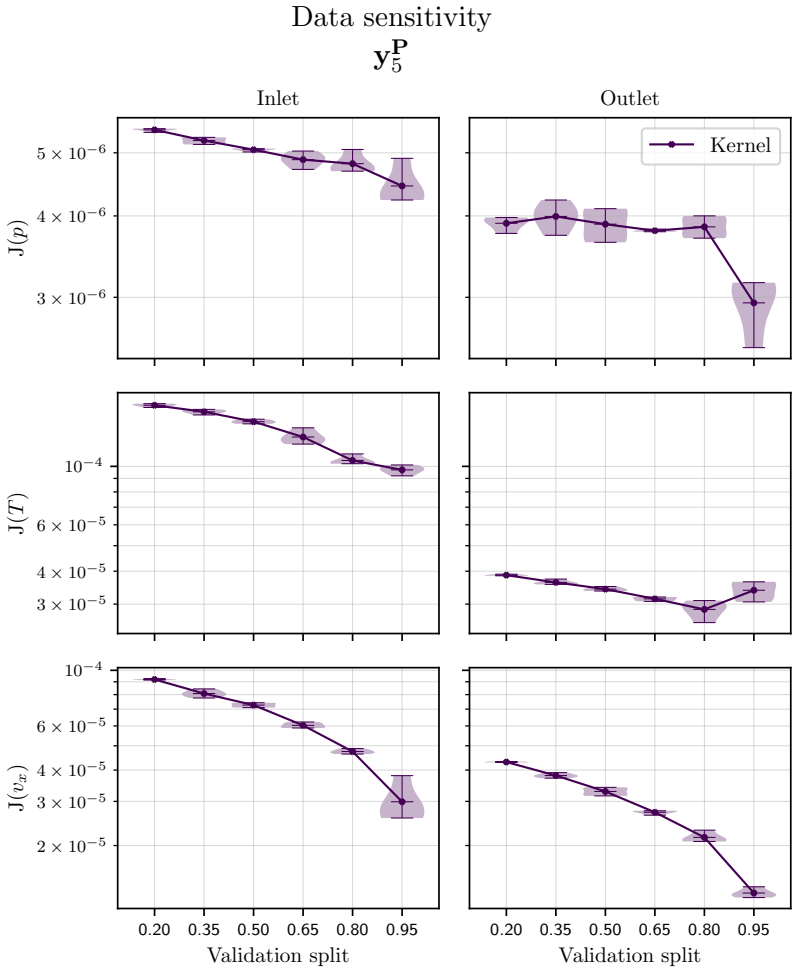


Figure 7.40: M-RBF benchmark - J($\mathbf{y}_5^{\mathbf{P}}$) against validation split of dataset $\boldsymbol{\omega}_5$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Modal output data the RBF is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

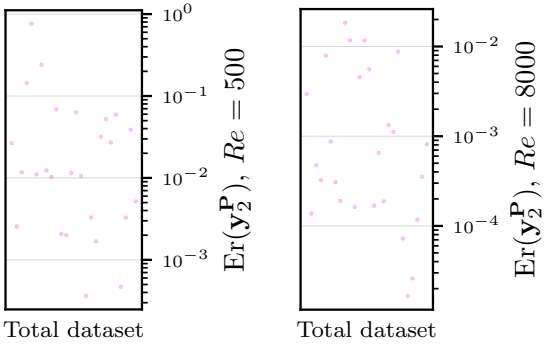


Figure 7.41: M-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_2^{\mathbf{P}})$ of independent cases prediction from the $\boldsymbol{\omega}_2$ feature RBF trained with a validation split of 35% vs high-fidelity data.

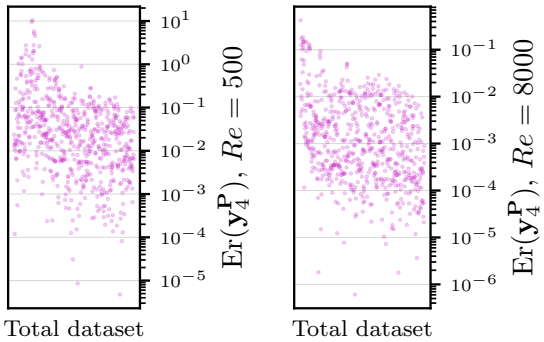


Figure 7.42: M-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_4^{\mathbf{P}})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature RBF trained with a validation split of 35% vs high-fidelity data.

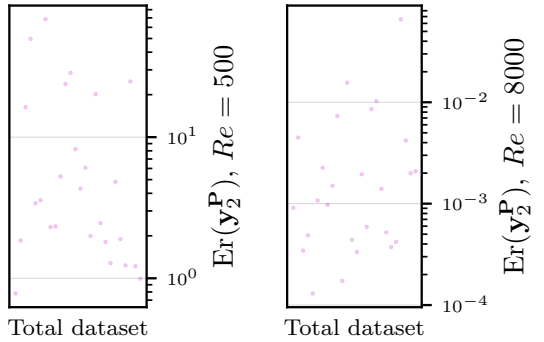


Figure 7.43: M-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_3^{\mathbf{P}})$ of independent cases prediction from the $\boldsymbol{\omega}_3$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

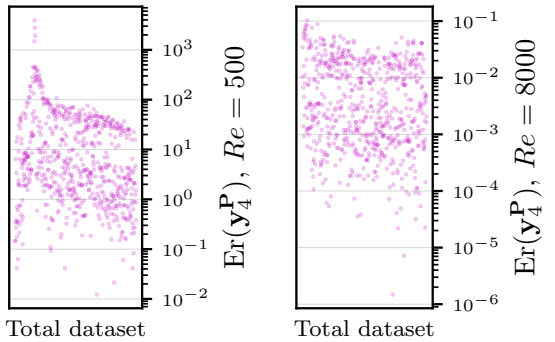


Figure 7.44: M-RBF benchmark - Relative error $\mathrm{Er}(\mathbf{y}_5^{\mathbf{P}})$ of independent cases prediction from the $\boldsymbol{\omega}_5$ feature RBF trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

RBF Preliminary findings

From the observations in Section 7.3.1, the sensitivity analysis of RBF models with increasing complexities in the problem is discussed alongside findings from said analysis.

- **Convergence.** Heavily sensitive to kernel function and feature set size, high loss values are observed at higher $\boldsymbol{\omega}_{\mathrm{N}_i}$ and the Kernel reaching lowest loss values consistently is Linear. Output size $\mathbf{x}^{\mathrm{N}_x}$ has a mild impact, more visible at higher $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$, where surprisingly the lowest $\mathbf{x}^{\mathrm{N}_x}$ got the highest losses.
- **Consistency.** Heavily sensitive to $\mathbf{x}^{\mathrm{N}_x}$ and kernel at $\boldsymbol{\omega}_3$, specifically.
- **Over-fitting.** Heavily sensitive to flow regime regardless of feature set size $\boldsymbol{\omega}_{\mathrm{N}_i}$ and output size $\mathbf{x}^{\mathrm{N}_x}$. $Re = 500$ has values further apart from each other $-\overline{\dot{Q}} = [-12, 5]-$ compared to $Re = 8000$ $-\overline{\dot{Q}} = [0, 1]-$, highly non-linear behaviour is assumed[1]. Over-fitting is enhanced by increasing $\boldsymbol{\omega}_{\mathrm{N}_i}$ and $\mathbf{x}^{\mathrm{N}_x}$, sensitivity of the model to increasing input-output sizes is non-negligible.
- **Data cost and sensitivity.** Due to the observed behaviour in over-fitting with a training dataset of 65% from the high fidelity dataset, a larger training set might be required in laminar regimes and higher $\boldsymbol{\omega}_{\mathrm{N}_i}$, implying high sensitivity to $\boldsymbol{\omega}_{\mathrm{N}_i}$ size and to non-linear behaviour in the function.
- **Computational cost.** Exponentially sensitive to increase in $\boldsymbol{\omega}_{\mathrm{N}_i}$. At $\boldsymbol{\omega}_4$ or lower, the computational cost is negligible, but at $\boldsymbol{\omega}_5$, the sensitivity to increasing output sizes $\mathbf{x}^{\mathrm{N}_x}$ and training dataset sizes is non-negligible.

Overall, the main advantages of using the RBF algorithm are its extremely small computational footprint with a steep loss convergence. RBF, as a simple linear system of equations, has a general accuracy close to the numerical errors of the machine precision when solved directly, like with the Linear kernel. However, this tool is heavily prone to over-fitting in low Re even if the flow is not a parameter in $\boldsymbol{\omega}_{\mathrm{N}_i}$. This is most likely related to the extreme changes observed at low Re $-\overline{\dot{Q}} = [-12, 8]-$, described in Chapter 6.3, which leads to increased non-linear behaviour of the function. The algorithm does not seem suitable to approach the problem with input features $\boldsymbol{\omega}_5$ with the current architectures, but has enough accuracy to approach it with $\boldsymbol{\omega}_4$, where Re are treated independently. The performance of all output dimensions are too similar to justify the use of the low-rank output $\mathbf{x}^{\mathbf{P}}$. This assessment satisfies Objectives VII.IV and VII.V.

7.3.2 Neural Networks

The general design of an NN is described in Section 4.4. NN architecture layer and neuron count assessed in this study are shown in Table 7.4. The performance index of the feedforward NN trained through backpropagation method is the models' *loss* J computed from (4.14), with

[1] The values are normalised to avoid the influence of magnitude differences in the performance of the models, but the non-linear behaviour remains part of the dataset.

Parameter	
Layers	{1, 2, 3}
Neurons	{8, 16, 32, 64, 128, 256}
Validation/training split	{20%, 35%, 50%, 65%, 80% 95%}

Table 7.4: NN benchmark - Numerical parameters for the neural network architecture benchmark study.

a target value of $\mathrm{J} = 10^{-6}$ imposed in the models to reduce computational cost. The NN's architecture benchmark for $\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}(\boldsymbol{\omega}_{\mathrm{N}_i})$ with output dimensions $\mathrm{N}_x = \{\dot{Q},\ \Gamma_f,\ \mathbf{P}\}$ and input features $\mathrm{N}_i = [2,\ 5]$, trained with validation set sizes from Table 7.4 are shown in Figures 7.45 to 7.50 for lumped $\mathbf{y}_{\mathrm{N}_i}^{\dot{Q}}$, 7.55 to 7.62 for spatial $\mathbf{y}_{\mathrm{N}_i}^{\Gamma_f}$ and 7.67 to 7.74 for modal $\mathbf{y}_{\mathrm{N}_i}^{\mathbf{P}}$. The architecture convergence study is evaluated from models trained with 65% of the available high fidelity data –or 35% validation split as shown in Table 7.4–, while the data sensitivity study uses an architecture containing 3 layers and 128 neurons per layer. These figures show a *violin plot shade*, representing the range of values of the loss from 4 independent models trained with the same architecture but randomly initialised weight parameters. Some figures from the data sensitivity study show partial results of other layer counts with a single validation percentage. These are remnant of results from the convergence study with 128 neurons, and are shown due to Hammerhead modular, batch plotting. The varying thickness of this shade represents the density of values within the region, and the mean value connects the plot line. The relative error $\mathrm{Er}\left(\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}\right)$ of the best architecture predictions against high fidelity data is shown in ascending order of $\overline{\dot{Q}}$ values in Figures 7.51 to 7.54 for lumped $\mathbf{y}_{\mathrm{N}_i}^{\dot{Q}}$, 7.63 to 7.66 for spatial $\mathbf{y}_{\mathrm{N}_i}^{\Gamma_f}$ and 7.75 to 7.78 for modal $\mathbf{y}_{\mathrm{N}_i}^{\mathbf{P}}$.

- **Convergence.** From the loss benchmark figures, we observe a minimum of 1 layer with 16 neurons is required to reach the target loss $\mathrm{J} = 10^{-6}$ with $\boldsymbol{\omega}_2$, regardless of the output size $\mathbf{x}^{\mathrm{N}_x}$. For $\boldsymbol{\omega}_5$, when training on Re as an added feature, neither model reached the desired $\mathrm{J} = 1e^{-6}$, and the consistent architecture that did reach the target loss for $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_4$ across all $\mathbf{x}^{\mathrm{N}_x}$ and variables has 3 layers and 128 neurons. The general trend with $\boldsymbol{\omega}_3$ to $\boldsymbol{\omega}_5$ shows that increasing neuron count reduces loss value, however most models show an increase in loss when the neuron count surpasses 200. This trend is similar when increasing layer count, where the loss values of 2 and 3 layers are close for most models. The increase in loss is most drastic for $\mathbf{y}_4^{\dot{Q}}$, and some variables of $\mathbf{y}_4^{\Gamma_f}$, while smooth and consistent $\mathbf{y}_{\mathrm{N}_i}^{\mathbf{P}}$.
- **Consistency.** Similarly to convergence, models with at least 16 neurons for $\boldsymbol{\omega}_2$ reach the target loss $\mathrm{J} = 10^{-6}$ regardless of the random initialisation, but this consistency reduces with less layers and lower neuron counts. For $\boldsymbol{\omega}_3$ to $\boldsymbol{\omega}_5$, the random initialisation has a significant influence in deeper networks in output sizes $\mathbf{y}_{\mathrm{N}_i}^{\dot{Q}}$ and $\mathbf{y}_{\mathrm{N}_i}^{\Gamma_f}$, while $\mathbf{y}_4^{\mathbf{P}}$ has the highest inconsistencies in 1 and 2 layers, but remains fairly consistent at $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$.
- **Over-fitting.** When observing $\mathrm{Er}(\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x})$ for all output sizes, the error cluster has the

form of an exponentially modified Gaussian distribution, and an extreme shift towards higher error values at low Re and $\overline{\dot{Q}}$. Most of the density area is in the range of $\mathrm{Er}(\mathbf{y}_4^{\mathrm{N}_x}) = [10^{-3},\ 10^{-1}]$ for $Re = 500$, with the shift going up to $\mathrm{Er}(\mathbf{y}_4^{\mathrm{N}_x}) = [10^{-2},\ 10^{0}]$. The range for $Re = 8000$ is $\mathrm{Er}(\mathbf{y}_4^{\mathrm{N}_x}) = [10^{-4},\ 10^{-2}]$, where the data is more sparse. The shift in the error density area for lower Re at $\boldsymbol{\omega}_5$ features grows to values within $\mathrm{Er}(\mathbf{y}_5^{\dot{Q},\Gamma_f}) = [10^{-1},\ 10^{1}]$ and $\mathrm{Er}(\mathbf{y}_5^{\mathbf{P}}) \approx 10^{1}$ on $Re = 500$. The outliers increase to values over $\mathrm{Er}(\mathbf{y}_5^{\Gamma_f}) = 10^{2}$ and $\mathrm{Er}(\mathbf{y}_{\mathrm{N}_i}^{\mathbf{P}}) = 10^{3}$. Note that when using the modal output, $\mathbf{y}_4^{\mathbf{P}}$ has the lowest outlier errors on $Re = 500$, but $\mathbf{y}_5^{\mathbf{P}}$ has the highest outlier error.

- **Data cost and sensitivity.** About the training set size study, with over-fitting to smaller batches for input features $\boldsymbol{\omega}_5$. From the loss benchmark figures showing the validation split, it seems that increasing the training set does not have a significant impact on the loss of the network with 3 layers and 128 neurons, where $\mathrm{J}(\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}) = [10^{-6},\ 10^{-4}]$. High over-fitting is observed even with a training set of 65% of the high fidelity dataset, reducing the training set size increases errors.

- **Computational cost.** The computational cost of an NN scales exponentially when increasing any of the parameters: layer count, neuron count, output size $\mathbf{x}^{\mathrm{N}_x}$, feature count $\boldsymbol{\omega}_{\mathrm{N}_i}$, and training data size. At $\boldsymbol{\omega}_2$, this is less visible since networks with higher neuron counts reach the target loss $\mathrm{J} = 10^{-6}$ in earlier *epoch*, where they stop their computations. CPU time of the lowest $\mathbf{x}^{\mathrm{N}_x}$ on a training set of 65% ranges from $t(\mathbf{y}_2^{\dot{Q}}) \approx 5$ s with 3 layers and 128 neurons, to $t(\mathbf{y}_5^{\dot{Q}}) \approx 2425$ s with 3 layers and 256 neurons. On a training set of 35% the values of the highest $\mathbf{x}^{\mathrm{N}_x}$ range from $t(\mathbf{y}_2^{\Gamma_f}) \approx 1$ s with 2 layers and 64 neurons, to $t(\mathbf{y}_5^{\Gamma_f}) \approx 2100$ s with 3 layers and 256 neurons.

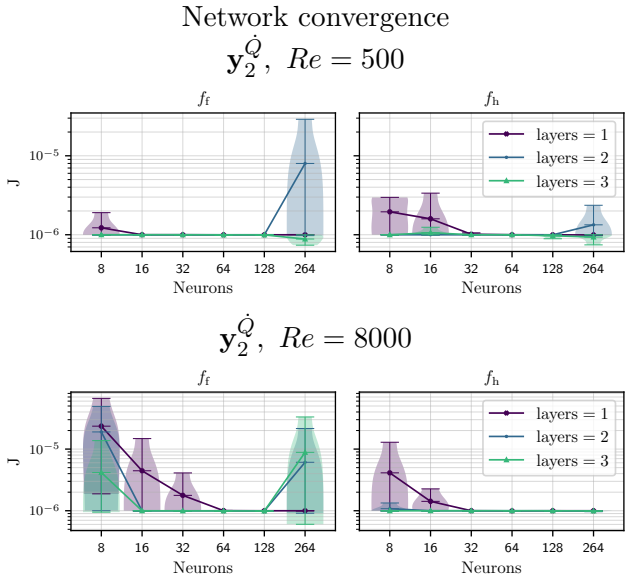


Figure 7.45: L-NN benchmark - J($\mathbf{y}_2^{\dot{Q}}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_2$ feature network trained with a validation split of 35%. The Lumped output data the network is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

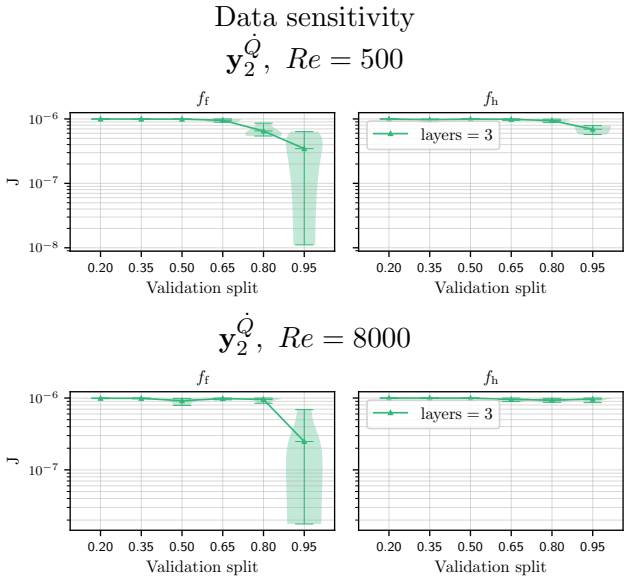


Figure 7.46: L-NN benchmark - J($\mathbf{y}_2^{\dot{Q}}$) against validation split of dataset $\boldsymbol{\omega}_2$ feature network trained with 128 neurons and 3 layers. The Lumped output data the network is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

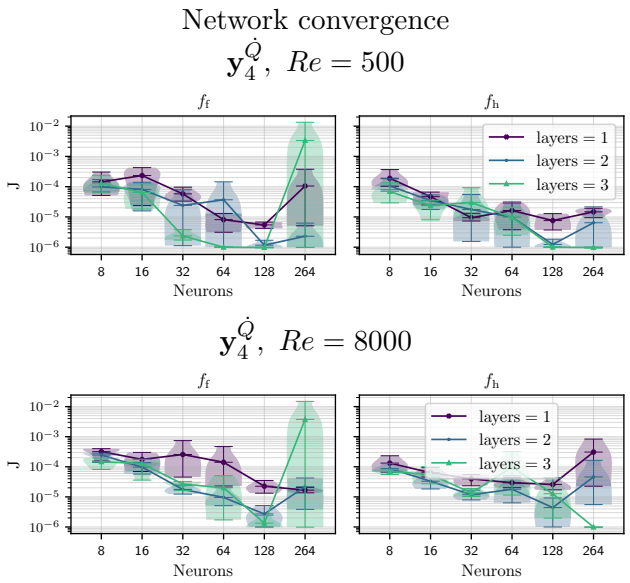


Figure 7.47: L-NN benchmark - J($\mathbf{y}_4^{\dot{Q}}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_4$ feature network trained with a validation split of 35%. The Lumped output data the network is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

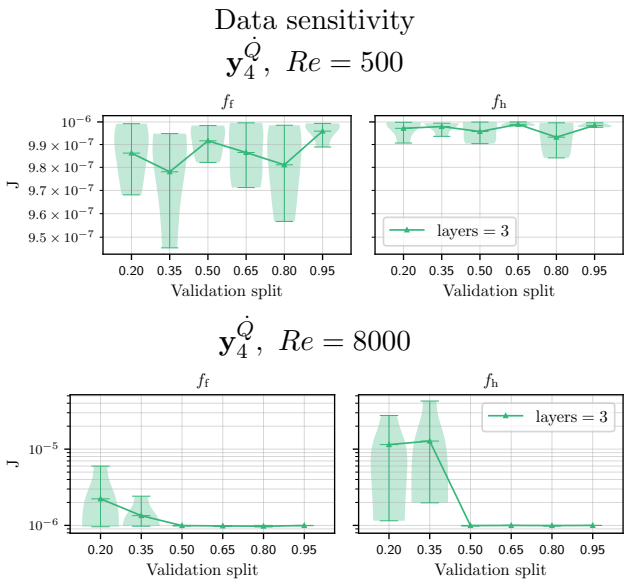


Figure 7.48: L-NN benchmark - J($\mathbf{y}_4^{\dot{Q}}$) against validation split of dataset $\boldsymbol{\omega}_4$ feature network trained with 128 neurons and 3 layers. The Lumped output data the network is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

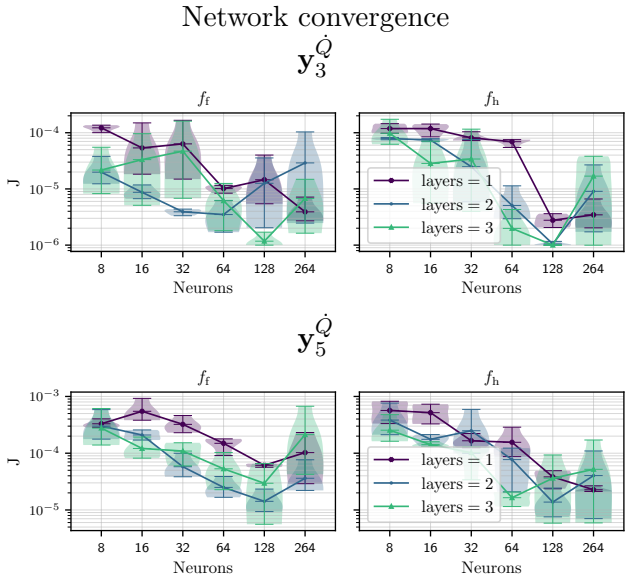


Figure 7.49: L-NN benchmark - $J(\mathbf{y}_{3,5}^{\dot{Q}})$ against Neurons and Layers of datasets –top to bottom– $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Lumped output data the network is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

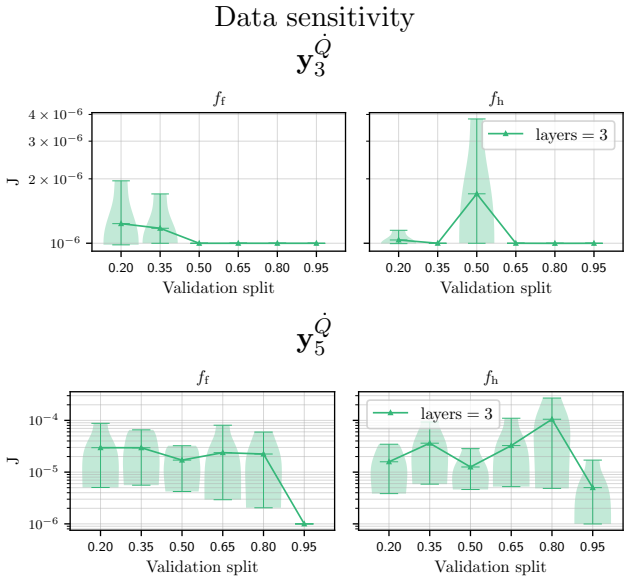


Figure 7.50: L-NN benchmark - $J(\mathbf{y}_{3,5}^{\dot{Q}})$ against validation split of datasets –top to bottom– $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with 128 neurons and 3 layers. The Lumped output data the network is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

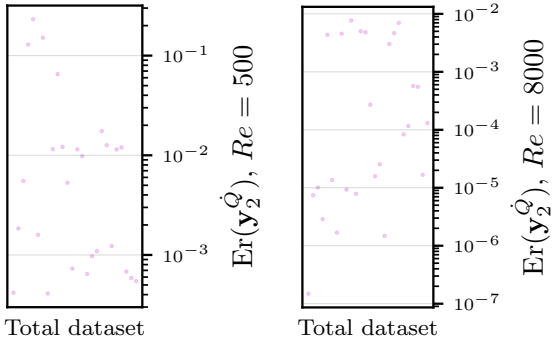


Figure 7.51: L-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_2^{\dot{Q}})$ of independent cases prediction from $\boldsymbol{\omega}_2$ feature network trained with a validation split of 35% vs high-fidelity data.

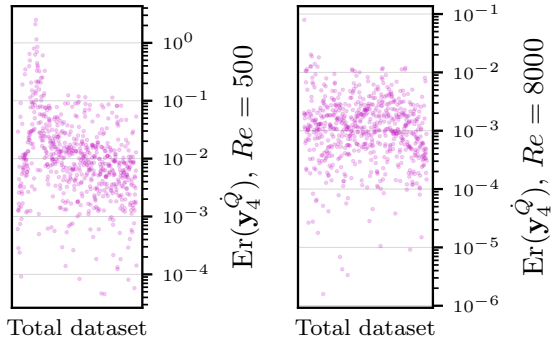


Figure 7.52: L-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_4^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature network trained with a validation split of 35% vs high-fidelity data.

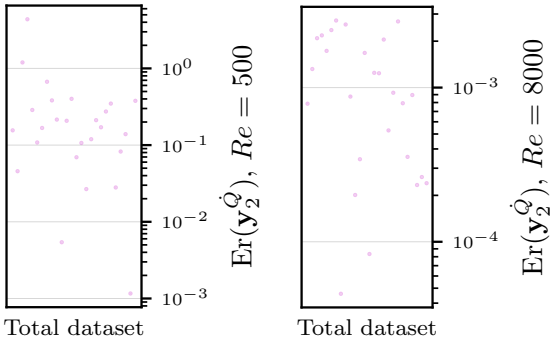


Figure 7.53: L-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_3^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_3$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

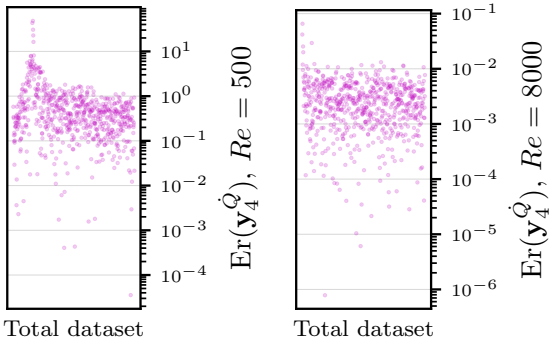


Figure 7.54: L-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_5^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_5$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

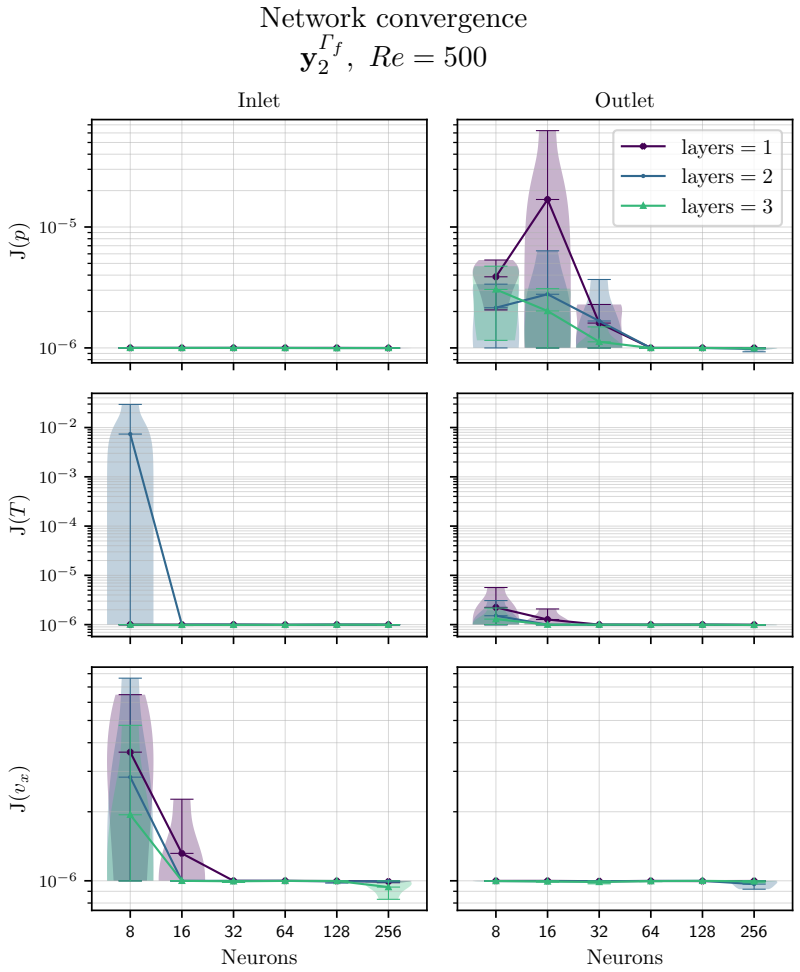


Figure 7.55: S-NN benchmark - J($\mathbf{y}_2^{\Gamma_f}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_2$ feature network trained with a validation split of 35%. The Spatial output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

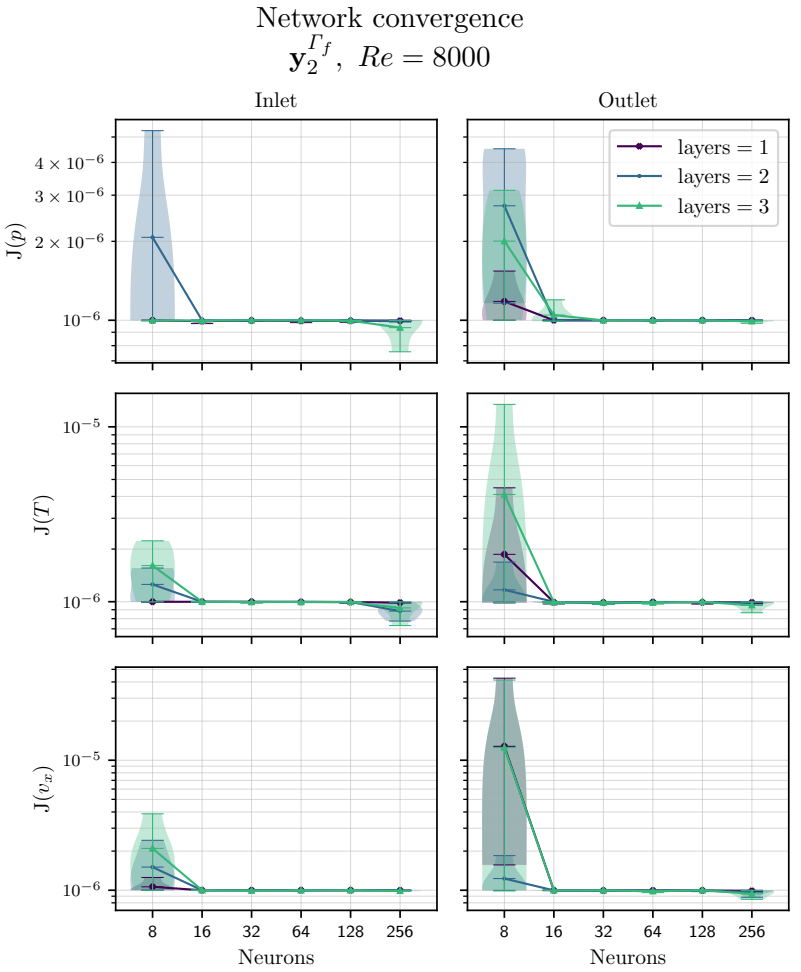


Figure 7.56: S-NN benchmark - J($\mathbf{y}_2^{\Gamma_f}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_2$ feature network trained with a validation split of 35%. The Spatial output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

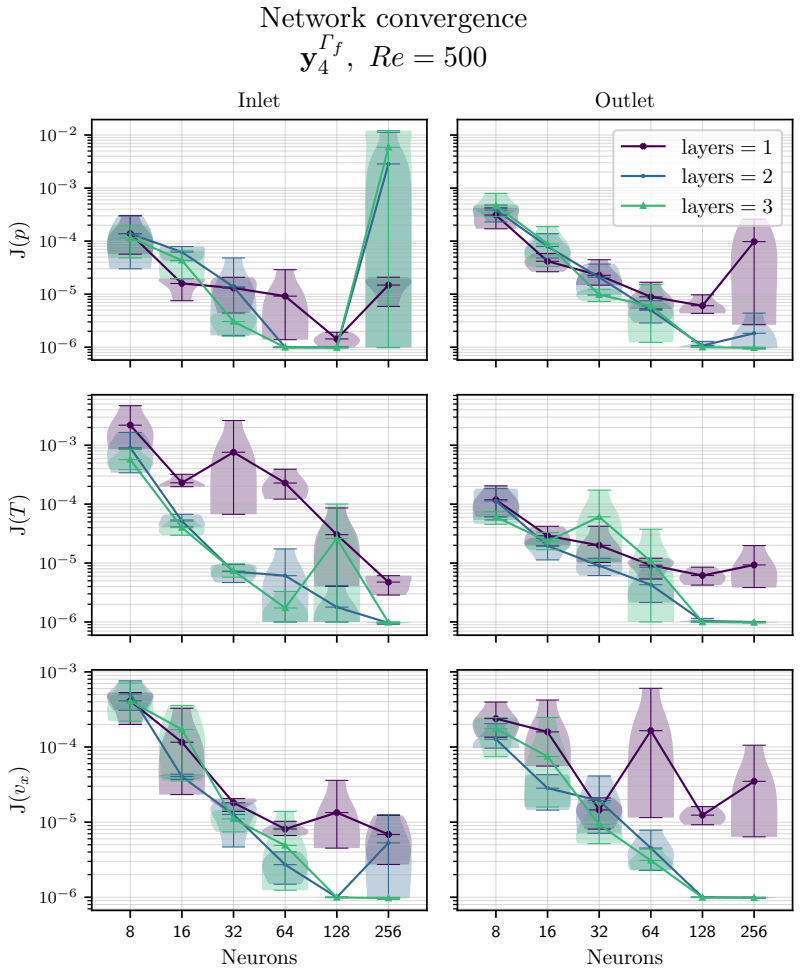


Figure 7.57: S-NN benchmark - J($\mathbf{y}_4^{\Gamma_f}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_4$ feature network trained with a validation split of 35%. The Spatial output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

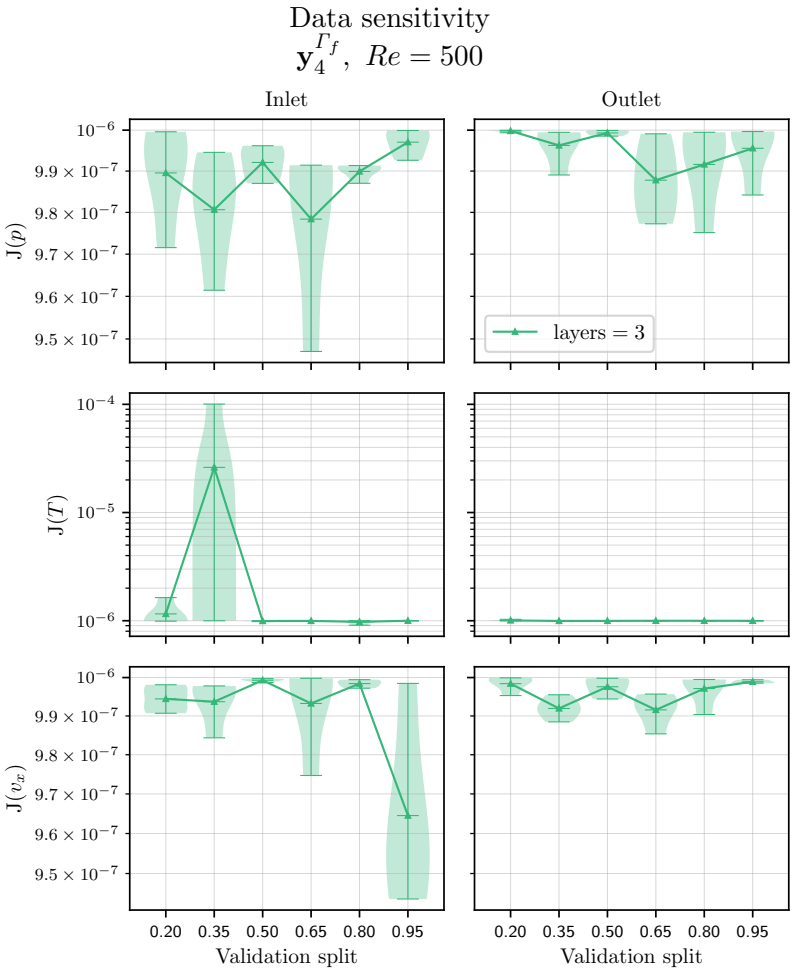


Figure 7.58: S-NN benchmark - J($\mathbf{y}_4^{\Gamma_f}$) against validation split of dataset $\boldsymbol{\omega}_4$ feature network trained with 128 neurons and 3 layers. The Spatial output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

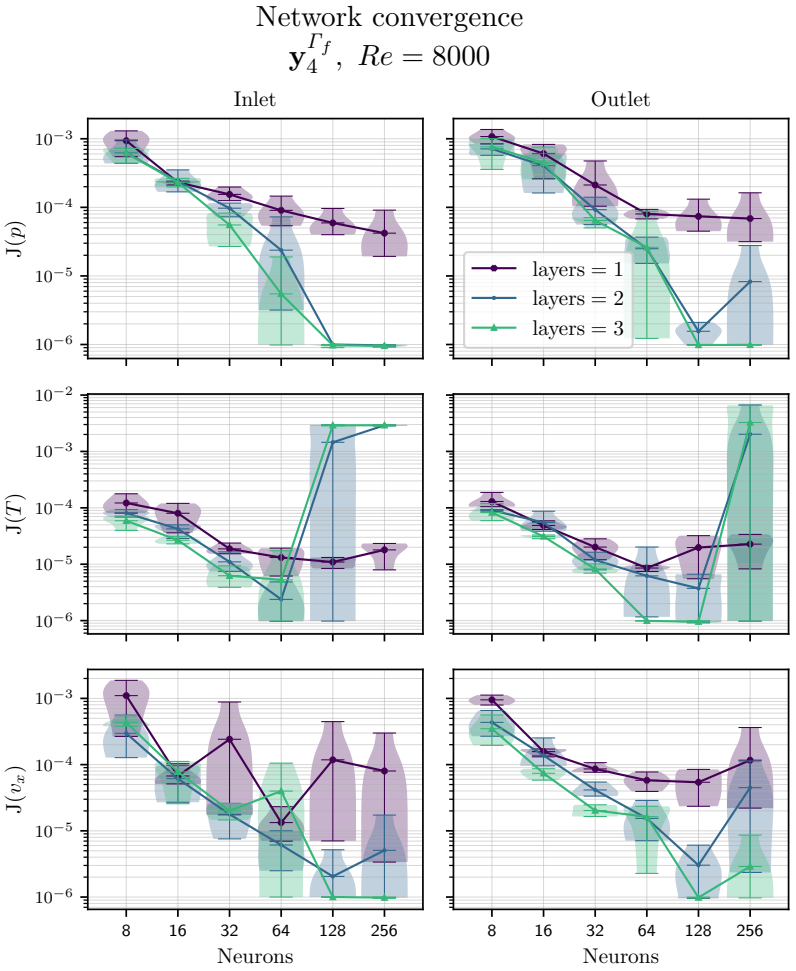


Figure 7.59: S-NN benchmark - J($\mathbf{y}_4^{\Gamma_f}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_4$ feature network trained with a validation split of 35%. The Spatial output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

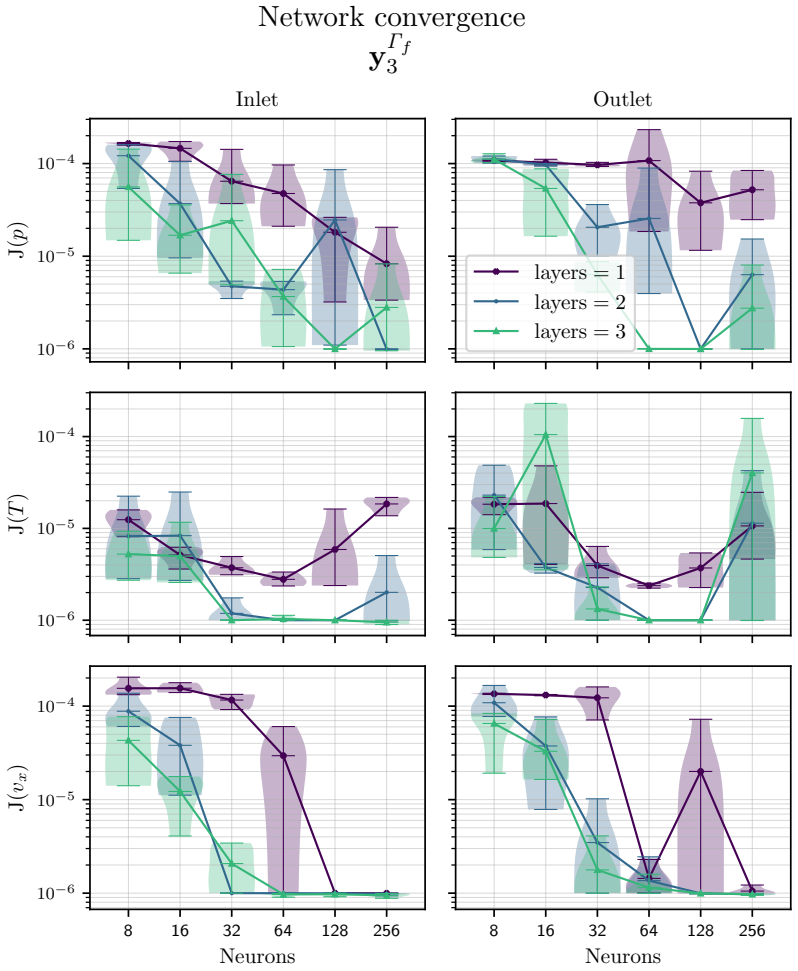


Figure 7.60: S-NN benchmark - J($\mathbf{y}_3^{\Gamma_f}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_3$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Spatial output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

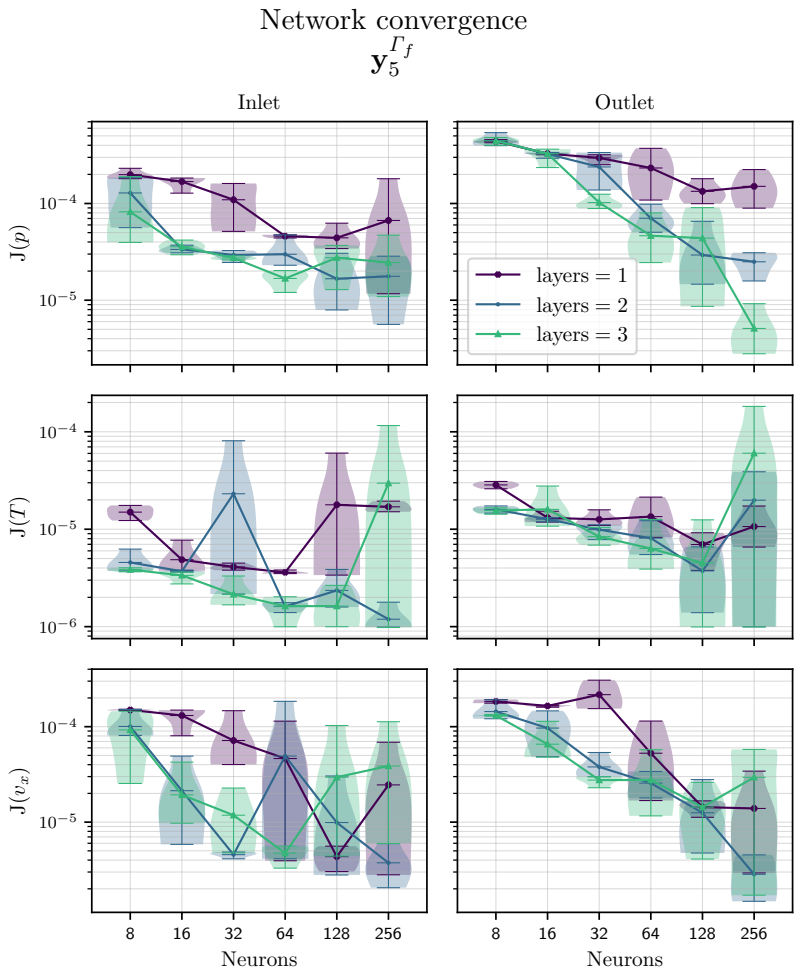


Figure 7.61: S-NN benchmark - $\mathrm{J}(\mathbf{y}_5^{\Gamma_f})$ against Neurons and Layers of dataset $\boldsymbol{\omega}_5$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Spatial output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

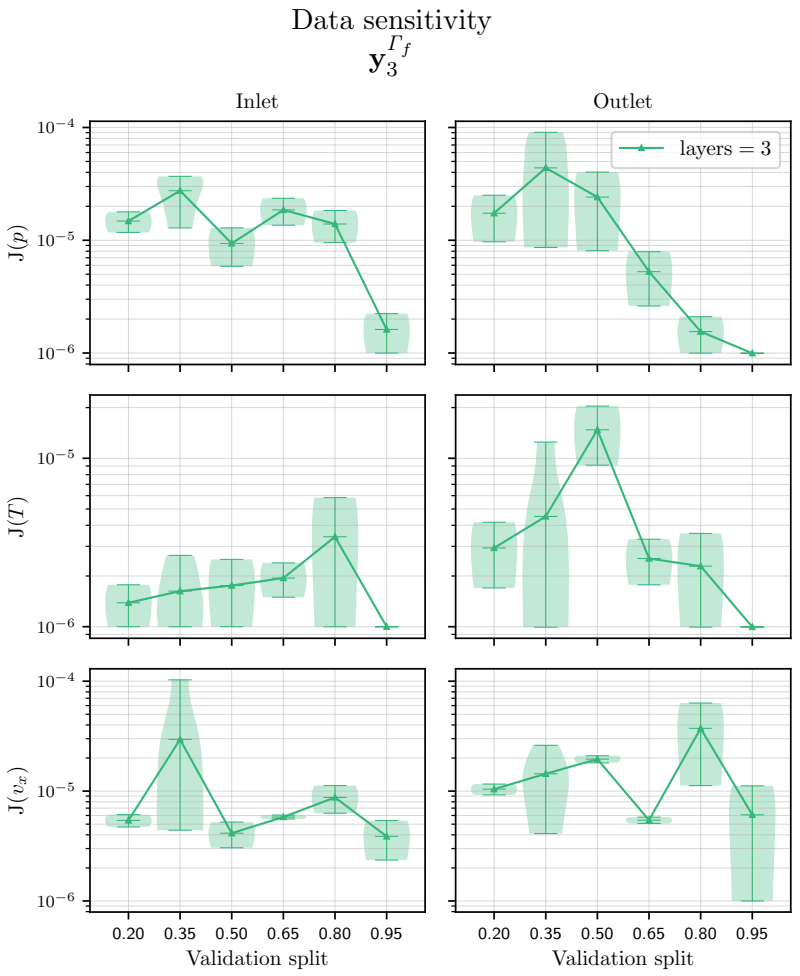


Figure 7.62: S-NN benchmark - $\mathrm{J}(\mathbf{y}_3^{\Gamma_f})$ against validation split of dataset $\boldsymbol{\omega}_3$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with 128 neurons and 3 layers. The Spatial output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

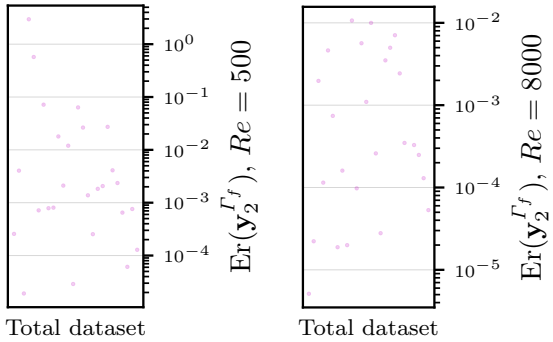


Figure 7.63: S-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_2^{\Gamma_f})$ of independent cases prediction from $\boldsymbol{\omega}_2$ feature network trained with a validation split of 35% vs high-fidelity data.

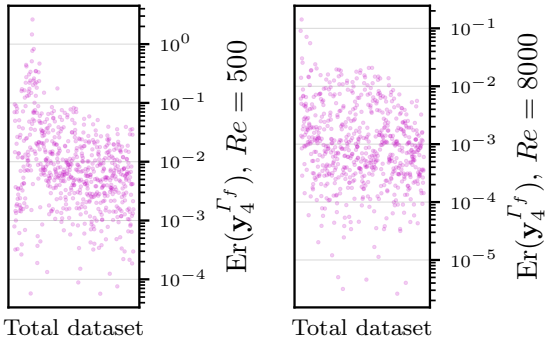


Figure 7.64: S-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_4^{\Gamma_f})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature network trained with a validation split of 35% vs high-fidelity data.

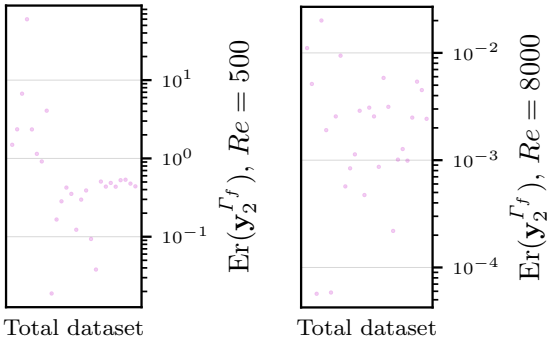


Figure 7.65: S-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_3^{\Gamma_f})$ of independent cases prediction from the $\boldsymbol{\omega}_3$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

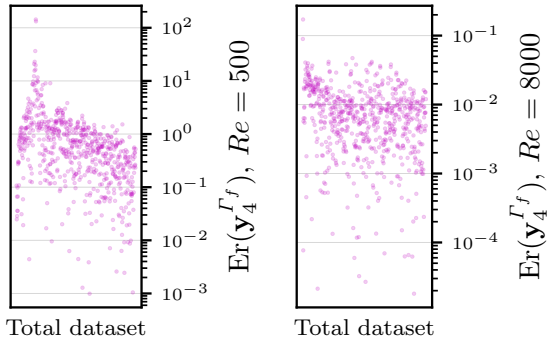


Figure 7.66: S-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_5^{\Gamma_f})$ of independent cases prediction from the $\boldsymbol{\omega}_5$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

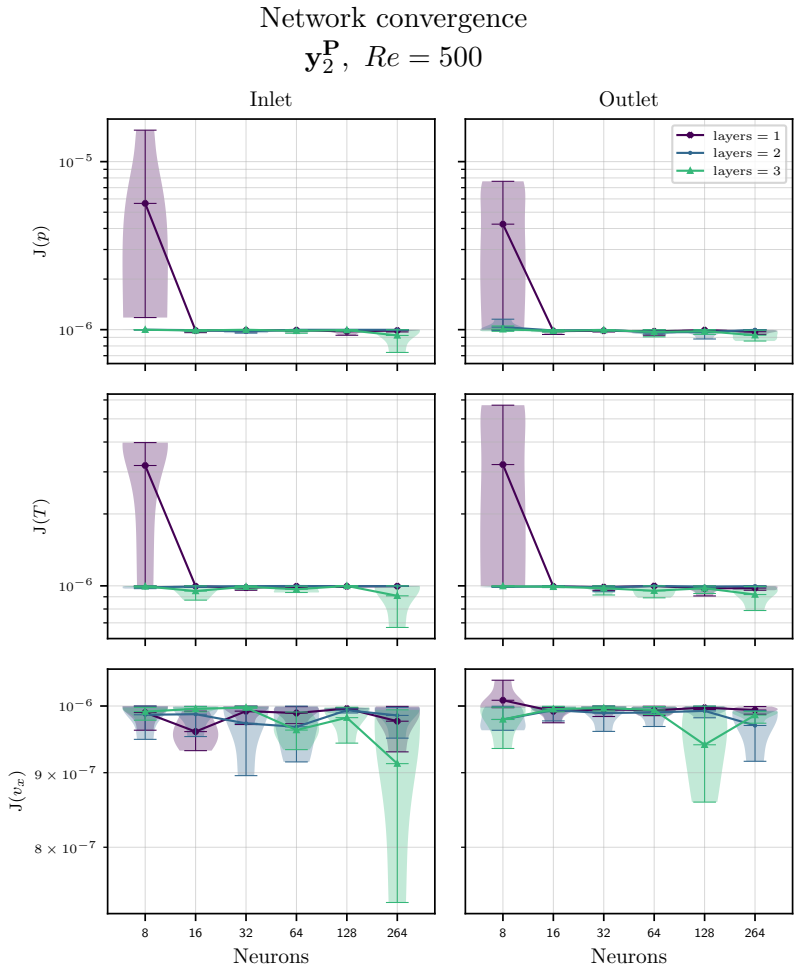


Figure 7.67: M-NN benchmark - J($\mathbf{y}_2^{\mathbf{P}}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_2$ feature network trained with a validation split of 35%. The Modal output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

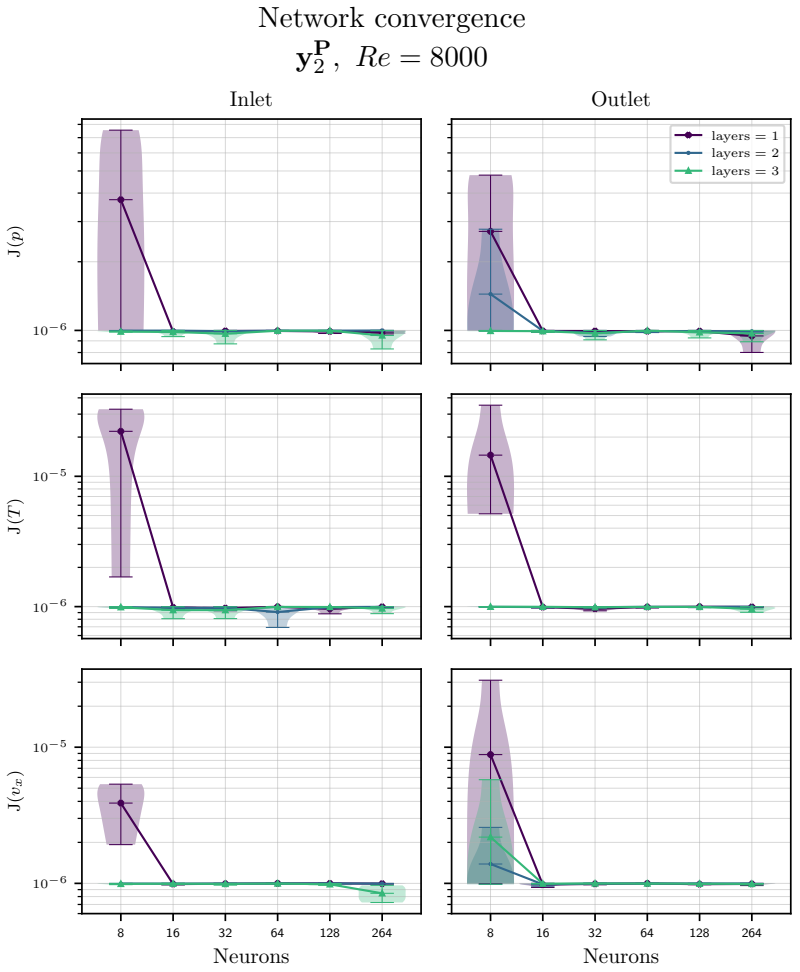


Figure 7.68: M-NN benchmark - J($\mathbf{y}_2^{\mathbf{P}}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_2$ feature network trained with a validation split of 35%. The Modal output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

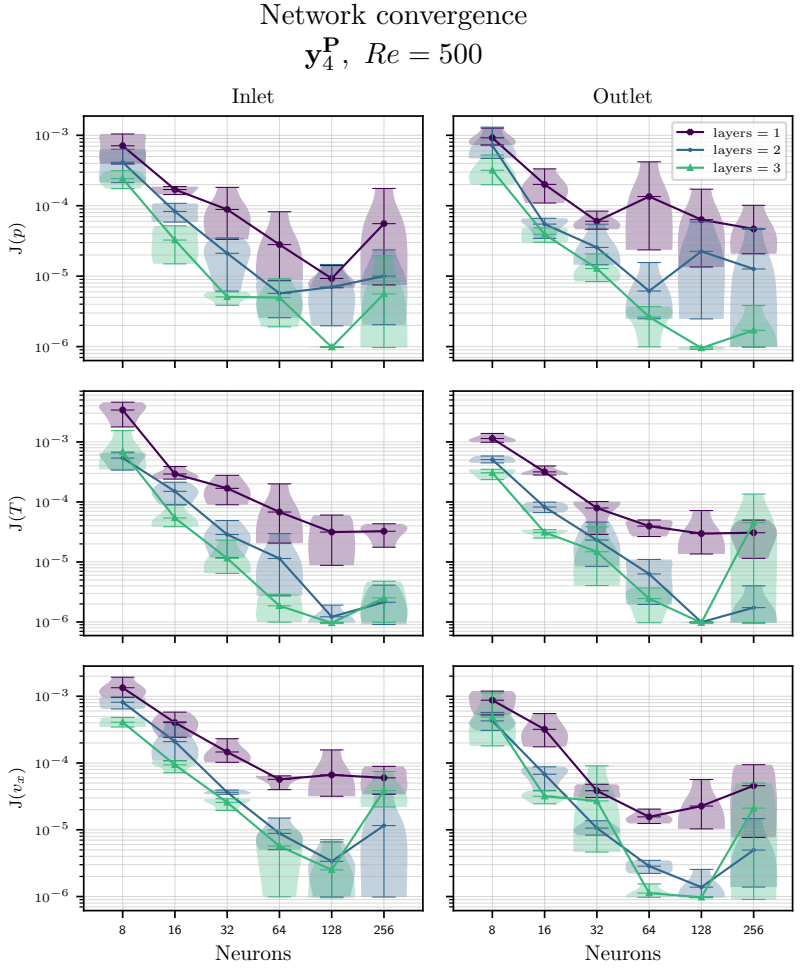


Figure 7.69: M-NN benchmark - J($\mathbf{y}_4^{\mathbf{P}}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_4$ feature network trained with a validation split of 35%. The Modal output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

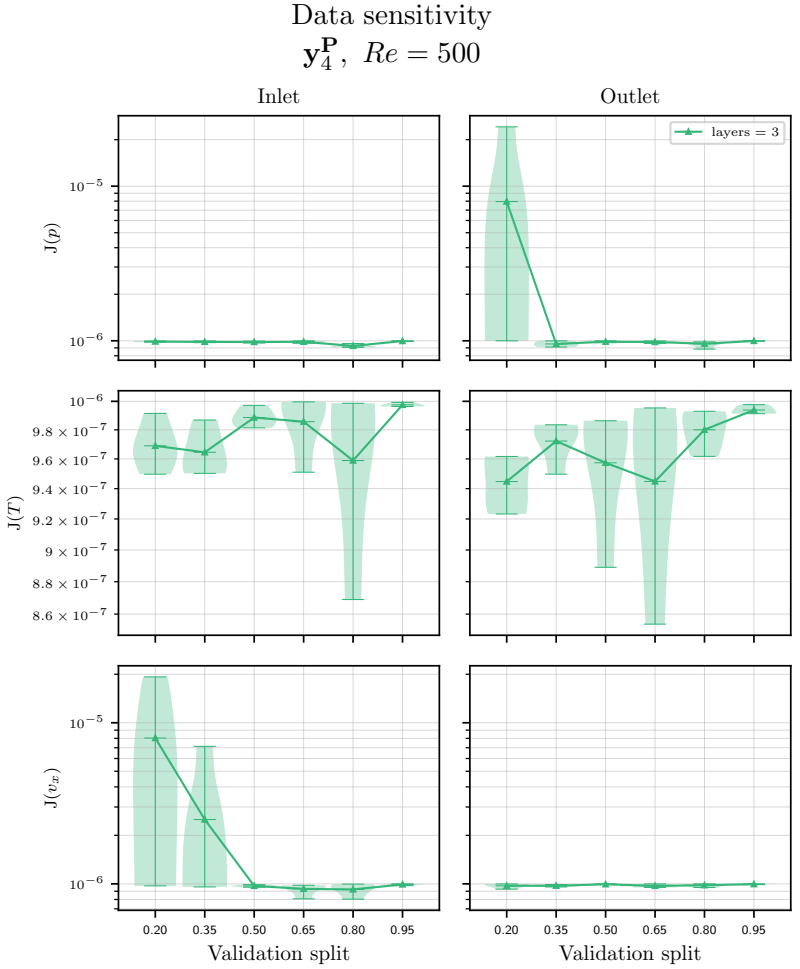


Figure 7.70: M-NN benchmark - J($\mathbf{y}_4^{\mathbf{P}}$) against validation split of dataset $\boldsymbol{\omega}_4$ feature network trained with 128 neurons and 3 layers. The Modal output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

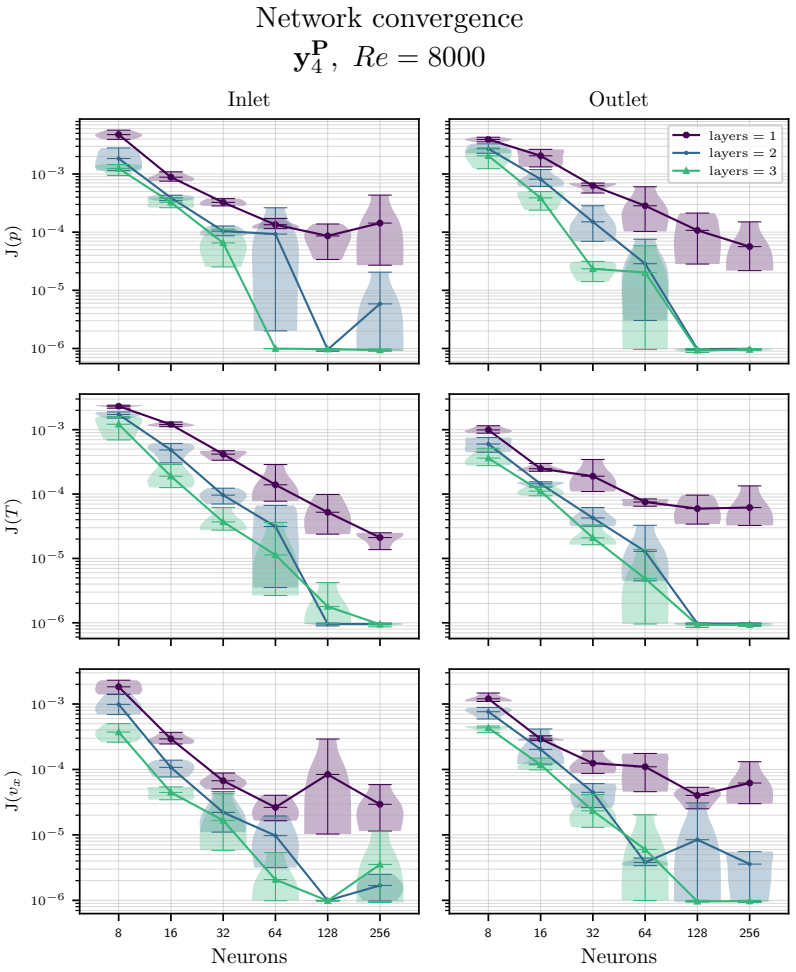


Figure 7.71: M-NN benchmark - J($\mathbf{y}_4^{\mathbf{P}}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_4$ feature network trained with a validation split of 35%. The Modal output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

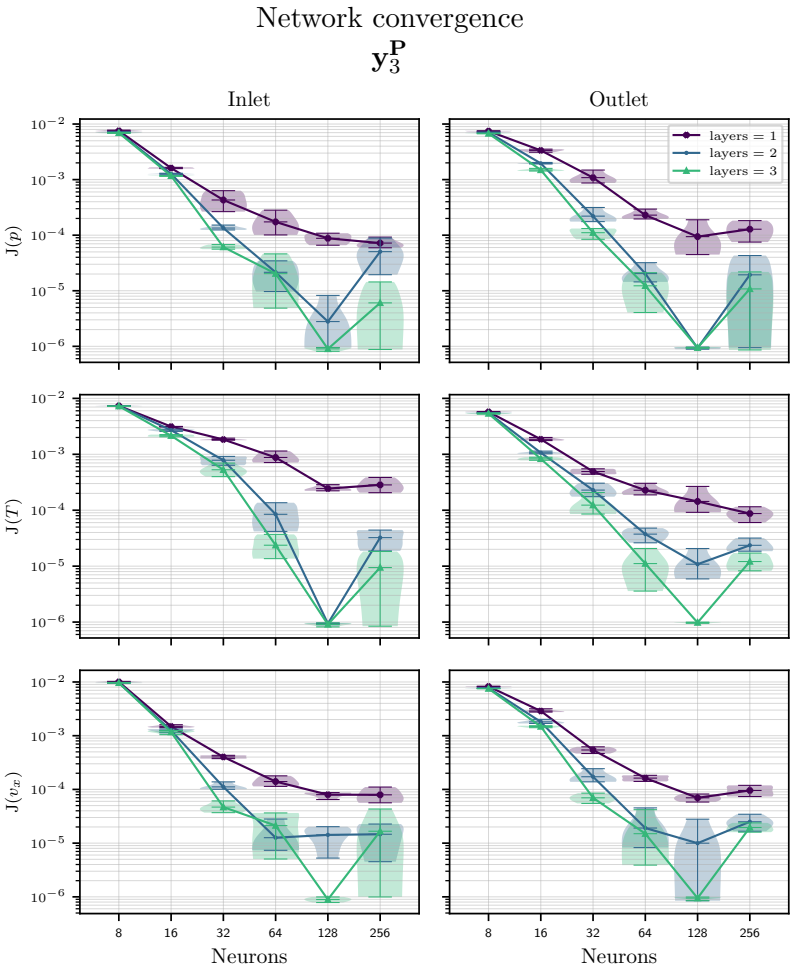


Figure 7.72: M-NN benchmark - J($\mathbf{y}_3^{\mathbf{P}}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_3$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Modal output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

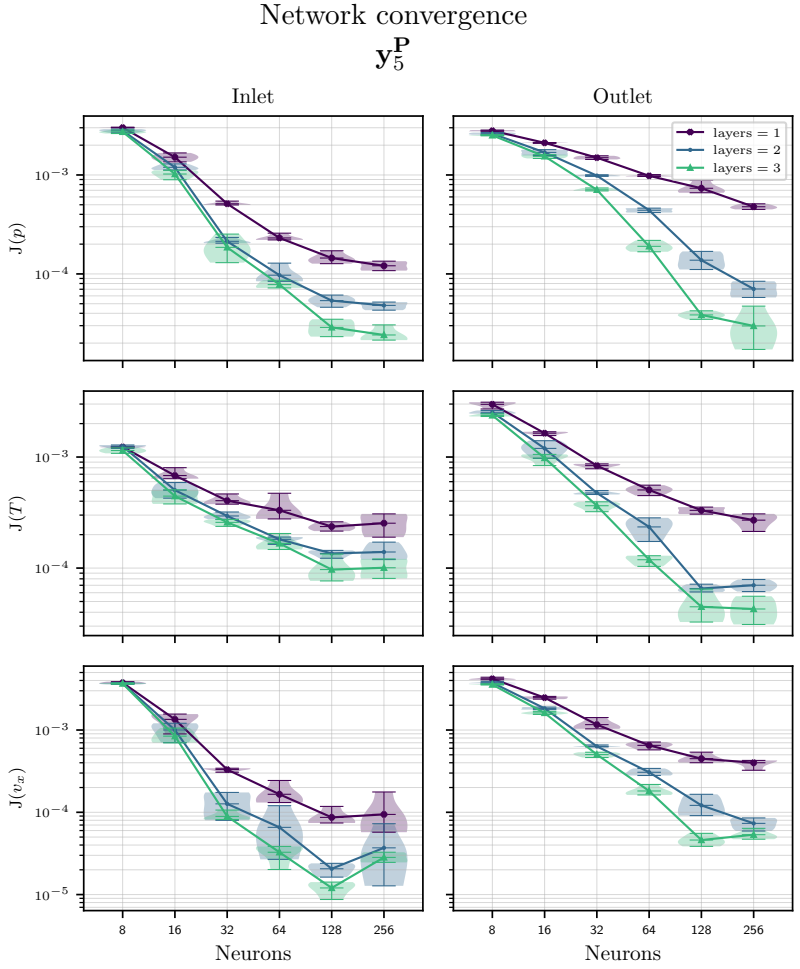


Figure 7.73: M-NN benchmark - J($\mathbf{y}_5^{\mathbf{P}}$) against Neurons and Layers of dataset $\boldsymbol{\omega}_5$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Modal output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

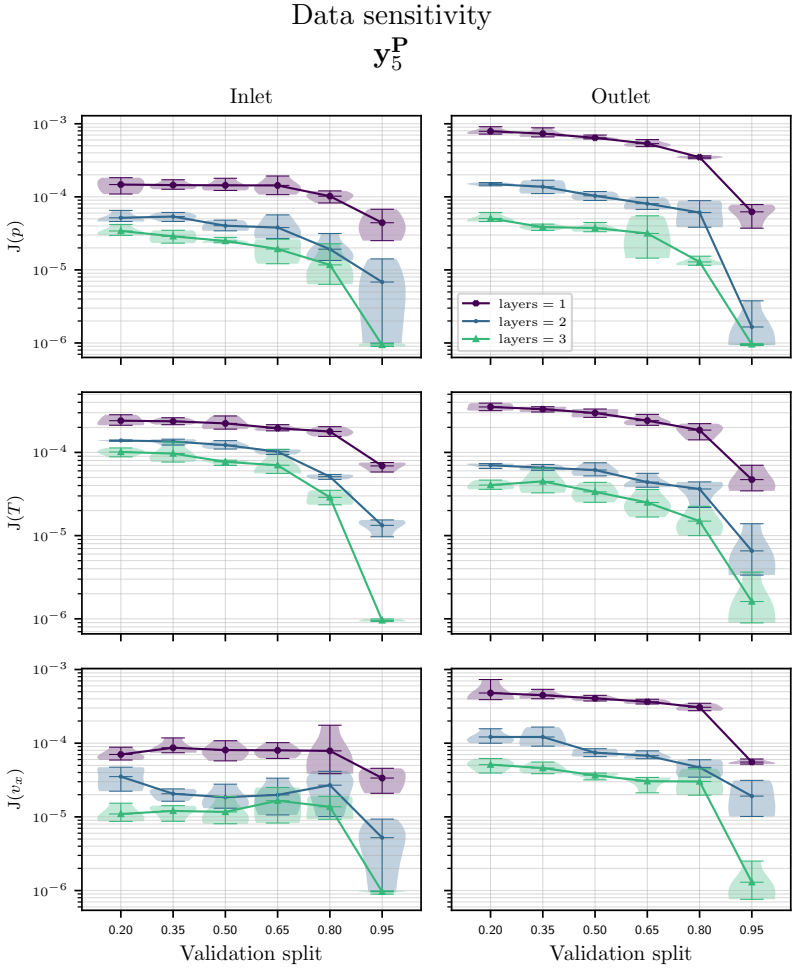


Figure 7.74: M-NN benchmark - J($\mathbf{y}_5^{\mathbf{P}}$) against validation split of dataset $\boldsymbol{\omega}_5$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with 128 neurons and 3 layers. The Modal output data the network is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

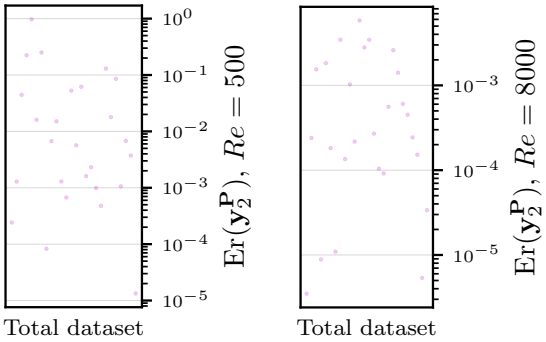


Figure 7.75: M-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_2^{\mathbf{P}})$ of independent cases prediction from $\boldsymbol{\omega}_2$ feature network trained with a validation split of 35% vs high-fidelity data.

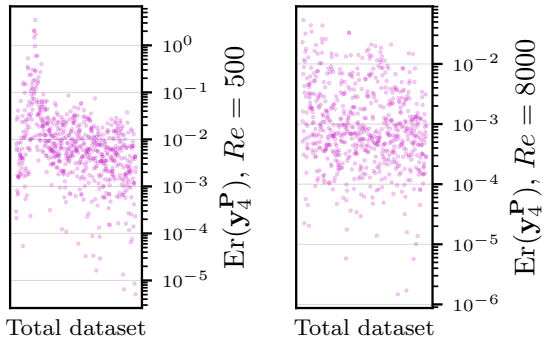


Figure 7.76: M-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_4^{\mathbf{P}})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature network trained with a validation split of 35% vs high-fidelity data.

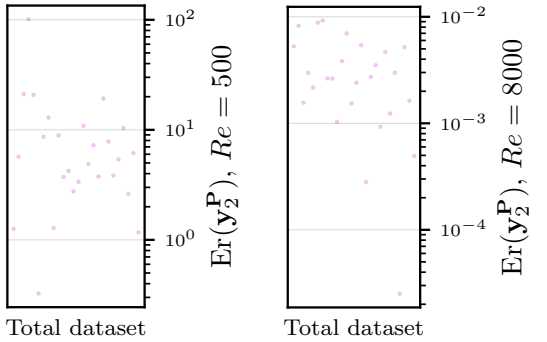


Figure 7.77: M-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_3^{\mathbf{P}})$ of independent cases prediction from the $\boldsymbol{\omega}_3$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

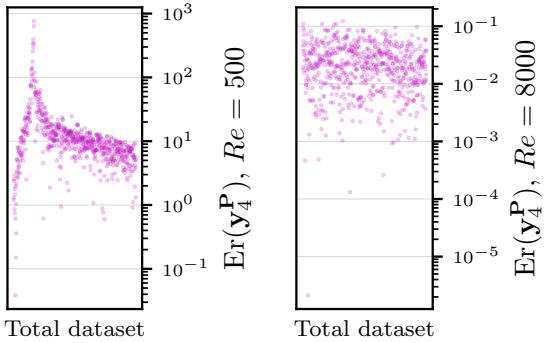


Figure 7.78: M-NN benchmark - Relative error $\mathrm{Er}(\mathbf{y}_5^{\mathbf{P}})$ of independent cases prediction from the $\boldsymbol{\omega}_5$ feature network trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

NN Preliminary findings

From the observations in Section 7.3.2, the sensitivity analysis of NN models with increasing complexities in the problem is discussed alongside findings from said analysis.

- **Convergence.** Highly sensitive to feature set size, higher $\boldsymbol{\omega}_{\mathrm{N}_i}$ did not reach the target loss or required deeper networks to do so. Highly sensitive to architecture, increasing neuron count up to a critical level enabled models to reach lower loss values.
- **Consistency.** Highly sensitive to $\boldsymbol{\omega}_{\mathrm{N}_i}$ at $\mathbf{y}_i^{\dot{Q},\Gamma_f}$. On $\mathbf{y}_i^{\mathbf{P}}$, sensitivity increases at lower layer counts.
- **Over-fitting.** Heavily sensitive to flow regime regardless of feature set size $\boldsymbol{\omega}_{\mathrm{N}_i}$ and output size $\mathbf{x}^{\mathrm{N}_x}$. $Re = 500$ has values further apart from each other $-\overline{\dot{Q}} = [-12, 5]-$ compared to $Re = 8000$ $-\overline{\dot{Q}} = [0, 1]-$, highly non-linear behaviour is assumed[2]. Over-fitting is enhanced by increasing $\boldsymbol{\omega}_{\mathrm{N}_i}$ and $\mathbf{x}^{\mathrm{N}_x}$, sensitivity of the model to increasing input-output sizes is non-negligible.
- **Data cost and sensitivity.** Due to the observed behaviour in over-fitting at $\boldsymbol{\omega}_5$, a larger training set might be required, implying high sensitivity to $\boldsymbol{\omega}_{\mathrm{N}_i}$ size and to non-linear behaviour in the function.
- **Computational cost.** Highly sensitive to increasing values of the combination of all network parameters assessed in this study. Critical architectures decrease computational cost by reaching target loss in early epoch stages.

Overall, the main advantage of using an NN model is its ability to address non-linearities in the data. However, the computational cost of a deep network hinders the gains from describing non-linear behaviour better. Over-fitting in low Re for $\boldsymbol{\omega}_5$ is most likely related to the extreme changes observed at low Re $-\overline{\dot{Q}} = [-12, 8]-$, described in Chapter 6.3, especially compared to flow behaviour in turbulence $-\overline{\dot{Q}} = [0, 1]-$, which leads to increases non-linearities of the function. The algorithm needs a deeper architecture to approach the problem with input features $\boldsymbol{\omega}_5$, but has enough accuracy to approach it with $\boldsymbol{\omega}_4$ with the current architecture of 3 layers and 128 neurons. The low-rank output $\mathbf{x}^{\mathbf{P}}$ has a performance closer to $\mathbf{x}^{\dot{Q}}$ while maintaining the flow variable information from $\mathbf{x}^{\Gamma_f}$, which makes it the most feasible approach to observe flow behaviour in more detail. This assessment satisfies Objectives VII.IV and VII.V.

7.3.3 Gaussian Processes

The description of a GP model and the training process is given in Section 4.5. The parameters assessed to find an ideal kernel for each GP are shown in Table 7.5, where the preliminary study used to select the Matern and RBF kernels for this assessment is presented in Appendix D.2. The marginal log-likelihood $\mathrm{L_J}$ is the performance index of the GP training, but a loss J is computed to address convergence. The GP models' benchmark for $\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}(\boldsymbol{\omega}_{\mathrm{N}_i})$ with output

[2] The values are normalised to avoid the influence of magnitude differences in the performance of the models, but the non-linear behaviour remains part of the dataset.

Parameter	
Kernels [132]	{RBF Kernel (RBFK), Matern Kernel (MK)}
Validation/training split	{5%, 35%, 50%, 65%, 95%}

Table 7.5: GP benchmark - Numerical parameters for the Gaussian process benchmark study. $^{*}\nu$ is required for the Matern kernel, RBF ignores this parameter.

dimensions $\mathrm{N}_x = \{\dot{Q},\ \Gamma_f,\ \mathbf{P}\}$ and input features $\mathrm{N}_i = [2,\ 5]$, trained with validation set sizes from Table 7.5 are shown in Figures 7.79 to 7.84 for lumped $\mathbf{y}^{\dot{Q}}_{\mathrm{N}_i}$, 7.89 to 7.95 for spatial $\mathbf{y}^{\Gamma_f}_{\mathrm{N}_i}$ and 7.100 to 7.107 for modal $\mathbf{y}^{\mathbf{P}}_{\mathrm{N}_i}$. The data sensitivity study uses a Matern kernel while the kernel convergence study is evaluated from models trained with 65% of the available high fidelity data –or 35% validation split as shown in Table 7.4– except for $\mathbf{y}^{\Gamma_f}_5$ which is trained with 5% of the available high fidelity data[3]. These figures show a *violin plot shade*, representing the range of values of the loss from 4 independent models trained with the same architecture but randomly initialised weight parameters. The varying thickness of this shade represents the density of values within the region, and the mean value connects the plot line. The relative error $\mathrm{Er}\left(\mathbf{y}^{\mathrm{N}_x}_{\mathrm{N}_i}\right)$ of the best kernel predictions against high fidelity data is shown in ascending order of $\overline{\dot{Q}}$ values in Figures 7.85 to 7.88 for lumped $\mathbf{y}^{\dot{Q}}_{\mathrm{N}_i}$, 7.63 to 7.89 for spatial $\mathbf{y}^{\Gamma_f}_{\mathrm{N}_i}$ and 7.108 to 7.100 for modal $\mathbf{y}^{\mathbf{P}}_{\mathrm{N}_i}$.

- **Convergence.** Analysing the loss benchmark figures, the kernel and feature set size $\boldsymbol{\omega}_{\mathrm{N}_i}$ have no significant impact in the J of $\mathbf{x}^{\dot{Q}}$, where the value stays around $\mathrm{J}(\mathbf{y}^{\dot{Q}}_{\mathrm{N}_i}) = 10^0$. At $\boldsymbol{\omega}_2$, the model with output $\mathbf{x}^{\dot{Q}}$ reached $\mathrm{J}(\mathbf{y}^{\dot{Q}}_{\mathrm{N}_i}) = 10^{-1}$ with a training data set size of arounf 50%. With $\mathbf{x}^{\Gamma_f}$ the values range $\mathrm{J}(\mathbf{y}^{\Gamma_f}_{\mathrm{N}_i}) = [10^{-1},\ 10^0]$ where the Matern kernel is consistently the lower value. However, some outlier RBF kernel values reach $\mathrm{J}(\mathbf{y}^{\Gamma_f}_{\mathrm{N}_i}) = 10^{-2}$, in the outlet pressure of $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$, and temperatures of $\boldsymbol{\omega}_4$ at $Re = 8000$. With $\mathbf{x}^{\mathbf{P}}$ most models behave similar to $\mathbf{x}^{\dot{Q}}$, with outliers in the RBF kernel of $\mathrm{J}(\mathbf{y}^{\mathbf{P}}_{3,5}) = 10^{-2}$ in the outlet pressure.

- **Consistency.** From the loss benchmark figures, kernels are highly consistent regardless of the random initialisation, dimension $\mathbf{x}^{\mathrm{N}_x}$ and features $\boldsymbol{\omega}_{\mathrm{N}_i}$. However, there is a non-negligible range of values in $\mathbf{y}^{\dot{Q}}_2$ at $Re = 500$ when changing the training data size.

- **Over-fitting.** When observing $\mathrm{Er}(\mathbf{y}^{\mathrm{N}_x}_{\mathrm{N}_i})$ for the Matern kernel, there are two clusters of high density with an extreme shift towards higher error values at low Re and $\overline{\dot{Q}}$ in the lumped model $\mathrm{x}^{\dot{Q}}$, while spatial x^{Γ_f} and modal $\mathrm{x}^{\mathbf{P}}$ have a cluster of high density with descending order of errors with increasing $\overline{\dot{Q}}$ values. The higher error density area is in the range of $\mathrm{Er}(\mathbf{y}^{\dot{Q}}_4) = [10^{-2},\ 10^0]$ in lumped, and $\mathrm{Er}(\mathbf{y}^{\Gamma_f,\mathbf{P}}_4) = [10^{-4},\ 10^1]$ in spatial and modal for $Re = 500$, with the shift in lumped going up to $\mathrm{Er}(\mathbf{y}^{\dot{Q}}_4) = [10^{-1},\ 10^1]$, and

[3]Training with a larger dataset on this model was computationally unfeasible due to memory usage.

$\mathrm{Er}(\mathbf{y}_4^{\mathrm{N}_x}) = [10^{-3},\ 10^{-1}]$ for high $Re = 8000$. The high errors for lower Re indicate over-fitting, especially in the lower $\overline{\dot{Q}}$ –with negative $\overline{\dot{Q}}$ values–, while the model predicts higher Re with more accuracy. This is observed to be more extreme for $\boldsymbol{\omega}_5$ features, when Re is used for training, where the high density areas in are found within $\mathrm{Er}(\mathbf{y}_5^{\dot{Q}}) = [10^{0},\ 10^{2}]$ on $Re = 500$, with a shift up to $\mathrm{Er}(\mathbf{y}_5^{\dot{Q}}) = [10^{1},\ 10^{2}]$. Note that there is no information available from $\mathbf{y}_{\mathrm{N}_i}^{\Gamma_f}$ and $\mathbf{y}_{\mathrm{N}_i}^{\mathbf{P}}$ due to the memory cost of the prediction with a large covariance matrix, with 5 variables and over 5 000 training cases.

- **Data cost and sensitivity.** From the loss benchmark figures showing the validation split, it seems that increasing the training set does not have a significant impact in J of the Matern kernel. High over-fitting is observed even with a training set of 65%.

- **Computational cost.** The GP models have a cubic computational complexity due to their use of a covariance matrix in both the training and prediction stages. This makes the increase of features $\boldsymbol{\omega}_{\mathrm{N}_i}$ the most impactful change in terms of computational cost. With a Matern kernel for validation of 65%, the CPU time of the models increases as $t(\mathbf{y}_2^{\dot{Q}}) \approx 1$ s, $t(\mathbf{y}_3^{\dot{Q}}) \approx 2$ s, $t(\mathbf{y}_4^{\dot{Q}}) \approx 3$ s, $t(\mathbf{y}_5^{\dot{Q}}) \approx 520$ s at the lowest output dimension $\mathbf{x}^{\dot{Q}}$. With $\mathbf{x}^{\Gamma_f}$, the CPU time increases as $t(\mathbf{y}_2^{\Gamma_f}) \approx 5$ s, $t(\mathbf{y}_3^{\Gamma_f}) \approx 60$ s, $t(\mathbf{y}_4^{\Gamma_f}) \approx 360$ s, $t(\mathbf{y}_5^{\Gamma_f}) \approx 3\,000$ s, where the model using $\boldsymbol{\omega}_5$ was trained with 10% of the available data. Lastly, the modal output $\mathbf{x}^{\mathbf{P}}$ has CPU time increase as $t(\mathbf{y}_2^{\mathbf{P}}) \approx 2$ s, $t(\mathbf{y}_3^{\mathbf{P}}) \approx 4$ s, $t(\mathbf{y}_4^{\mathbf{P}}) \approx 11$ s, $t(\mathbf{y}_5^{\mathbf{P}}) \approx\ 1\,620$s. It is worth noting that the RBF kernel on $\mathbf{y}_4^{\Gamma_f}$ had a CPU time of $t \approx 1\,200$ s, several orders of magnitude over the Matern kernel.

Figures 7.111 and 7.112 show the *confidence region* of the $\mathbf{y}_4^{\mathbf{P}}\left(\boldsymbol{\omega}_4, \mathbf{x}^{\mathbf{P}}\right)$ models trained on $Re = \{500,\ 8000\}$ as a grey surface surrounding the prediction surface provided by the model. This data allows us to understand where the model requires more information to fit the function better, which in the case of the visible surfaces is close to low amplitude and wavenumbers.

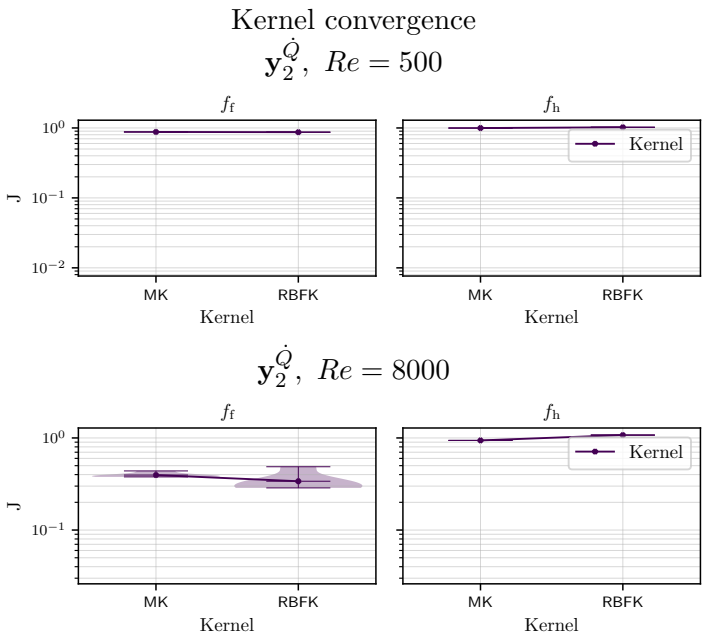


Figure 7.79: L-GP benchmark - J($\mathbf{y}_2^{\dot{Q}}$) against kernels of dataset $\boldsymbol{\omega}_2$ feature GP trained with a validation split of 35%. The Lumped output data the GP is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

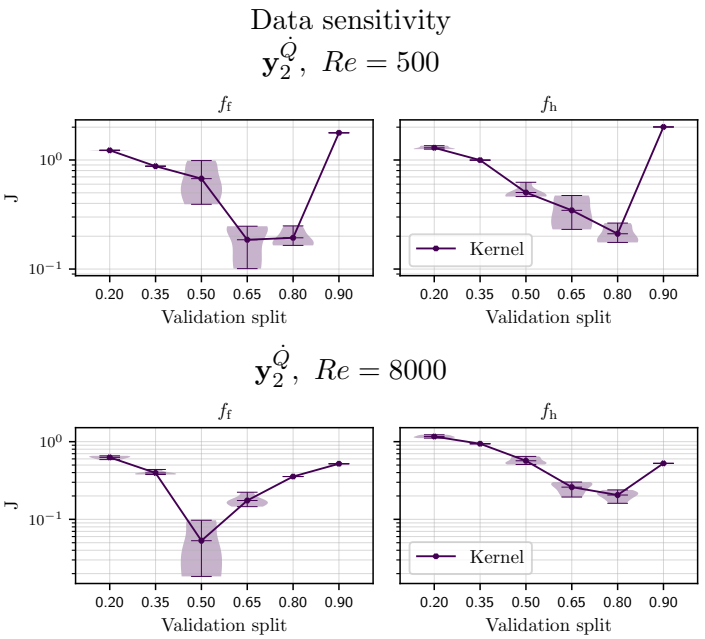


Figure 7.80: L-GP benchmark - J($\mathbf{y}_2^{\dot{Q}}$) against validation split of dataset $\boldsymbol{\omega}_2$ feature GP trained with a Matern kernel. The Lumped output data the GP is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

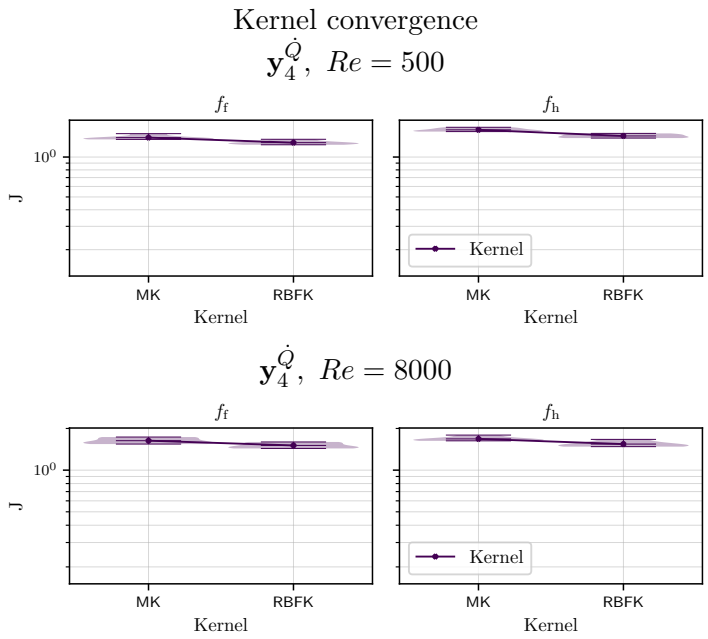


Figure 7.81: L-GP benchmark - J($\mathbf{y}_4^{\dot{Q}}$) against kernels of dataset $\boldsymbol{\omega}_4$ feature GP trained with a validation split of 35%. The Lumped output data the GP is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

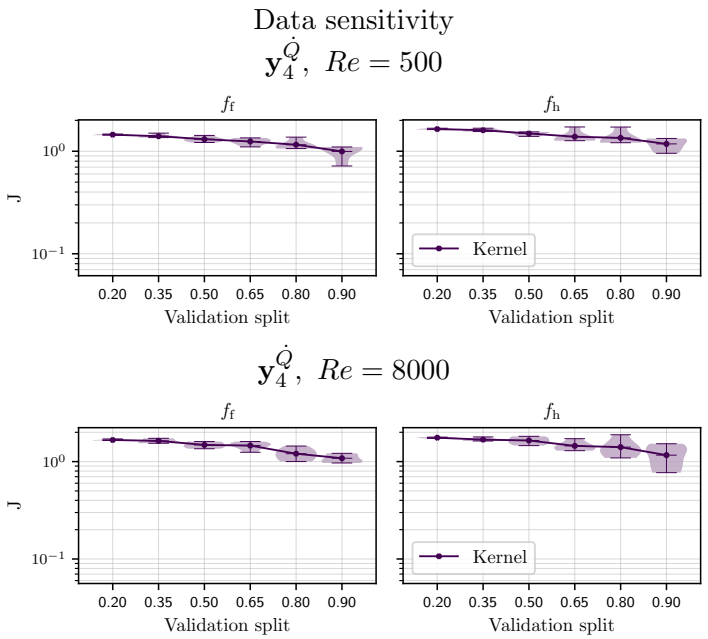


Figure 7.82: L-GP benchmark - J($\mathbf{y}_4^{\dot{Q}}$) against validation split of dataset $\boldsymbol{\omega}_4$ feature GP trained with a Matern kernel. The Lumped output data the GP is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

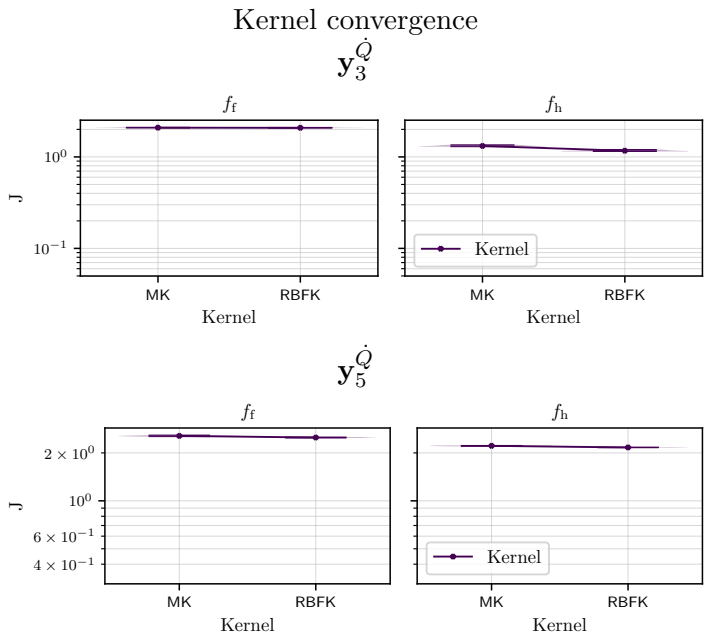


Figure 7.83: L-GP benchmark - $\mathrm{J}(\mathbf{y}_{3,5}^{\dot{Q}})$ against kernels of datasets –top to bottom– $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Lumped output data the GP is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

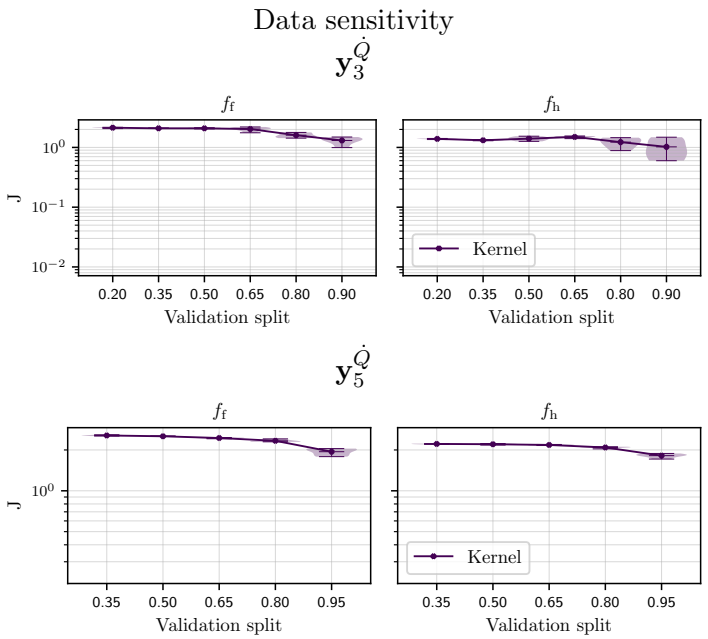


Figure 7.84: L-GP benchmark - $\mathrm{J}(\mathbf{y}_{3,5}^{\dot{Q}})$ against validation split of datasets –top to bottom– $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a Matern kernel. The Lumped output data the GP is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

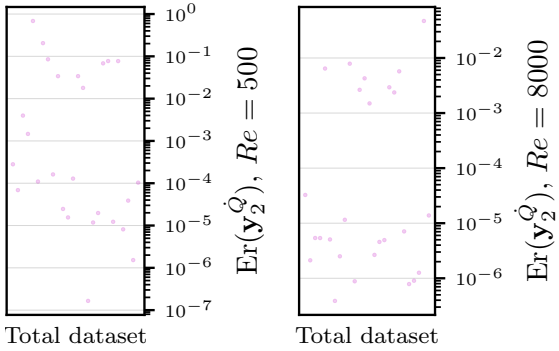


Figure 7.85: L-GP benchmark - Relative error $\mathrm{Er}(\mathbf{y}_2^{\dot{Q}})$ of independent cases prediction from $\boldsymbol{\omega}_2$ feature GP trained with a validation split of 35% vs high-fidelity data.

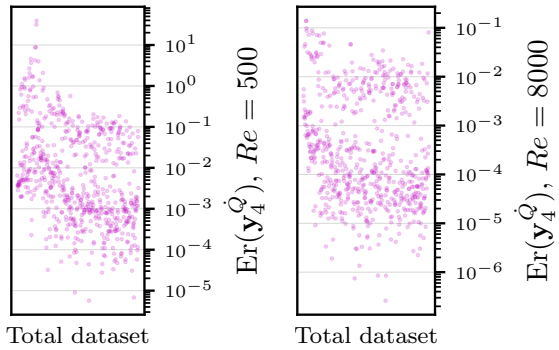


Figure 7.86: L-GP benchmark - Relative error $\mathrm{Er}(\mathbf{y}_4^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature GP trained with a validation split of 35% vs high-fidelity data.

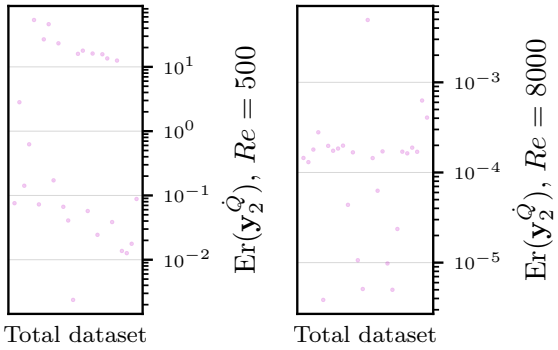


Figure 7.87: L-GP benchmark - Relative error $\mathrm{Er}(\mathbf{y}_3^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_3$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

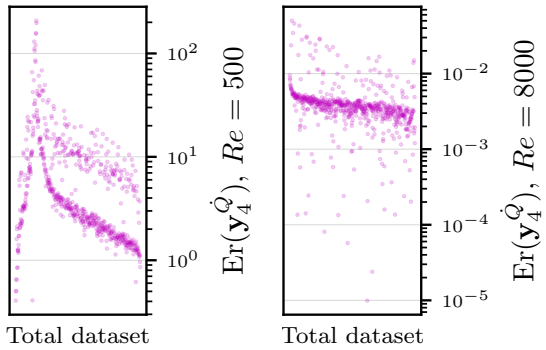


Figure 7.88: L-GP benchmark - Relative error $\mathrm{Er}(\mathbf{y}_5^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_5$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

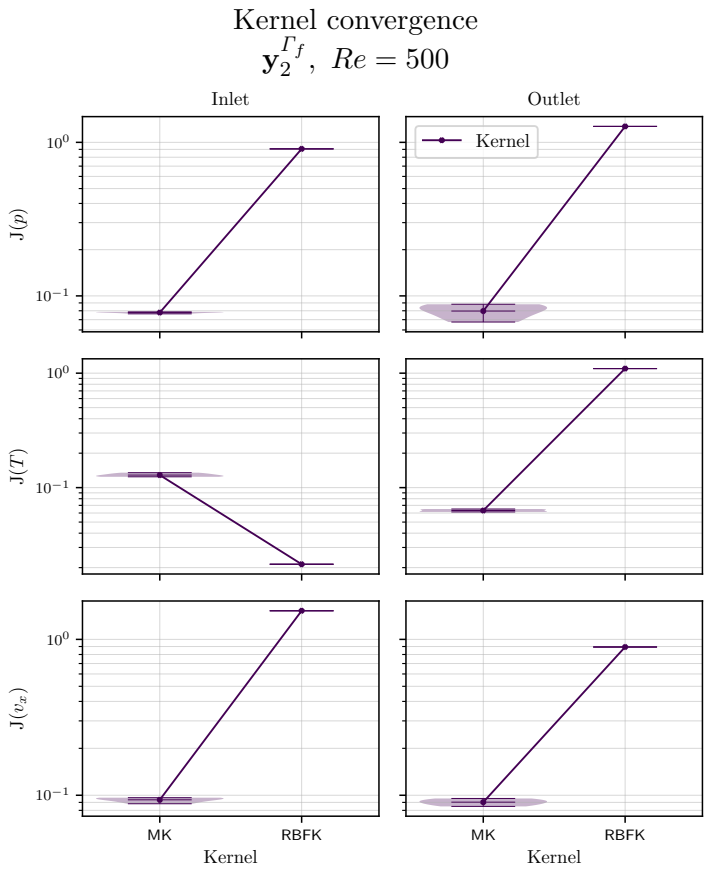


Figure 7.89: S-GP benchmark - J($\mathbf{y}_2^{\Gamma_f}$) against kernels of dataset $\boldsymbol{\omega}_2$ feature GP trained with a validation split of 35%. The Spatial output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

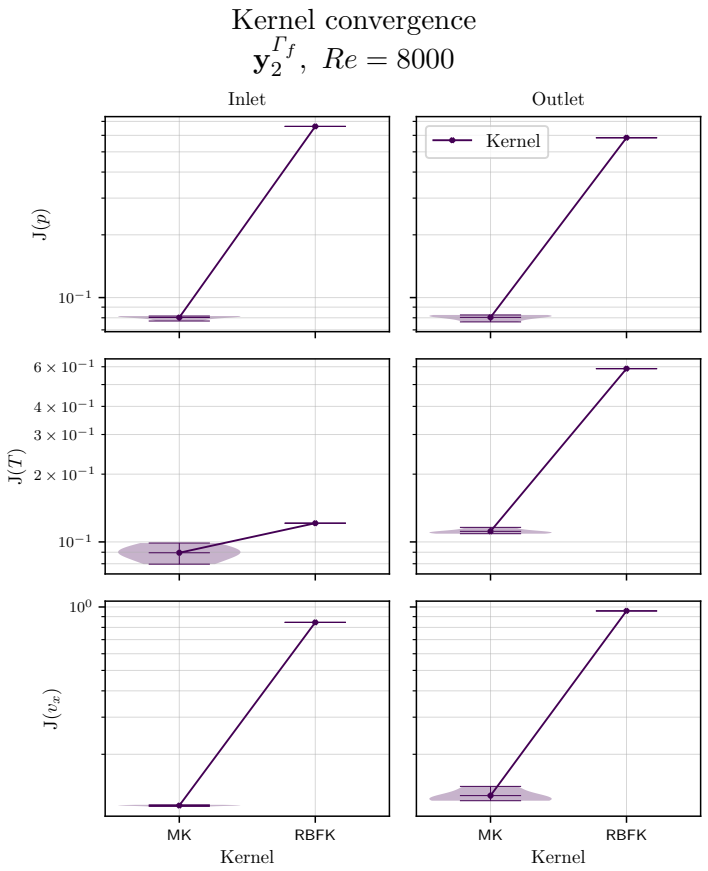


Figure 7.90: S-GP benchmark - J($\mathbf{y}_2^{\Gamma_f}$) against kernels of dataset $\boldsymbol{\omega}_2$ feature GP trained with a validation split of 35%. The Spatial output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

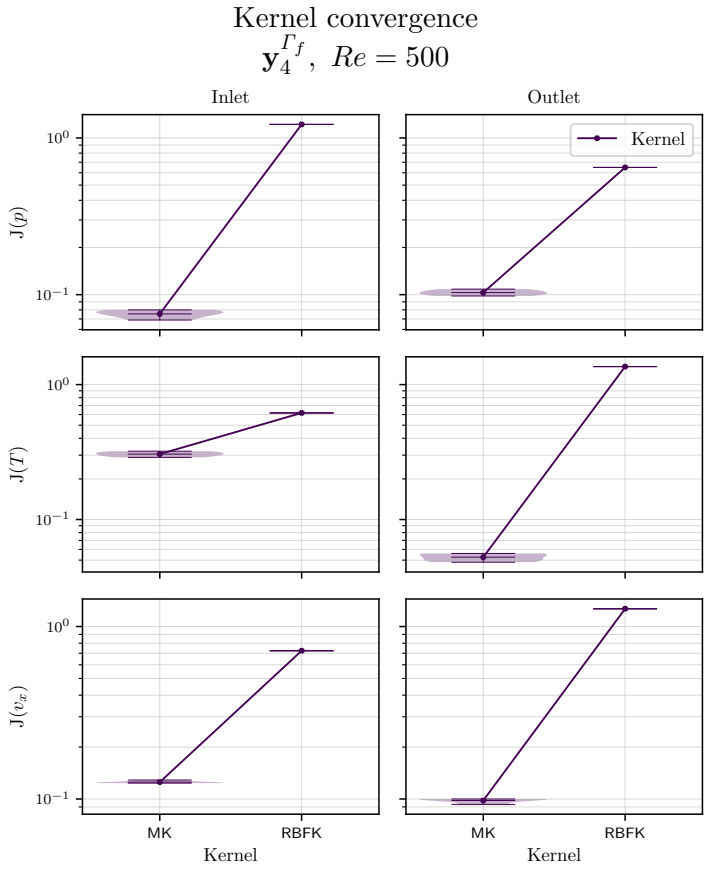


Figure 7.91: S-GP benchmark - J($\mathbf{y}_4^{\Gamma_f}$) against kernels of dataset $\boldsymbol{\omega}_4$ feature GP trained with a validation split of 35%. The Spatial output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

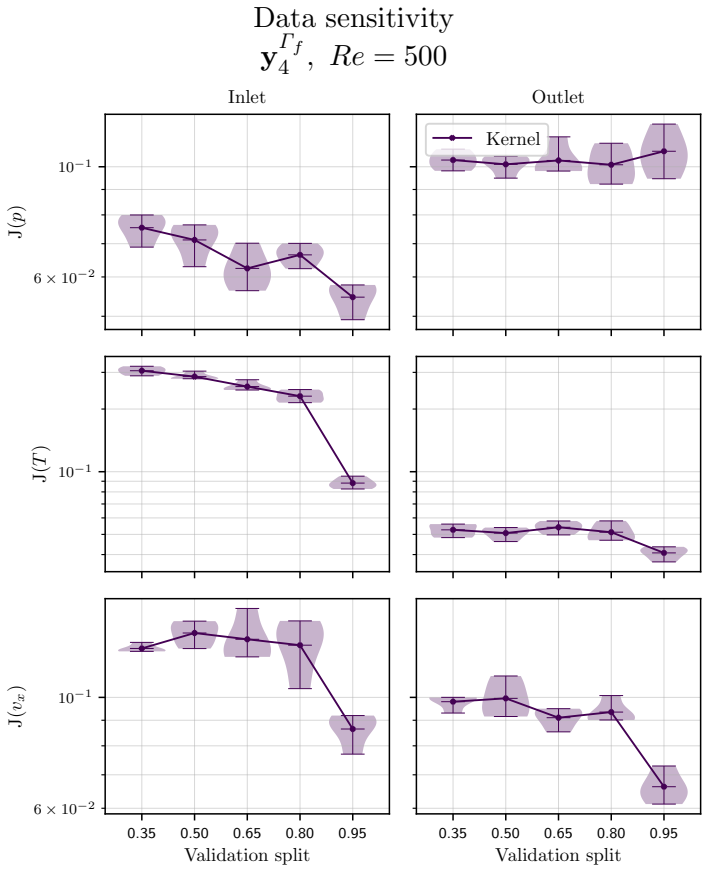


Figure 7.92: S-GP benchmark - J($\mathbf{y}_4^{\Gamma_f}$) against validation split of dataset $\boldsymbol{\omega}_4$ feature GP trained with a Matern kernel. The Spatial output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

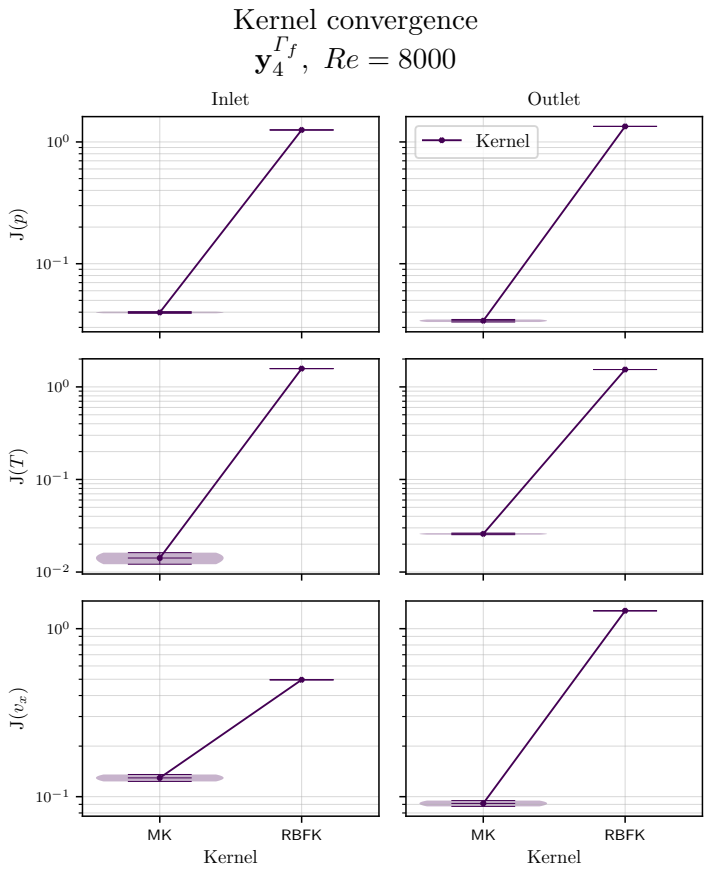


Figure 7.93: S-GP benchmark - J($\mathbf{y}_4^{\Gamma_f}$) against kernels of dataset $\boldsymbol{\omega}_4$ feature GP trained with a validation split of 35%. The Spatial output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

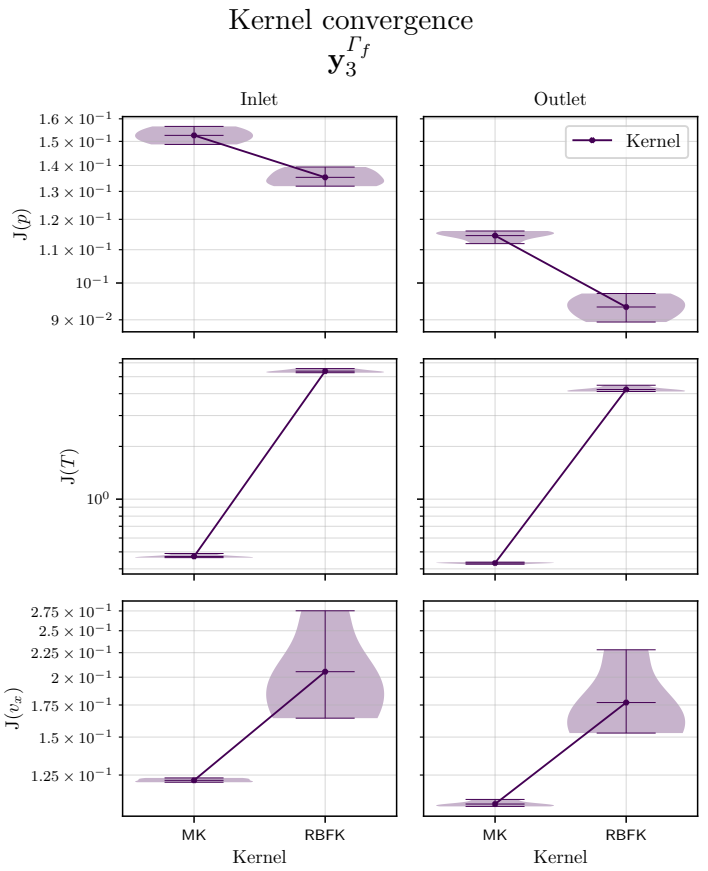


Figure 7.94: S-GP benchmark - J($\mathbf{y}_3^{\Gamma_f}$) against kernels of dataset $\boldsymbol{\omega}_3$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Spatial output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

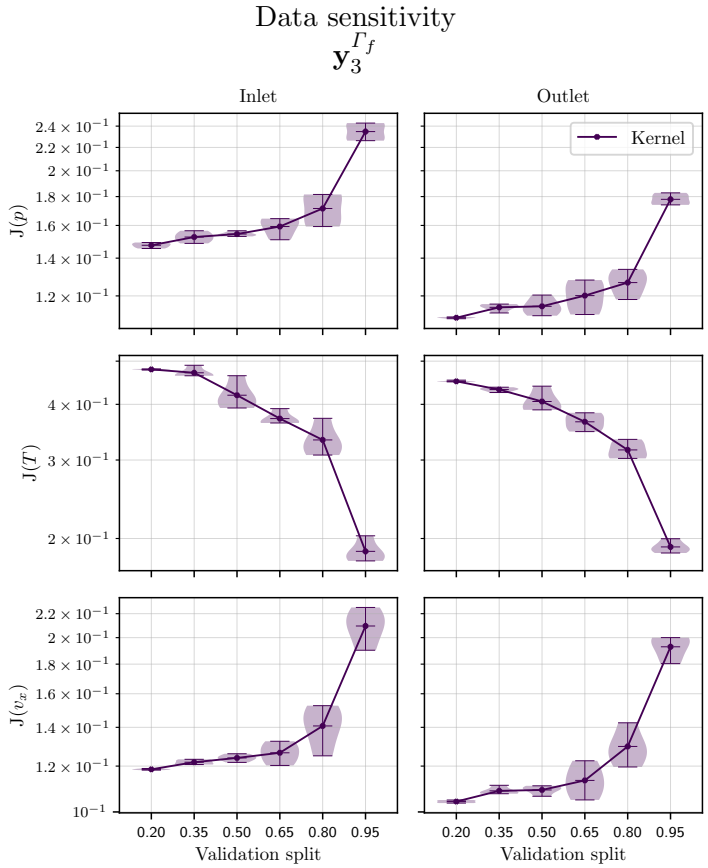


Figure 7.95: S-GP benchmark - $\mathrm{J}(\mathbf{y}_3^{\Gamma_f})$ against validation split of dataset $\boldsymbol{\omega}_3$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a Matern kernel.The Spatial output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

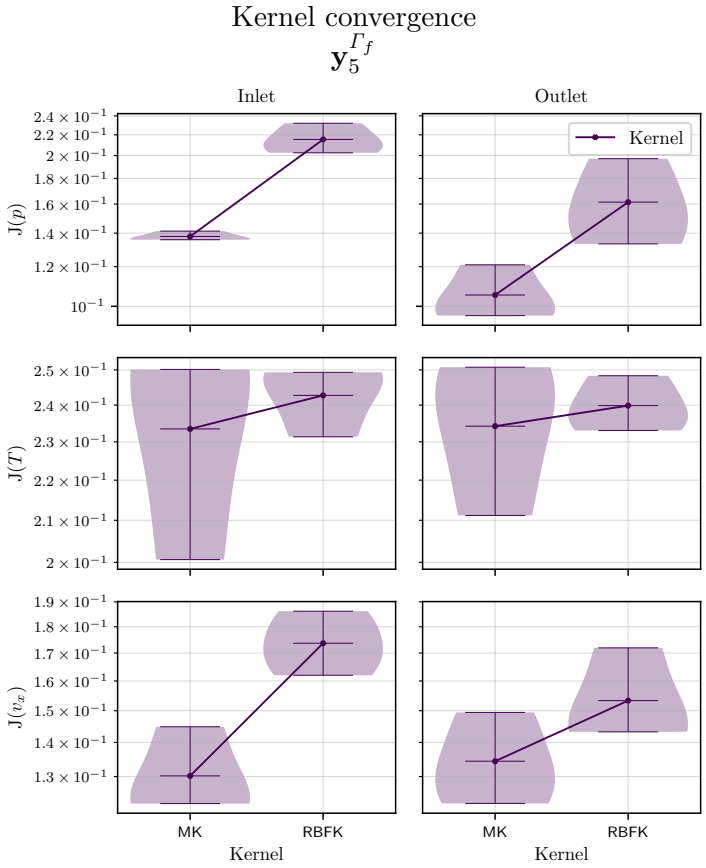


Figure 7.96: S-GP benchmark - $\mathrm{J}(\mathbf{y}_5^{\Gamma_f})$ against kernels of dataset $\boldsymbol{\omega}_5$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 95%. The Spatial output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

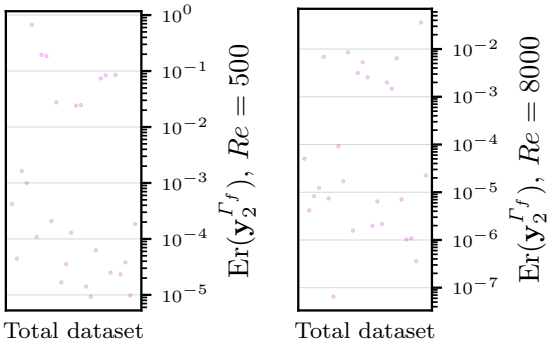


Figure 7.97: S-GP benchmark - Relative error $\mathrm{Er}(\mathbf{y}_2^{\Gamma_f})$ of independent cases prediction from $\boldsymbol{\omega}_2$ feature GP trained with a validation split of 35% vs high-fidelity data.

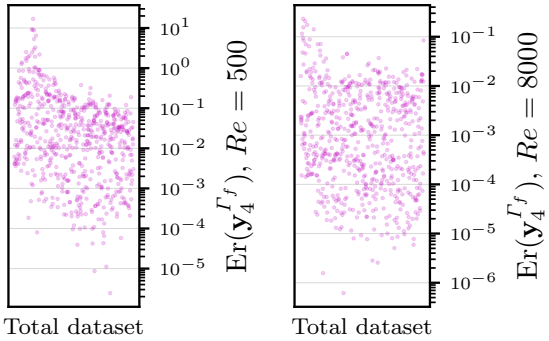


Figure 7.98: S-GP benchmark - Relative error $\mathrm{Er}(\mathbf{y}_4^{\Gamma_f})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature GP trained with a validation split of 35% vs high-fidelity data.

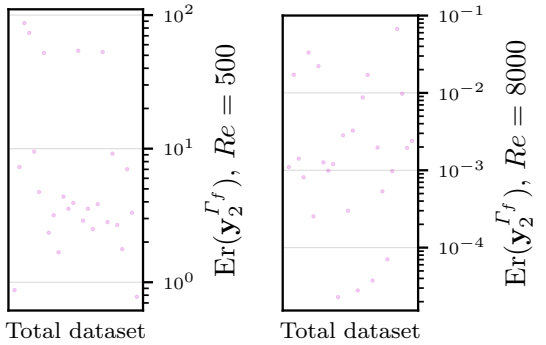


Figure 7.99: S-GP benchmark - Relative error $\mathrm{Er}(\mathbf{y}_3^{\Gamma_f})$ of independent cases prediction from the $\boldsymbol{\omega}_3$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

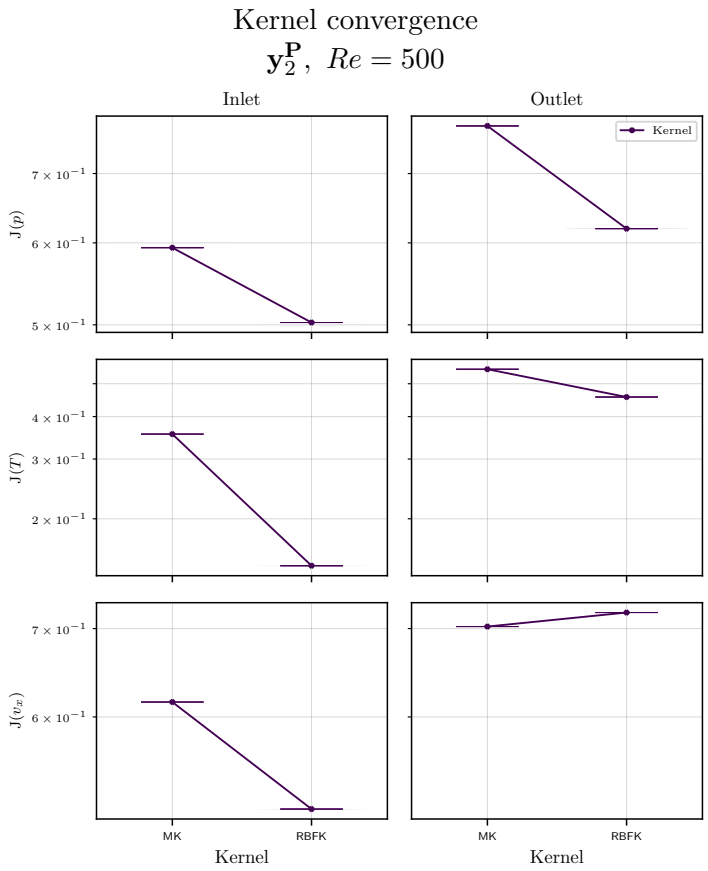


Figure 7.100: M-GP benchmark - J($\mathbf{y}_2^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_2$ feature GP trained with a validation split of 35%. The Modal output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

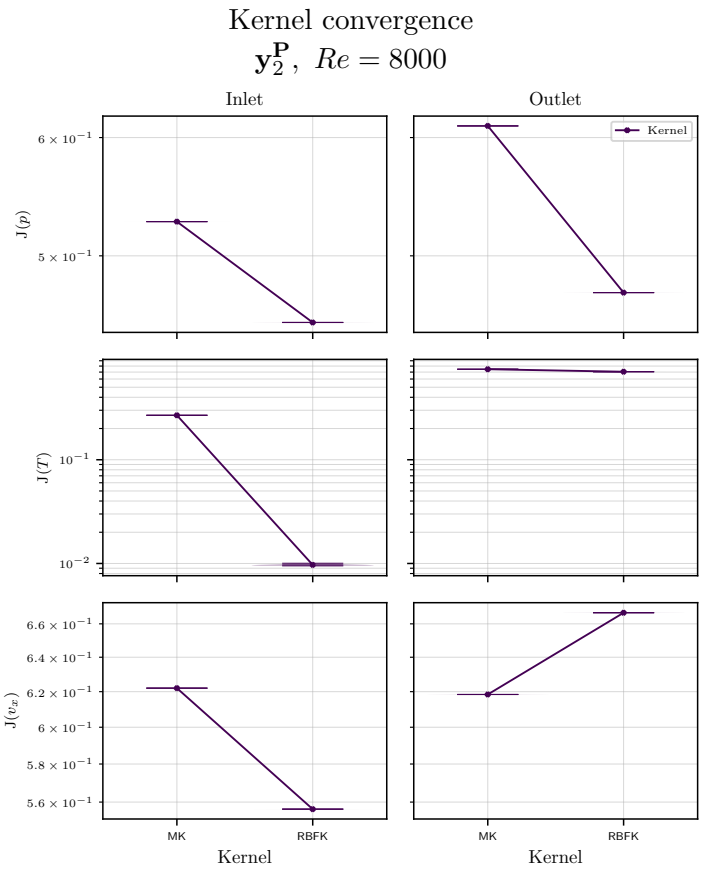


Figure 7.101: M-GP benchmark - J($\mathbf{y}_2^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_2$ feature GP trained with a validation split of 35%. The Modal output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

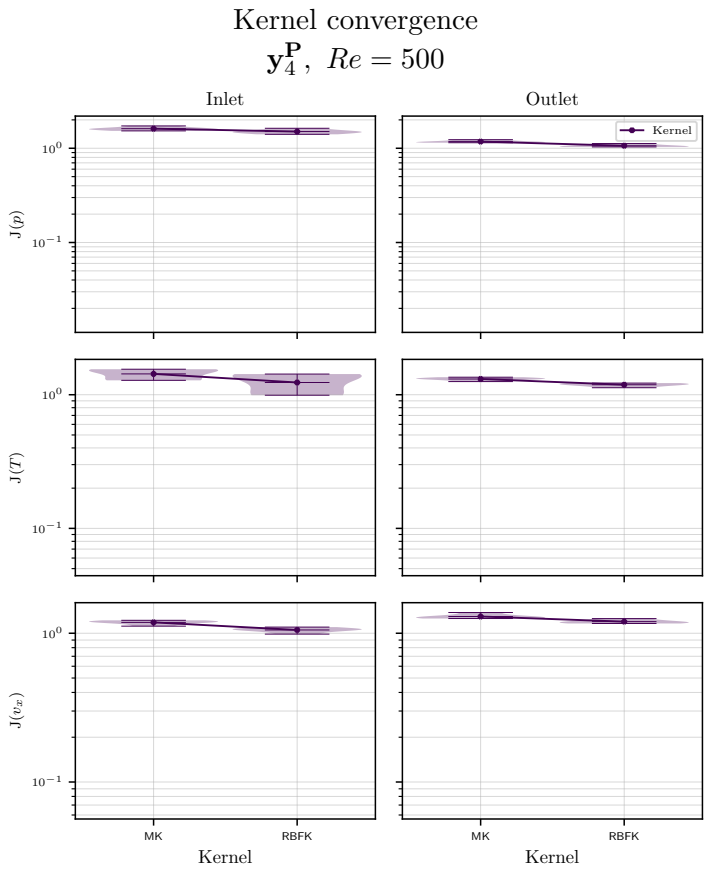


Figure 7.102: M-GP benchmark - J($\mathbf{y}_4^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_4$ feature GP trained with a validation split of 35%. The Modal output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

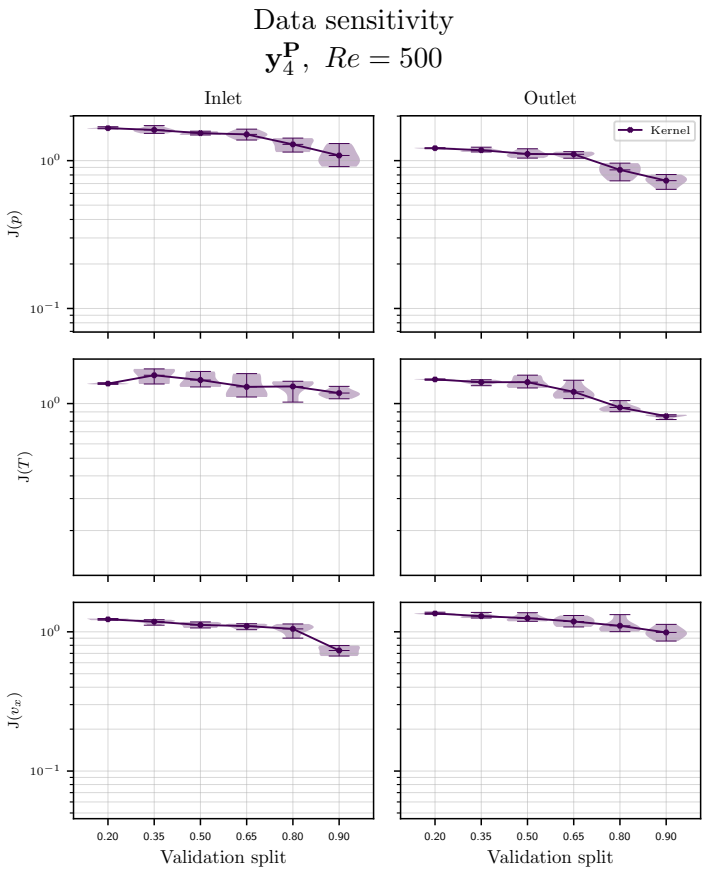


Figure 7.103: M-GP benchmark - J($\mathbf{y}_4^{\mathbf{P}}$) against validation split of dataset $\boldsymbol{\omega}_4$ feature GP trained with a Matern kernel. The Modal output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

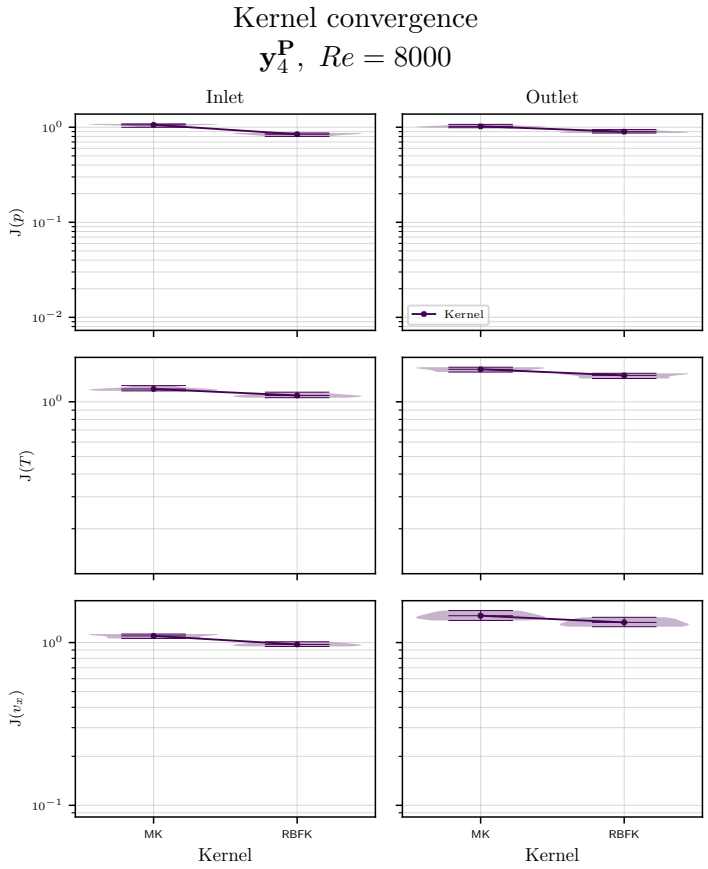


Figure 7.104: M-GP benchmark - J($\mathbf{y}_4^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_4$ feature GP trained with a validation split of 35%. The Modal output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

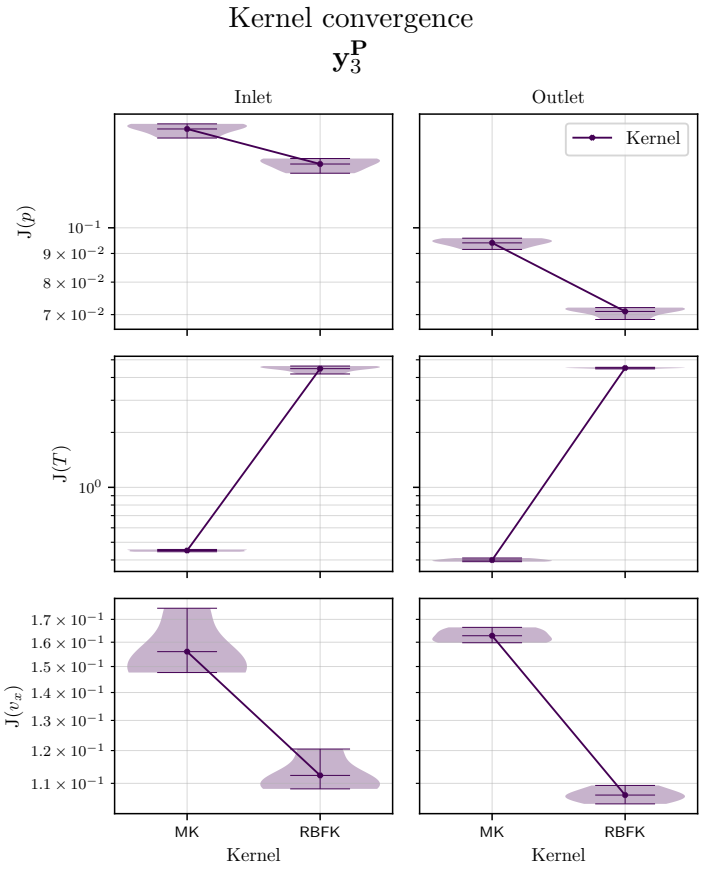


Figure 7.105: M-GP benchmark - J($\mathbf{y}_3^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_3$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Modal output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

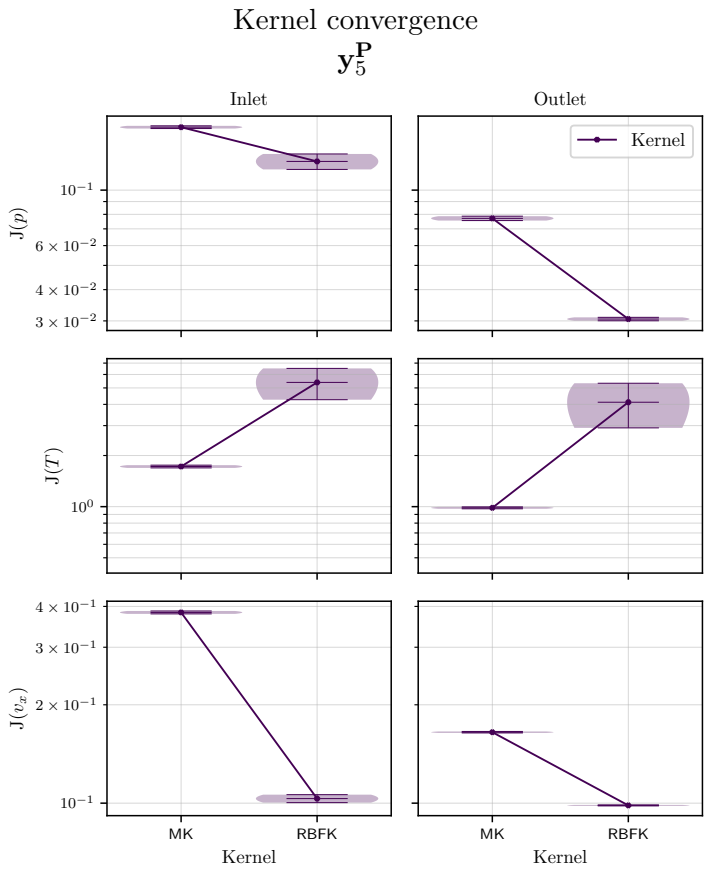


Figure 7.106: M-GP benchmark - J($\mathbf{y}_5^{\mathbf{P}}$) against kernels of dataset $\boldsymbol{\omega}_5$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35%. The Modal output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

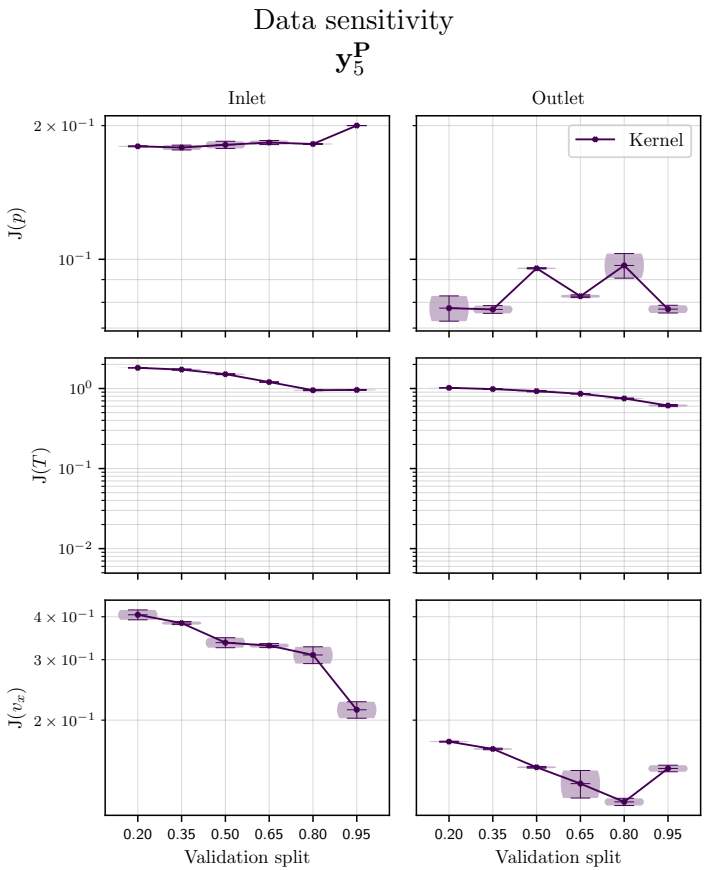


Figure 7.107: M-GP benchmark - J($\mathbf{y}_5^{\mathbf{P}}$) against validation split of dataset $\boldsymbol{\omega}_5$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a Matern kernel. The Modal output data the GP is trained on is based off –left to right, top to bottom– inlet and outlet pressure p, temperature T and velocity v.

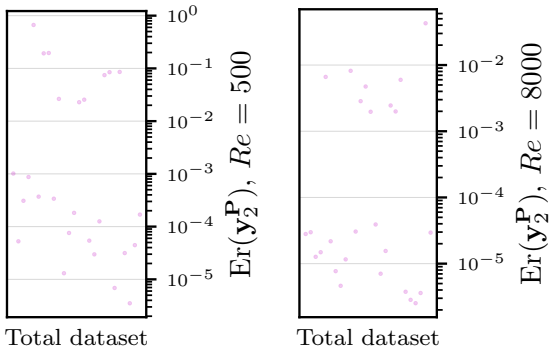


Figure 7.108: M-GP benchmark - Relative error $\mathrm{Er}(\mathbf{y}_2^{\mathbf{P}})$ of independent cases prediction from the $\boldsymbol{\omega}_2$ feature GP trained with a validation split of 35% vs high-fidelity data.

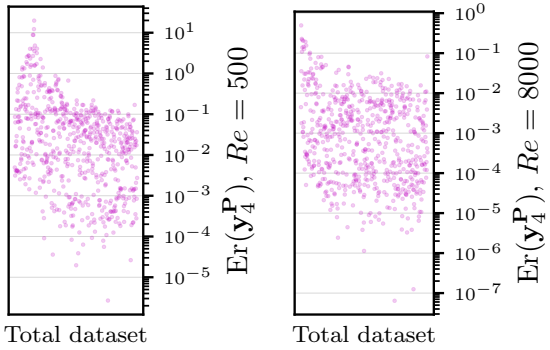


Figure 7.109: M-GP benchmark - Relative error $\mathrm{Er}(\mathbf{y}_4^{\mathbf{P}})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature GP trained with a validation split of 35% vs high-fidelity data.

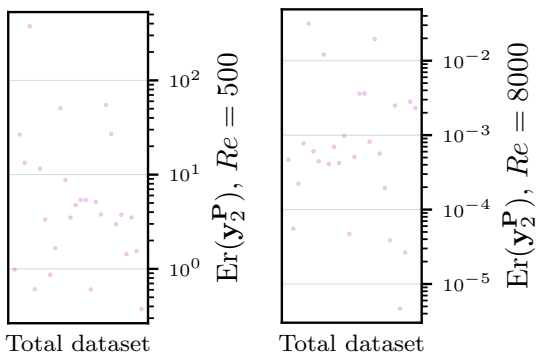


Figure 7.110: M-GP benchmark - Relative error $\mathrm{Er}(\mathbf{y}_3^{\mathbf{P}})$ of independent cases prediction from the $\boldsymbol{\omega}_3$ feature GP trained with $Re = [500, ..., 8000]$ at increments of 500 with a validation split of 35% vs high-fidelity data.

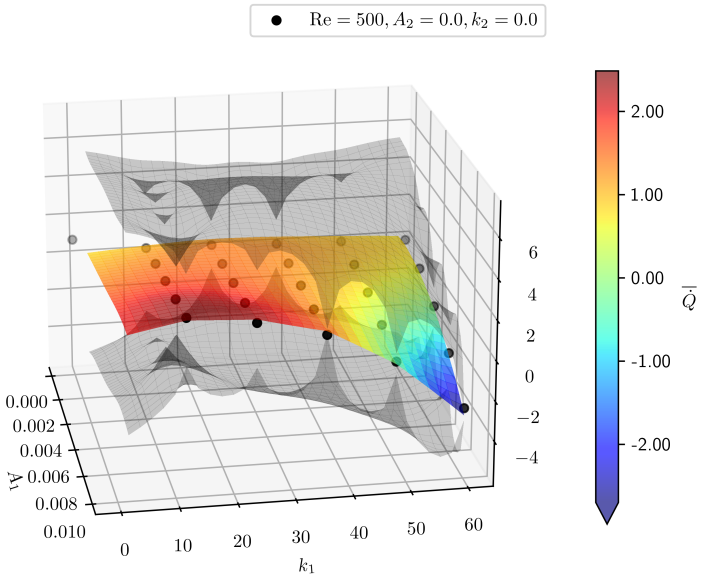


Figure 7.111: M-GP benchmark - $\overline{\dot{Q}} = \dot{Q}_m/\dot{Q}_0$ prediction surface vs high fidelity data points from the $\boldsymbol{\omega}_4$ feature GP trained with a validation split of 35% for $Re = 500$.

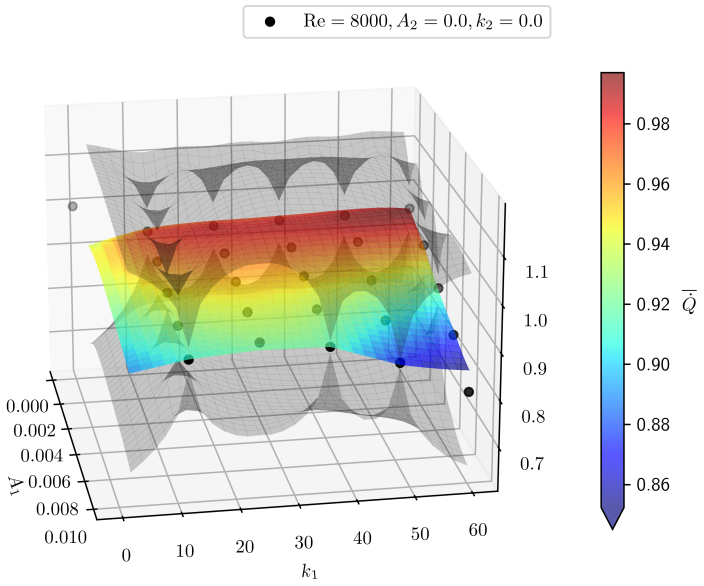


Figure 7.112: M-GP benchmark - $\overline{\dot{Q}} = \dot{Q}_m/\dot{Q}_0$ prediction surface vs high fidelity data points from the $\boldsymbol{\omega}_4$ feature GP trained with a validation split of 35% for $Re = 8000$.

GP Preliminary findings

From the observations in Section 7.3.3, the sensitivity analysis of GP models with increasing complexities in the problem is discussed alongside findings from said analysis.

- **Convergence.** Convergence sensitivity is low, most models reach similar L_J. Slightly sensitive to $\mathbf{x}^{N_x}$, more visible at higher $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$, where surprisingly the highest $\mathbf{x}^{N_x}$ got the lowest log likelihood.

- **Consistency.** Consistency sensitivity is low, most models reach similar L_J regardless of random initialisation.

- **Over-fitting.** Heavily sensitive to flow regime regardless of feature set size $\boldsymbol{\omega}_{N_i}$ and output size $\mathbf{x}^{N_x}$. $Re = 500$ has values further apart from each other $-\overline{\dot{Q}} = [-12, 5]-$ compared to $Re = 8000$ $-\overline{\dot{Q}} = [0, 1]-$, highly non-linear behaviour is assumed[4]. Over-fitting is enhanced by increasing $\boldsymbol{\omega}_{N_i}$ and $\mathbf{x}^{N_x}$, sensitivity of the model to increasing input-output sizes is non-negligible.

- **Data cost and sensitivity.** Due to the observed behaviour in over-fitting, high sensitivity to $\boldsymbol{\omega}_{N_i}$ size and to non-linear behaviour in the function is assumed. However, due to the memory and computational costs of this algorithm, it is not feasible to increase the training data size.

- **Computational cost.** Extremely sensitive to increase in $\boldsymbol{\omega}_{N_i}$. At $\boldsymbol{\omega}_5$ the memory expense of the covariance matrix leads to computations becoming unfeasible for large training datasets. The memory expense is enhanced by the increase in $\mathbf{x}^{N_x}$ due to the points being treated as independent outputs, where the gradient of L_J –which includes the covariance matrix– is computed for each output and stored in memory to update the weight coefficients.

Overall, the main advantage of using a GP model is the computation of probability distribution around the prediction mean values. Some observed advantages include the consistency of the model across all parameters, seemingly converging to similar values regardless of random initialisation, dimensions, kernel and training dataset. However, the memory cost and computational complexity of large $\boldsymbol{\omega}_{N_i}$ and increasing training dataset size becomes unfeasible quite fast. Input features $\boldsymbol{\omega}_5$ made the computations unfeasible, but the models with $\boldsymbol{\omega}_4$ have enough accuracy to approach the flow optimisation problem. The low-rank output $\mathbf{x}^{\mathbf{P}}$ with $\boldsymbol{\omega}_4$ has a performance closer to $\mathbf{x}^{\dot{Q}}$ while maintaining the flow variable information from $\mathbf{x}^{\Gamma_f}$, which makes it a more feasible approach to observe flow behaviour in more detail. This assessment satisfies Objectives VII.IV and VII.V.
Going back to Figures 7.111 and 7.112, the decrease in confidence when the model predictions approach low amplitude and wavenumbers is due to the *baseline* data point –seen as the right far side point in the Figures– being part of the training set. The model is fitting the function around this point with no information close to it, which makes it fall outside of the boundaries of

[4]The values are normalised to avoid the influence of magnitude differences in the performance of the models, but the non-linear behaviour remains part of the dataset.

the parametric space. With this information we can re-train the models excluding the baseline point and analyse the performance difference, or increase the high fidelity data availability closer to that region. This insight is not straight forward with RBF or NN, which gives GP an advantage for re-training purposes.

7.3.4 Discussion and findings

The description of output dimensions $\mathbf{x}^{\mathrm{N}_x}$ and input sets $\boldsymbol{\omega}_{\mathrm{N}_i}$ used in the training of surrogate models aligns with Objective VII.III. To recap the general steps of surrogate model training with the ML algorithms evaluated in this study, the method used in NN and GP is

1. Randomly select values for the unknown weight coefficients $\mathbf{W}_{\mathrm{N}_i}$,
2. Compute the prediction $\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}(\boldsymbol{\omega}_{\mathrm{N}_i}, \mathbf{W}_{\mathrm{N}_i})$ with the corresponding architecture,
3. Compare the prediction $\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}$ with the training data output $\mathbf{D}_i^{\mathrm{N}_x}$ by computing the loss J if NN, and the log likelihood $\mathrm{L_J}$ if GP,
4. Compute the gradient estimate of J or $\mathrm{L_J}$ in terms of the weight coefficients $\mathbf{W}_{\mathrm{N}_i}$ as $\frac{\partial \mathrm{J}}{\partial \mathbf{W}_{\mathrm{N}_i}}$, $\frac{\partial \mathrm{L_J}}{\partial \mathbf{W}_{\mathrm{N}_i}}$,
5. Update the weight coefficients $\mathbf{W}_{\mathrm{N}_i}$ from the gradient estimate,
6. Repeat from step 2. until target loss has been achieved, or the current iteration reaches the maximum *epoch*.

The method used in RBF interpolation has a simple linear system of equations, and the only training step involves solving the linear system for $\mathbf{W}_{\mathrm{N}_i}$ when $\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x} = \mathbf{D}_i^{\mathrm{N}_x}$. The observations and findings regarding the performance of RBF, NN and GP models presented in Sections 7.3.1, 7.3.2 and 7.3.3, were identified through a sensitivity analysis as complexities in the problem increase. The final conclusion of the algorithms' assessment concerns the comparison of their characteristics

- **Convergence.** RBF and NN models' loss J convergence depends primarily on the architecture and increasing input dimension $\boldsymbol{\omega}_{\mathrm{N}_i}$, especially at $\boldsymbol{\omega}_5$ where the problem has high non-linearities due to the low correlation between shape and flow parameters. RBF has the lowest loss overall with a Linear kernel at $\mathrm{J}(\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}) \approx 10^{-30}$, while the best overall architecture for the NN models is 3 layers and 128 neurons, which with most models reached $\mathrm{J}(\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}) \approx 10^{-6}$. GP has a different performance index from the log likelihood $\mathrm{L_J}$, and the convergence has some sensitivity to N_x, where surprisingly the highest output dimension $\mathbf{x}^{\Gamma_f}$ has the overall log likelihood $\mathrm{J}(\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}) \approx 10^{-1}$.
- **Consistency.** NN yielded the models with least consistency, where the architectures reached different J values when increasing the non-linearities of the problem through $\boldsymbol{\omega}_{\mathrm{N}_i}$. The sensitivity is most visible with outputs $\mathbf{x}^{\Gamma_f}$ and $\mathbf{x}^{\dot{Q}}$, and the low-rank $\mathbf{x}^{\mathbf{P}}$ had consistent results. This is unexpected, since this output has a size in between $\mathbf{x}^{\Gamma_f}$ and $\mathbf{x}^{\dot{Q}}$, where the best behaviour was expected from the lowest size $\mathbf{x}^{\dot{Q}}$. This leads us to believe

that function created from the POD modes might have less non-linear behaviour than the THP evaluation. RBF and GP were consistent across all parameters, which means these algorithms do not require to test multiple random initialisations of the weight coefficients to yield the best model with the same kernel.

- **Over-fitting.** Models from RBF and GP are heavily over-fitting at $Re = 500$ due to the extreme changes observed at laminar regimes, described in Section 6.3. The increased non-linear behaviour of the function is better portrayed by NN models at $\boldsymbol{\omega}_4$, but the low correlation of flow and shape parameters at $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$ is more difficult to qualify with the architectures assessed in this study. It is concluded that $\boldsymbol{\omega}_5$ requires training with increasing epoch and potentially deeper networks to approach the non-linearities of these input sets, but NN is still the best candidate to do so.

- **Data cost and sensitivity.** It is generally not recommended to use the 100% of the available data to train surrogate models to avoid over-fitting. This is to retain data unseen by the models as verification of the robustness for prediction outside of their own database. However, when the flow parameter is considered in the input $\boldsymbol{\omega}_{\mathrm{N}_i}$ at $i = \{3, 5\}$, all models struggle to predict properly the lower Re regimes with a training set of 64% the available data. The data cost of including the Re as a feature is larger than the gain from using surrogate modelling, at least for RBF and GP, specially since GP's computational complexity grows with increasing input features as well. The use of surrogate modelling to reduce the computational cost of high fidelity models is hindered when these non-linearities and low correlation between features require a larger high fidelity dataset. An approach to this situation could be using dimensionality reduction in the input sets as well. Nevertheless, as stated in the previous characteristic comparison, NN models can be trained with larger epochs or deeper architectures to improve their capability at portraying such non-linearities with smaller datasets.

- **Computational cost.** RBF is by several orders of magnitude the cheapest model across all parameters evaluated in this study. NN models increase the amount of calculations with deeper architectures or with more input and output dimensions, explaining the higher computational cost at increasing values of the parameter combination. However, critical architectures can find a set target loss in early epoch stages, stopping the process and yielding lower CPU times, observed in the mid range neuron count. GP models are memory expensive with large datasets, with a cubic computational complexity on account of the correlation matrix of $\boldsymbol{\omega}_{\mathrm{N}_i}$. Recall that at higher $\boldsymbol{\omega}_{\mathrm{N}_i}$ with the same percentage, the amount of data available is larger, since the complete dataset for $\boldsymbol{\omega}_5$ has about 10 000 cases, and the datasets for lower features are taken from the available 10 000 cases. Training and prediction in a GP algorithm require the training set to do the calculations, which is unfeasible for $\boldsymbol{\omega}_5$, especially with increasing output sizes $\mathbf{x}^{\mathrm{N}_x}$. This is more related to the memory cost, since multi-output models compute the gradient of $\mathrm{L_J}$ –which includes the covariance matrix– for each output and store them in memory to update the weight coefficients. The regression format used in this study, `GPyTorch`'s batch independent model, does not learn the similarities between output points, reducing the computational cost of training. However, the memory expense depends on the covariance matrix alone,

which can only be reduced by lowering the size of $\boldsymbol{\omega}_{\mathrm{N}_i}$ or the training dataset.

From this analysis, we conclude that neither of the algorithms were suitable to approach the THP optimisation problem with input features including Re as a parameter, $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$, at least with the assessed architectures. However, each algorithm has a characteristic that makes it reliable for the problem with input features $\boldsymbol{\omega}_4$.
The cheapest in computational cost with good accuracy in more linear functions –turbulent regimes where $\overline{\dot{Q}} = [0, 1]$– is RBF with a Linear kernel. To address the non-linear behaviour of THP optimisation in laminar regimes –$\overline{\dot{Q}} = [-12, 8]$–, NN has the required accuracy. In NN models, the modal output $\mathbf{x}^{\mathbf{P}}$, comprised of the prominent POD modes, has CPU times close to the smallest output $\mathbf{x}^{\dot{Q}}$, but provides the flow variable information from $\mathbf{x}^{\Gamma_f}$. Furthermore, NN has the potential to approach the problem with input features $\boldsymbol{\omega}_5$ by building deeper architectures with larger epoch, although the current best architecture –3 layers, 128 neurons– could yield better results if trained for longer as well.
The GP models have a very *niche* characteristic, which is helpful for the three algorithms. The region of confidence gives insight into the function behaviour, showing areas of interest where more data might be required or, on the contrary, could be leading to ill-posedness, poisoning the models' robustness. For future work, and with the good consistency of the algorithm, a confidence region can be computed by training a single GP model from a small high fidelity dataset –no need for over 10 000 cases, lowering the computational expense of the database population process–. The function insight gained from this information can be used to selectively feed the available training data into NN, RBF and even GP to reduce ill-posedness and over-fitting, as well as running new simulations to fill the gaps in the parametric areas that need refinement. Overall, the performance assessment of ML algorithms with varying input and output sizes satisfies Objectives VII.IV and VII.V.

7.4 Surrogate model prediction for shape optimisation

Let us explore lower order models to approximate the problem of THP evaluation for pipe flows within laminar and turbulent conditions when in contact with parametrised geometries in the wetted surface. Let us explore the data-driven multi-objective optimisation solution of THP in pipe flows within laminar and turbulent conditions when in contact with parametrised geometries in the wetted surface. Three data-driven surrogate models are proposed to approximate the optimisation problem from the numerical solution examples in Section 6.3, and are based on the algorithms presented in Chapter 4. These models are the results of ML assessment from Section 7.3. The objectives of this example are:

Obj VII.VI. Demonstrate the capacity of the surrogate models presented in Section 7.3 to predict the influence of geometry variation on the thermo-hydraulic behaviour of the flow based on the high fidelity model domains.

Obj VII.VII. Explore the advantages of using surrogate models to approach the shape parameter optimisation problem in terms of obtaining real-time solutions via interpolation.

Model	$\boldsymbol{\omega}_4$
M-RBF	Linear kernel
M-NN	Layers=3, Neurons=128
M-GP	Matern kernel

Table 7.6: THP optimisation benchmark of the surrogate models - Selected architecture for each model $\mathbf{y}_4^5(\boldsymbol{\omega}_4)$ set.

Parameter	**Values**	**Parameter**	**Values**
Re	{500, 1000, 1500, 7000, 7500, 8000}	$\mathrm{N_h}$	2
A_i	[0, 0.01] m	k_i	[0, 60] m^{-1}

Table 7.7: Surrogate model's benchmark - Range of values of the input features $\boldsymbol{\omega}$ used to find the set that maximises $\mathbf{y}_{\mathrm{N}_i}^{\mathrm{N}_x}(\boldsymbol{\omega}_{\mathrm{N}_i})$.

Obj VII.VIII. Expose the shortcomings of using surrogate models to approach the shape parameter optimisation problem in terms of accuracy and *batch learning* time expense, as well as the limitations in terms of interpolation vs extrapolation.

From the examples in Chapter 6, the discretisation of a domain defined by a two-dimensional cross section of a pipe, with solid Ω_s in contact with fluid flow Ω_f on one side $-\Omega = \Omega_f \bigcup \Omega_s-$ is considered symmetric at the *pipe-like* geometry stream-wise centre line. To study the influence of micro-structure geometries in interaction with the flow on the overall THP behaviour, the surface in contact with the fluid Γ_{sf} is parametrised using a simplified harmonic function representing the shark skin inspired biomimetic structure described in Section 2.5.2 with H, A_i and k_i as the number of harmonics to implement, and the amplitude and wavenumber of the harmonic, respectively. The system-scale THP is defined through the dissipation rate and advected heat flux from (2.52) and (2.54), respectively, in Section 2.4. The observed results are portrayed as the normalised net thermal power output $\overline{\dot{Q}}$ as defined in (6.6).
The *parametric problem* applied to the THP optimisation is taken from the examples in Section 7.2. This is comprised of a set of *snapshots* $\mathrm{N_s}$, referring to the set of cases available from the high fidelity data. Since these were computed with varying surface roughness and flow regimes, each case is dependent on a set of input features $\boldsymbol{\omega}_{\mathrm{N}_i}$ with $_{\mathrm{N}_i}$ parameters based on shape, amplitude A_i and wavenumber k_i, and flow parameter Re. The variables of interest from each case are considered in the set of outputs $\mathbf{D}(\boldsymbol{\omega})$, which encompasses the numerical measurements at locations $\mathbf{x} \in \mathbb{R}^{\mathrm{N}_x}$. The data is arranged such that the element D_i^j is the measurement in the $j-$th location taken with the $i-$th iteration of features $\boldsymbol{\omega}_{\mathrm{N}_i}$, where the features are considered the independent variable.
Based on the ML assessment from Section 7.3, three models were selected for the ML-aided THP optimisation study, presented in Table 7.6. The ML performance analysis presented $\boldsymbol{\omega}_4$ and $\mathbf{x}^{\mathbf{P}}$ to be the model dimensions best fitted to approach this problem due to the surrogate

models' accuracy with this input feature set and the low computational cost of the output dimension for the prediction of flow variables. The output size is set as $\mathrm{N}_x = 5$, a value that represents the most prominent modes of the dataset for $\boldsymbol{\omega}_4$, obtained from the eigen-analysis presented in Section 7.2. The datasets $\boldsymbol{\omega}_4$ and $\mathbf{x}^5$ have the form

$$\boldsymbol{\omega}_4 = [\mathrm{A}_1, \mathrm{k}_1, \mathrm{A}_1, \mathrm{k}_2]\,, \qquad \text{when } Re = constant,\ \mathrm{N_h} = 2\,, \tag{7.5a}$$

$$\mathbf{x}^5 = \left[\mathbf{P}^{in}_{v_x}, \mathbf{P}^{in}_{T}, \mathbf{P}^{in}_{p}, \mathbf{P}^{out}_{v_x}, \mathbf{P}^{out}_{T}, \mathbf{P}^{out}_{p}\right], \text{ then } \mathrm{N}_x = 5\,, \tag{7.5b}$$

where $\mathbf{P}$ are the POD modes for each flow variable, and the superscripts *in* and *out* refer to the variable profiles taken at the inlet and outlet, respectively. Recall that the flow variables are treated independently, each variable is represented by an output $\mathbf{x}^5$ and has a separate surrogate model. The surrogate models approximate the dataset $\mathbf{D}^5_4(\boldsymbol{\omega}_4)$ in the form of a function $\mathbf{y}^5_4(\boldsymbol{\omega}_4)$, and `SciPy optimize.differential_evolution`, described in Appendix D.1, is used to obtain the parameters with maximum $\dot{Q}_m$. The features $\boldsymbol{\omega}_{\mathrm{N}_i}$ boundary values used to limit the search of a maximum $\dot{Q}_m$ are listed in Table 7.7. These are listed in Table 7.8, alongside the model and Re set they correspond to and $\dot{Q}_0$ predicted by the same model. The obtained shape parameters are then used to compute high fidelity models, from which the $\overline{\dot{Q}}$ is presented in Table 7.8 against the predicted $\overline{\dot{Q}}$.

All models are in accordance with the high fidelity data about the flow regimes and general parameter region the maximum $\dot{Q}_m$ is found in. Results within the flow region Re = [2000, 6500] yielded no optimisation over the baseline and the flow regimes where optimisation was found are close to the boundaries of the parameter space, as observed in the high fidelity study from Section 6.3. In turbulent regimes the models found the optimised geometry out of the amplitude bounds from the parametric region in the high fidelity data –studied region is $\mathrm{A}_i = [0.002,\ 0.01]$ m with the baseline at $\mathrm{A}_i = 0.0$m, the geometries found are under $\mathrm{A}_i = 0.002$m–. Interestingly, MGP found a rough surface with $\dot{Q}_m$ above the baseline at $Re = 7500$, but this geometry did not result in optimisation with high fidelity modelling. Neither of the model predictions at $Re = 7000$ resulted in optimisation when verified with high fidelity modelling, however, the high fidelity data at $Re = 8000$ gave better results than the predicted by MNN, going over the optimisation found in Section 6.3 within this Re value. Similarly, MNN and MGP found geometries that had optimised behaviour in the high fidelity data at 1000 and $Re = 1500$, respectively, compared to the results found in Section 6.3. Overall, the models seem to underestimate in laminar regimes and overestimate in turbulent regimes, but MGP specially struggles predicting $\dot{Q}_0$ and $\dot{Q}_m$ accurately.

With the modal output, it is possible to reconstruct the inlet and outlet flow variables related to both $\dot{Q}_0$ and $\dot{Q}_m$, presented in Figures 7.113 to 7.122 for MRBF, 7.123 to 7.132 for MNN and 7.133 to 7.144 for MGP, alongside the corresponding high fidelity data. Note that while MRBF and MNN are properly portraying the flow variables of the baseline, MGP fails to do so for pressure p and turbulent temperature T, although the overall laminar values and inlet turbulent values are similar to the high fidelity data regardless of the errors. For $Re = 1500$ and higher, MRBF is overestimating the temperature gradient from the wall, an increase in the velocity values just above the boundary layer and the pressure drop reduction, a behaviour seen in MNN in turbulent regimes. MNN at $Re = 8000$ is overestimating the roughness impact on the boundary layer, increasing the thickness –reducing the shear stress–, but is overestimating pressure drop. In this case, the boundary layer in the high fidelity results

Re	**Model**	A_1 [m]	A_2 [m]	k_1 [m^{-1}]	k_2 [m^{-1}]	$\dot{Q}_0$ [W/m^2]	Optimum $\overline{\dot{Q}}$	Real $\overline{\dot{Q}}$
500	M-RBF	$1e^{-2}$	$1e^{-2}$	12	12	$3.2e^5$	5.0655	5.2145
	M-NN	$1e^{-2}$	$1e^{-2}$	17	13	$3.2e^5$	5.3159	4.621
	M-GP	$1e^{-2}$	$1e^{-2}$	12	12	$2.0e^5$	8.1242	5.2145
1000	M-RBF	$9.9e^{-3}$	$9.6e^{-3}$	7	29	$1.4e^6$	4.7659	5.654
	M-NN	$9.9e^{-3}$	$9.9e^{-3}$	14	7	$1.3e^6$	4.8952	5.818
	M-GP	$9.7e^{-3}$	$9.6e^{-3}$	13	33	$2.8e^6$	2.3089	5.1973
1500	M-RBF	$8.4e^{-3}$	0.0	36	0	$2.2e^6$	7.4833	3.3321
	M-NN	$9.2e^{-3}$	$1e^{-2}$	39	20	$2.3e^6$	7.6703	7.5335
	M-GP	$9.3e^{-3}$	$9.5e^{-3}$	33	15	$8.3e^6$	2.0483	8.0096
7000	M-RBF	$0.1e^{-3}$	$0.1e^{-3}$	47	8	$3.6e^8$	1.0033	0.9978
	M-NN	$0.4e^{-3}$	$0.1e^{-3}$	52	58	$3.6e^8$	1.0082	0.9974
	M-GP	$1.9e^{-3}$	$0.7e^{-3}$	51	3	$3.3e^8$	1.0799	0.9858
7500	M-GP	$2e^{-3}$	$2.1e^{-3}$	50	49	$3.5e^8$	1.0739	0.9704
8000	M-RBF	$0.1e^{-3}$	$0.5e^{-3}$	45	33	$4.1e^8$	1.0051	0.9967
	M-NN	$1e^{-3}$	$0.3e^{-3}$	45	54	$4.1e^8$	1.0023	1.0052
	M-GP	$1.9e^{-3}$	$2.1e^{-3}$	46	54	$3.7e^8$	1.0682	0.9765

Table 7.8: THP optimisation benchmark of the surrogate models - Net power output $\dot{Q}_0$ of baseline surface in different Re regimes and shape parameters of the rough surface with optimum $\overline{\dot{Q}}$ in the corresponding flow conditions.

seems to have negligible changes, but the velocity values above the layer increase similarly to MNN's predictions, although in a smaller scale. MGP on the other hand, overestimates the velocity values just above the boundary layer while portraying no impact on the layer thickness, also predicting heavily underestimated $\overline{f}_h$ and overestimated $\overline{f}_f$. However, the pressure and temperature values from the rough surface are in accordance with the high fidelity data, and the inaccuracies comes from the poorly predicted baseline variable profiles. For laminar regimes, MNN is closer to the high fidelity profiles besides the temperature values at $Re = 500$ and the pressure profile at $Re = 1500$, while MRBF has good accuracy for all besides $Re = 1500$. MNN and MGP at $Re = 1000$ are underestimating the velocity development from a parabolic to a logarithmic profile –very subtle change–.
Overall, the three models, MRBF, MNN and MGP, found geometries at $Re = 1000$ with optimised $\dot{Q}_m$ compared to any from Section 6.3 where the best was given by MNN, while MNN and MGP yielded geometries for $Re = 8000$ and $Re = 1500$, respectively, all verified with high fidelity modelling.

7.4.1 Radial Basis Function interpolation

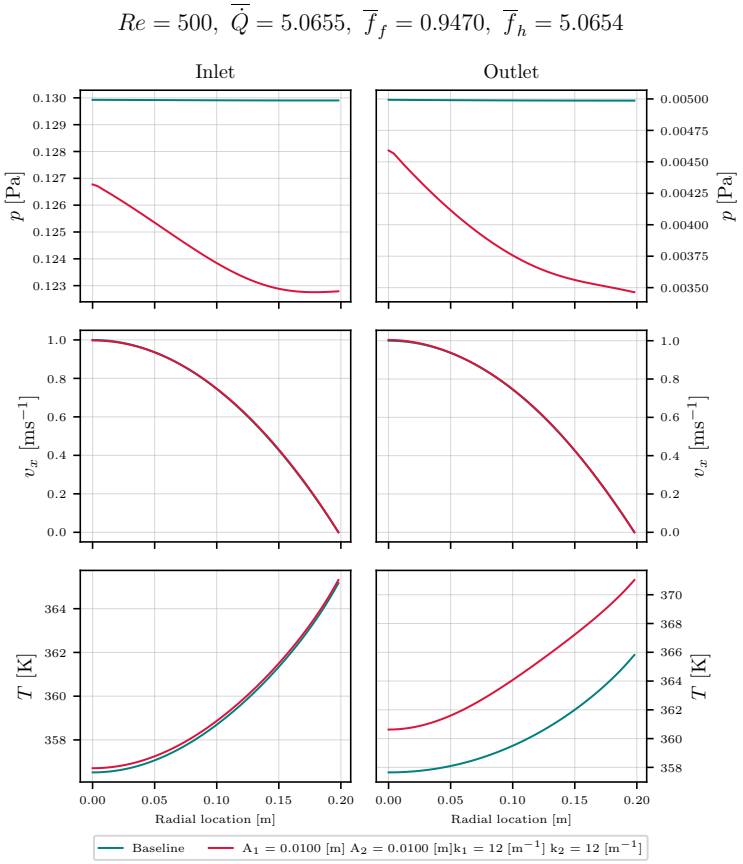


Figure 7.113: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface plate with highest $\overline{\dot{Q}}$ predicted by the M-RBF surrogate model at $Re = 500$.

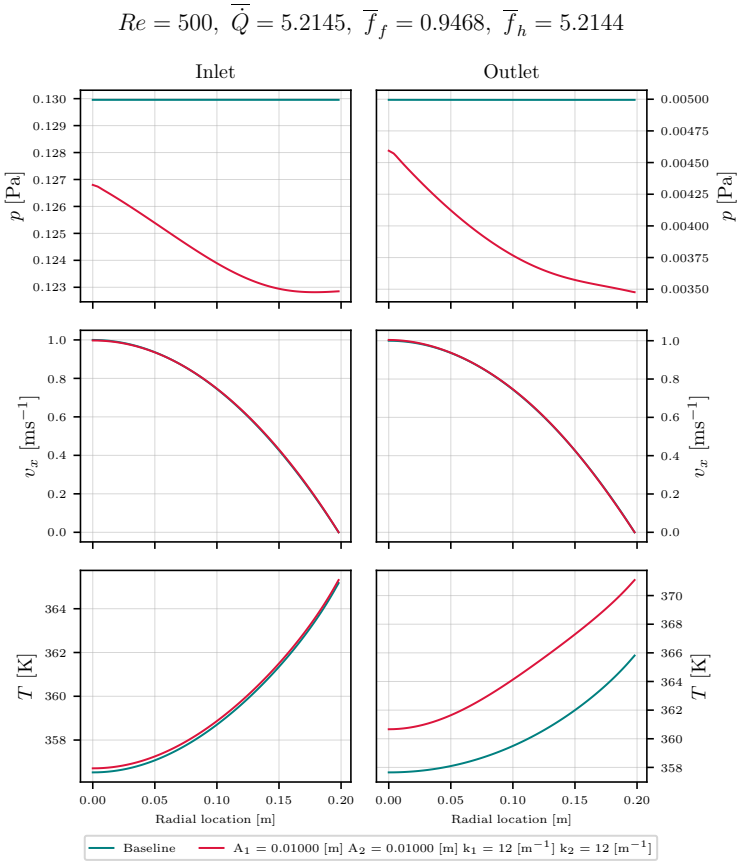


Figure 7.114: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 500$.

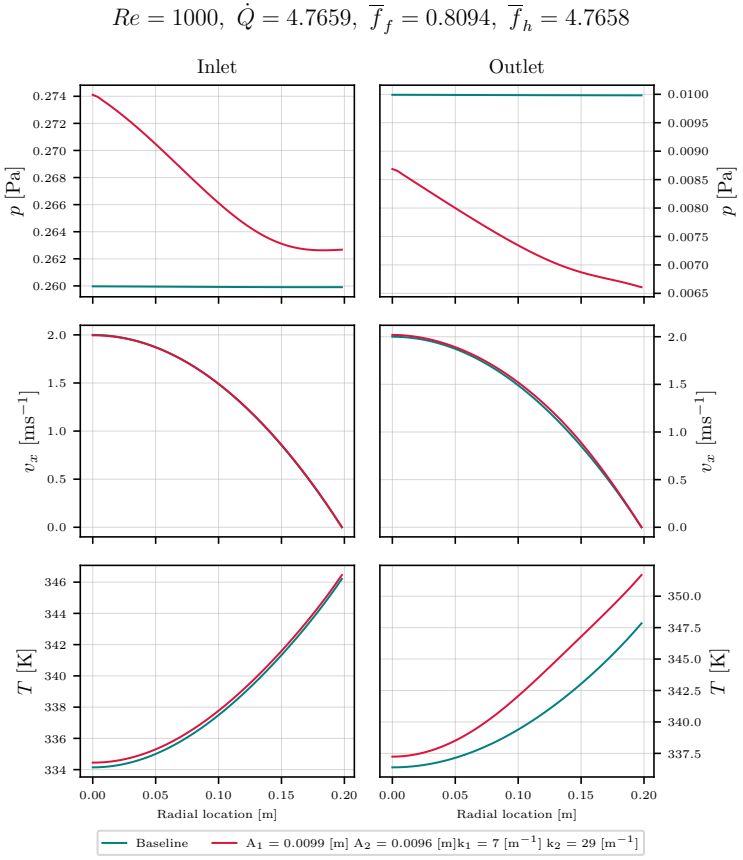


Figure 7.115: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface plate with highest $\overline{\dot{Q}}$ predicted by the M-RBF surrogate model at $Re = 1000$.

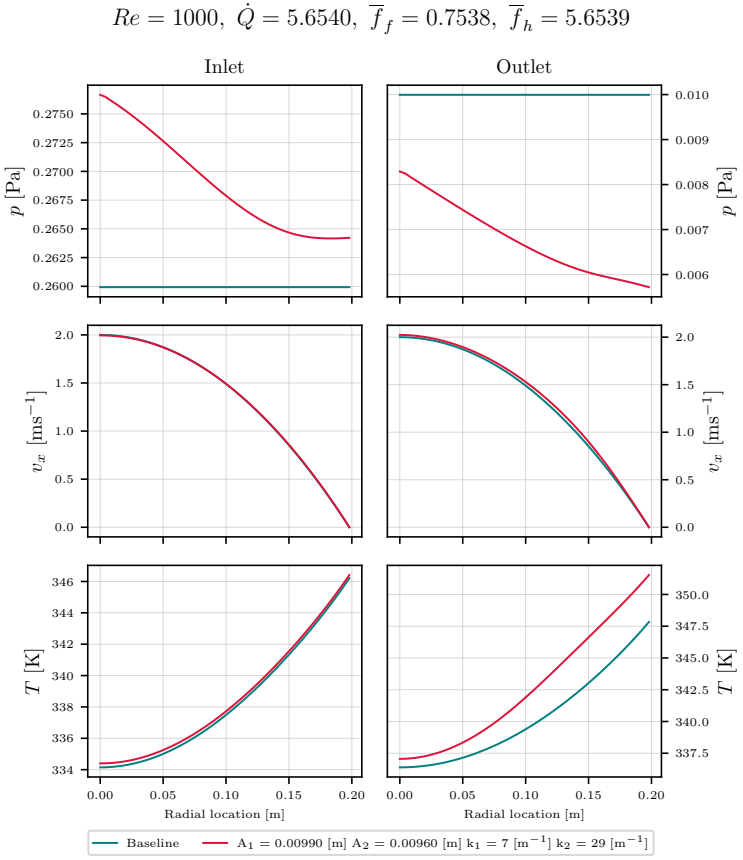


Figure 7.116: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 1000$.

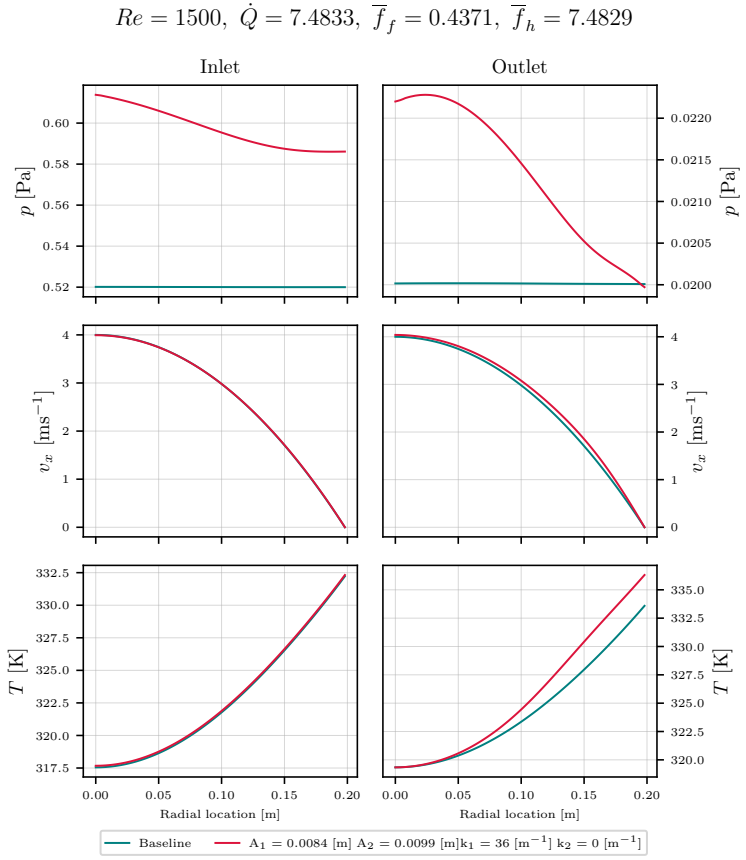


Figure 7.117: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface plate with highest $\overline{\dot{Q}}$ predicted by the M-RBF surrogate model at $Re = 1500$.

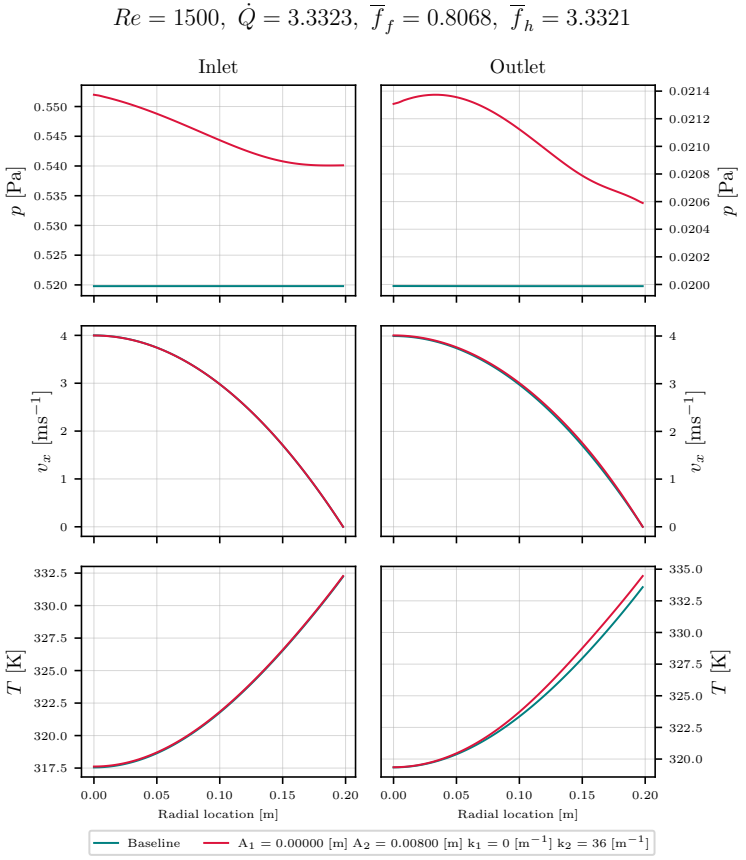


Figure 7.118: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 1500$.

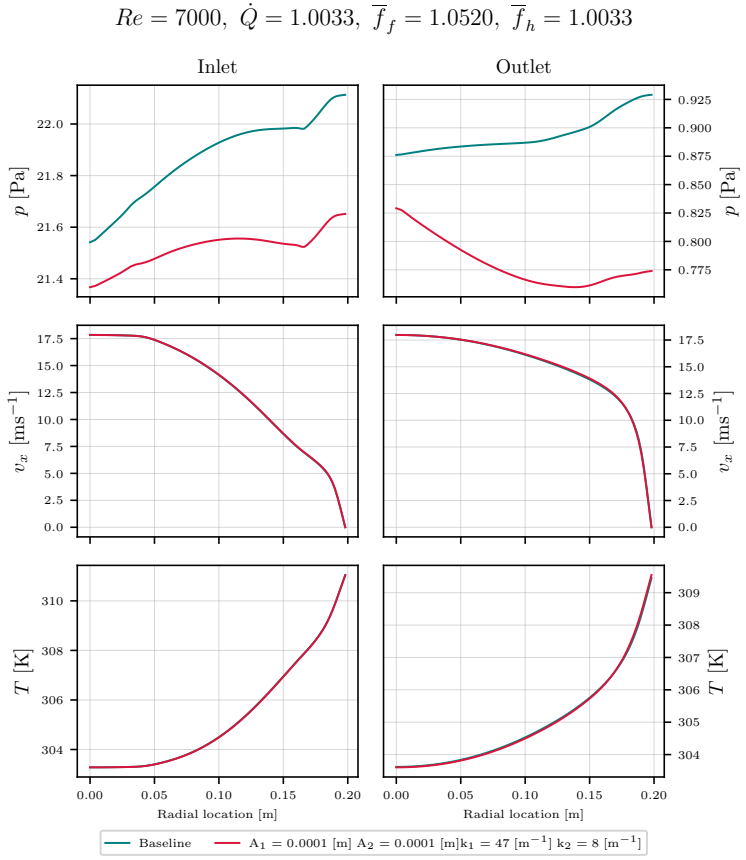


Figure 7.119: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface plate with highest $\overline{\dot{Q}}$ predicted by the M-RBF surrogate model at $Re = 7000$.

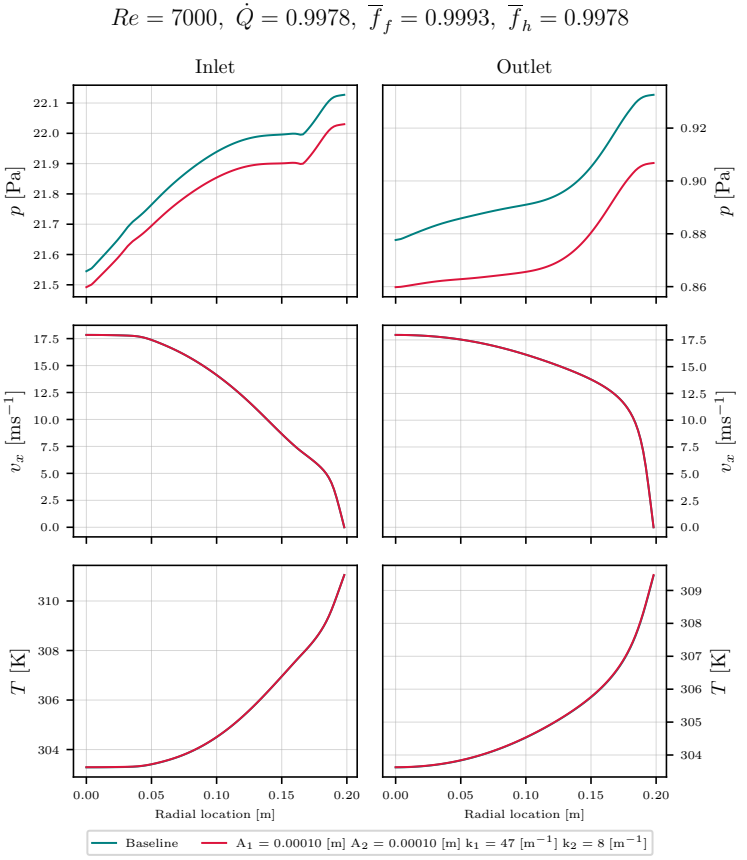


Figure 7.120: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 7000$.

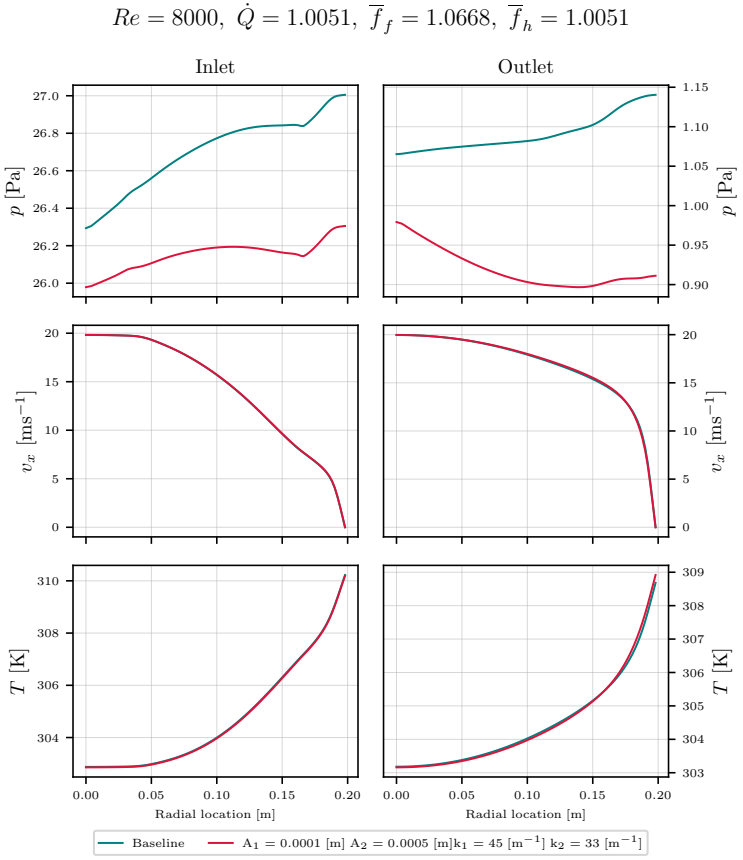


Figure 7.121: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface plate with highest $\overline{\dot{Q}}$ predicted by the M-RBF surrogate model at $Re = 8000$.

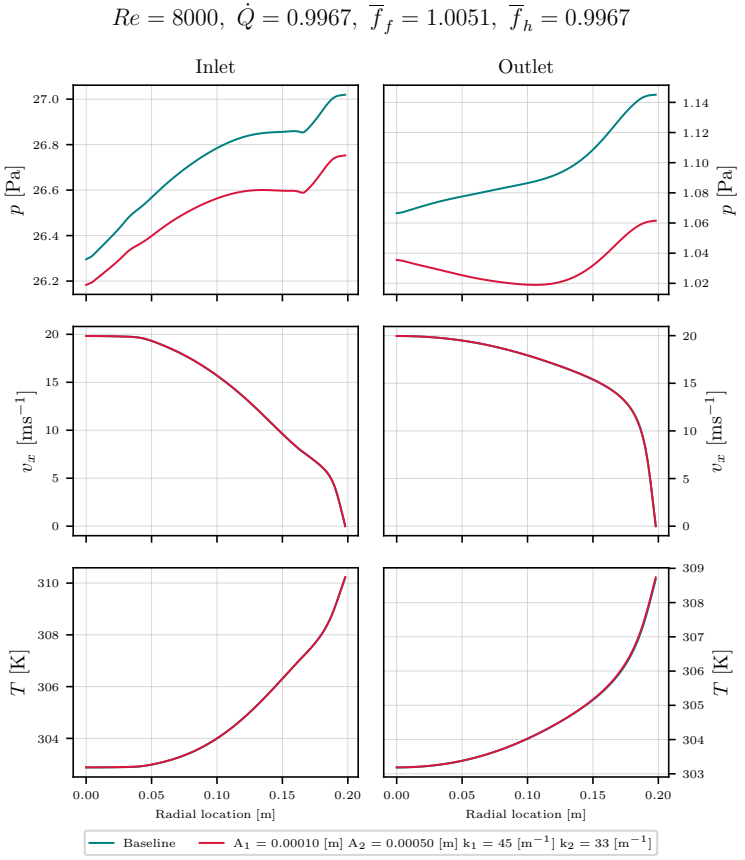


Figure 7.122: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 8000$.

7.4.2 Neural Networks

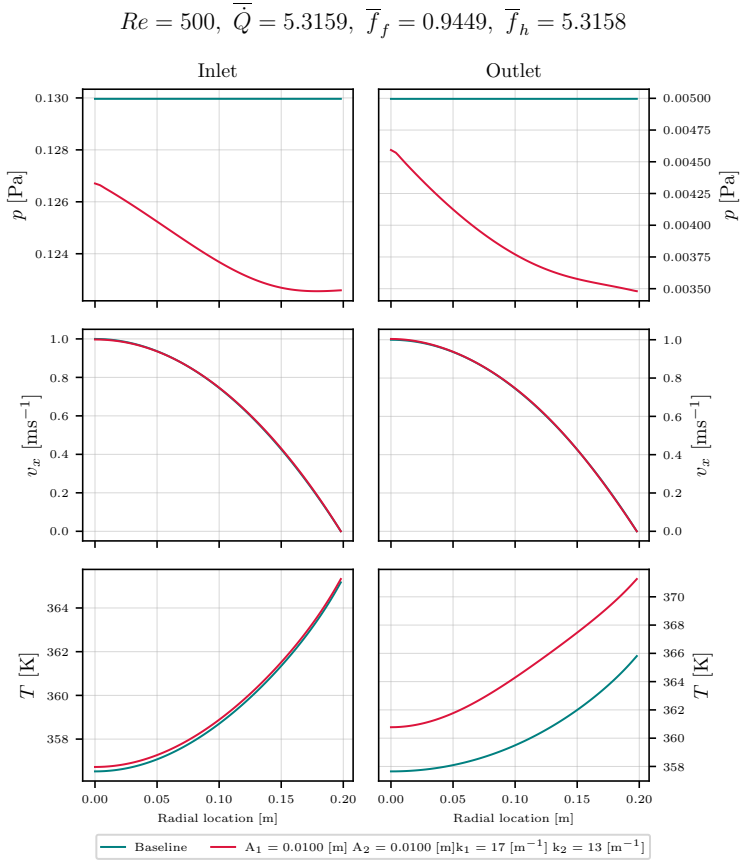


Figure 7.123: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface plate with highest $\dot{Q}_m$ vs baseline $\dot{Q}_0$ predicted by the M-NN surrogate model at $Re = 500$.

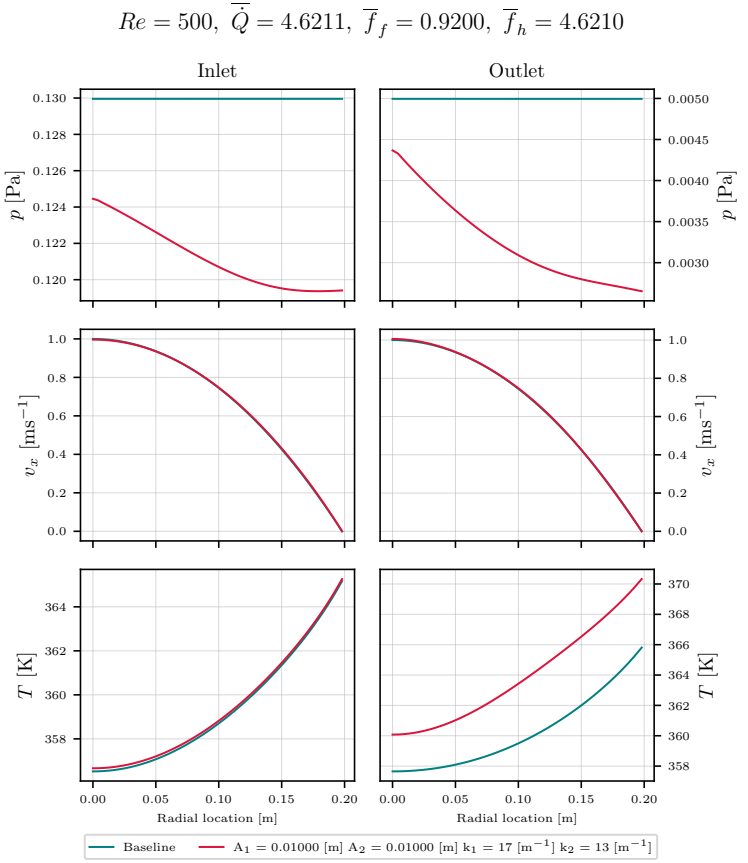


Figure 7.124: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 500$.

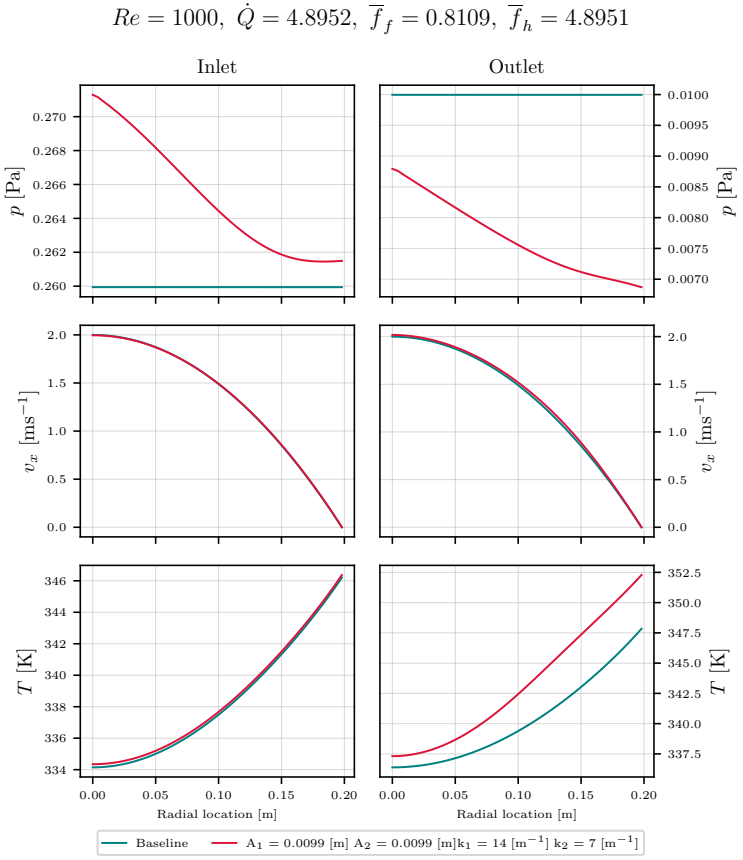


Figure 7.125: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface plate with highest $\dot{Q}_m$ vs baseline $\dot{Q}_0$ predicted by the M-NN surrogate model at $Re = 1000$.

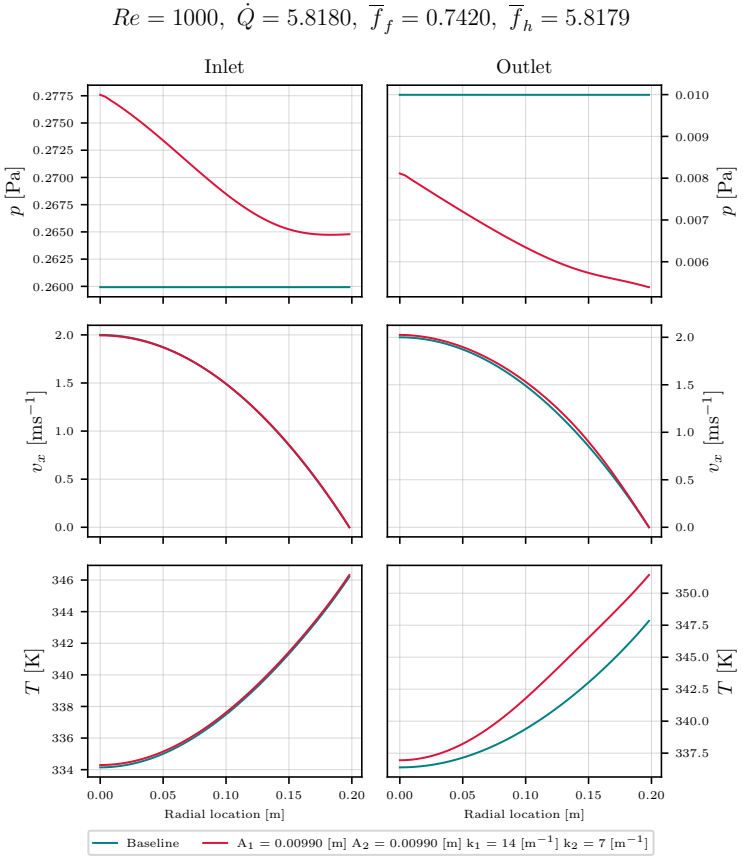


Figure 7.126: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 1000$.

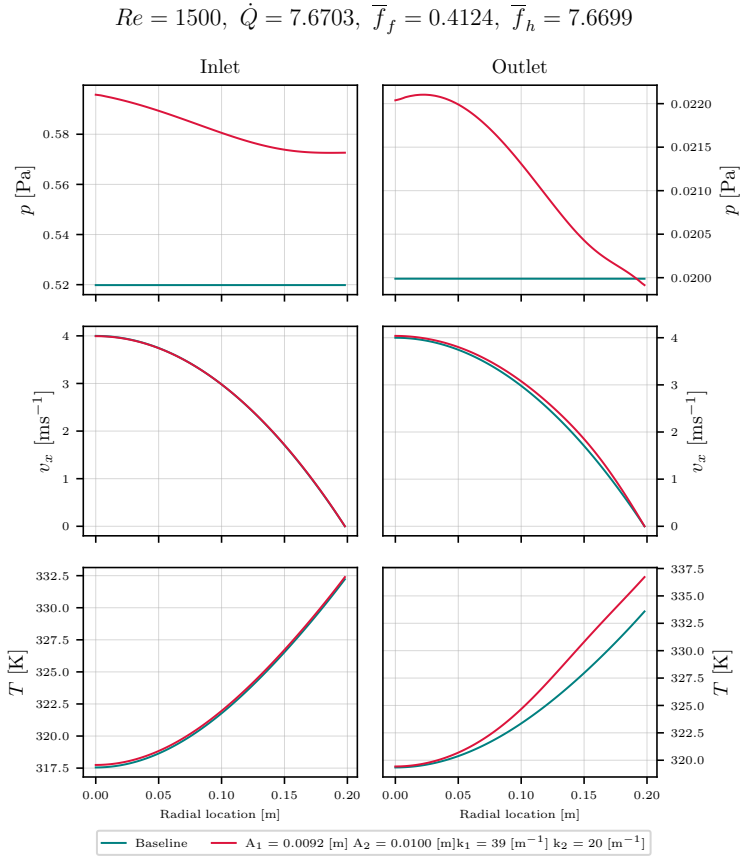


Figure 7.127: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface plate with highest $\dot{Q}_m$ vs baseline $\dot{Q}_0$ predicted by the M-NN surrogate model at $Re = 1500$.

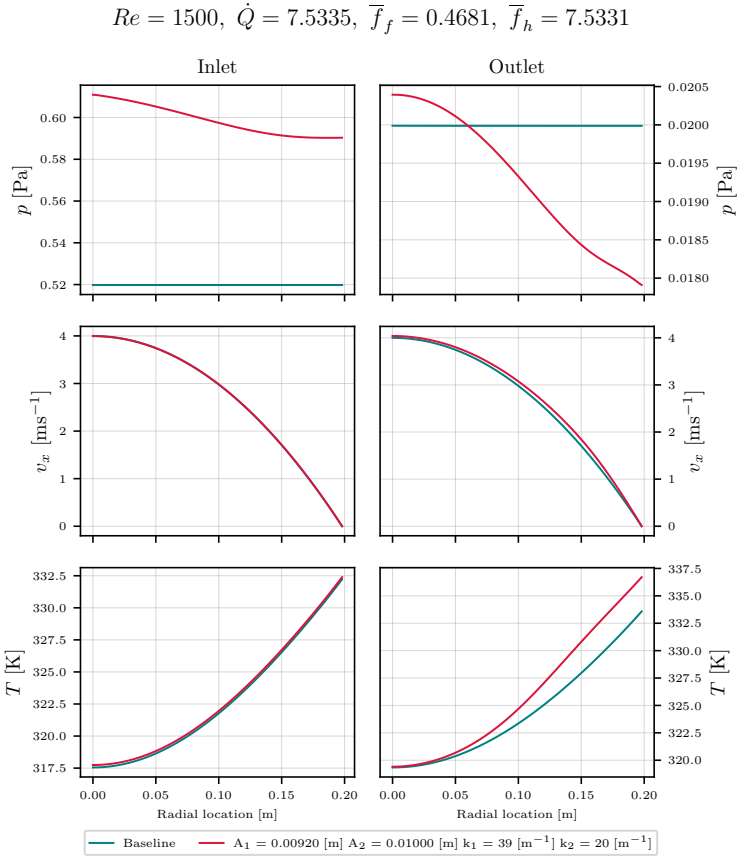


Figure 7.128: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 1500$.

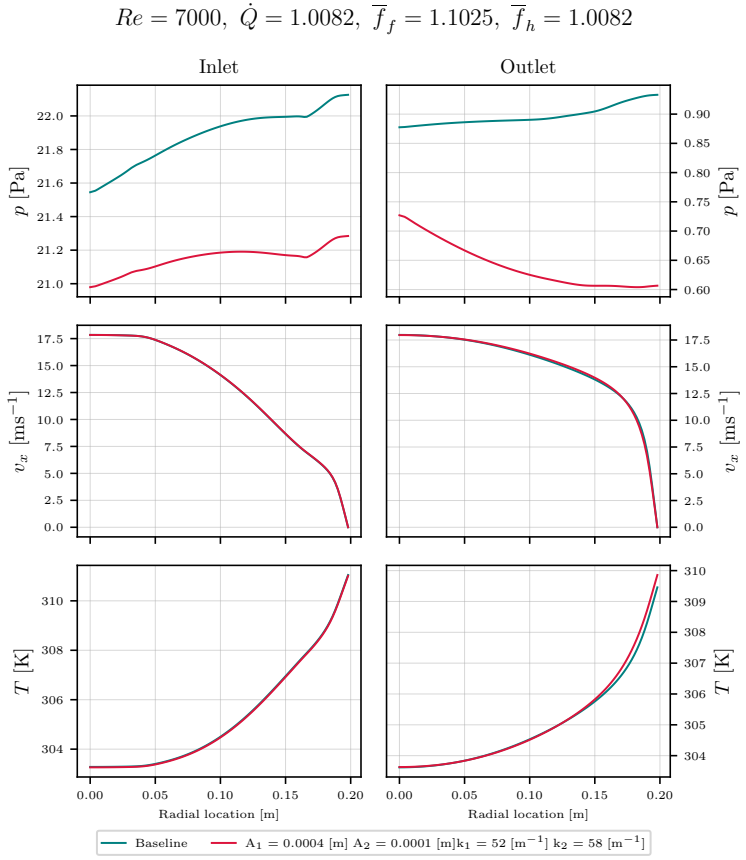


Figure 7.129: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface plate with highest $\dot{Q}_m$ vs baseline $\dot{Q}_0$ predicted by the M-NN surrogate model at $Re = 7000$.

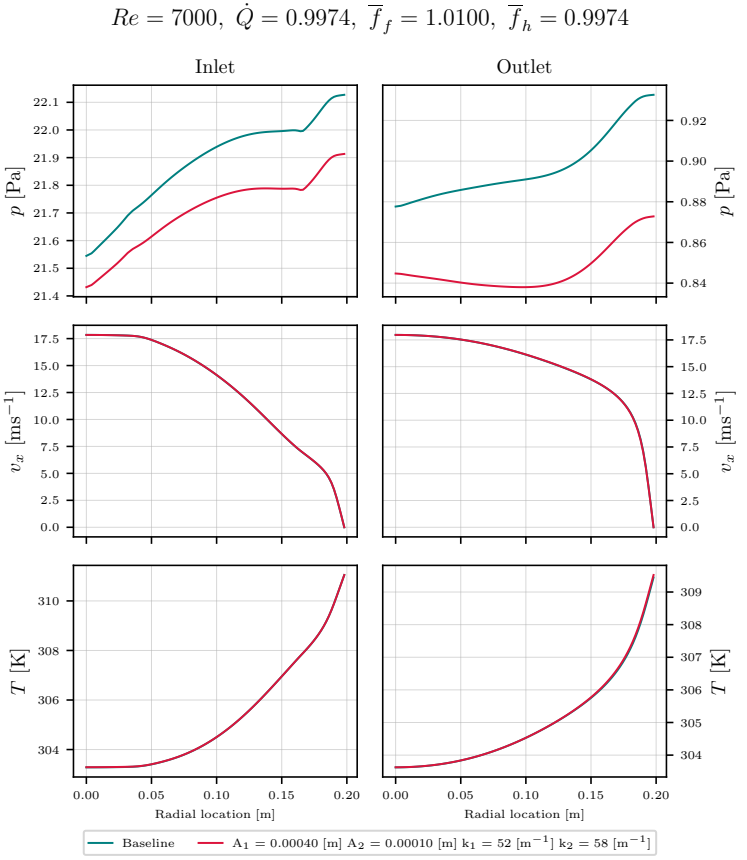


Figure 7.130: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 7000$.

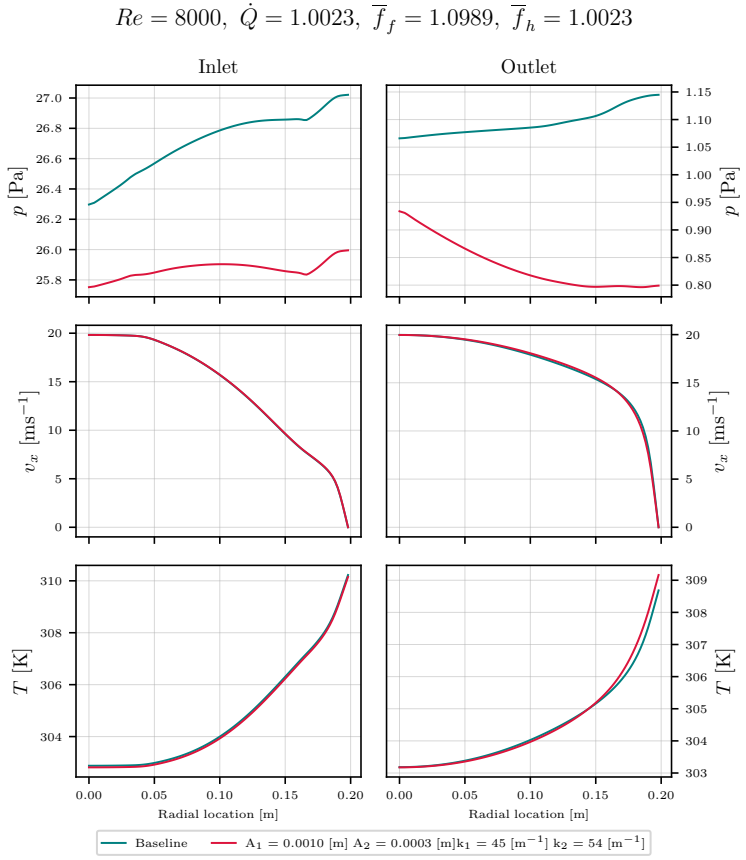


Figure 7.131: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface plate with highest $\dot{Q}_m$ vs baseline $\dot{Q}_0$ predicted by the M-NN surrogate model at $Re = 8000$.

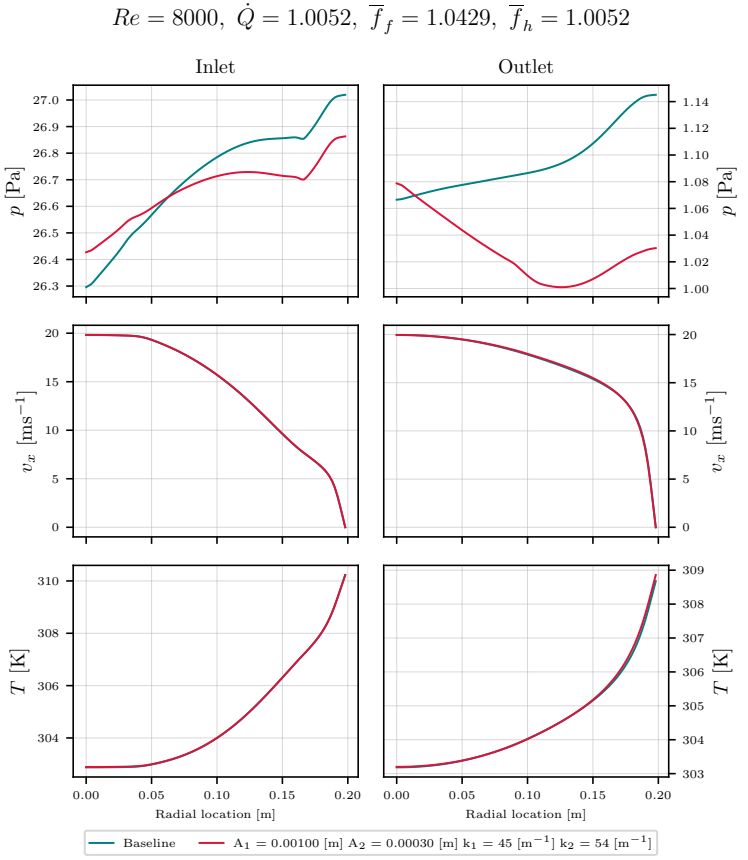


Figure 7.132: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 8000$.

7.4.3 Gaussian Processes

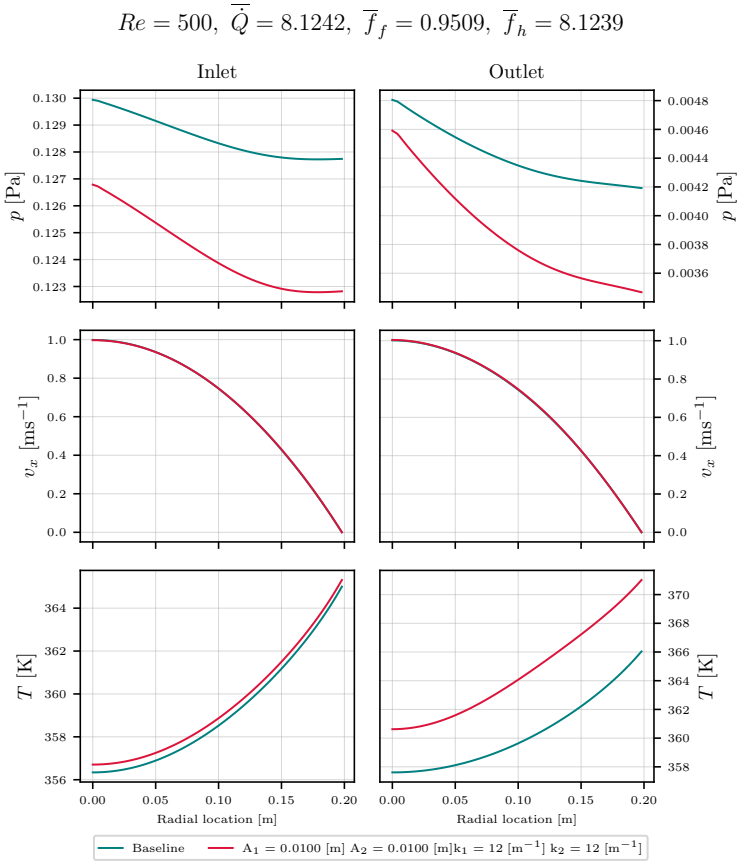


Figure 7.133: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface obtained with the M-GP surrogate model at $Re = 500$.

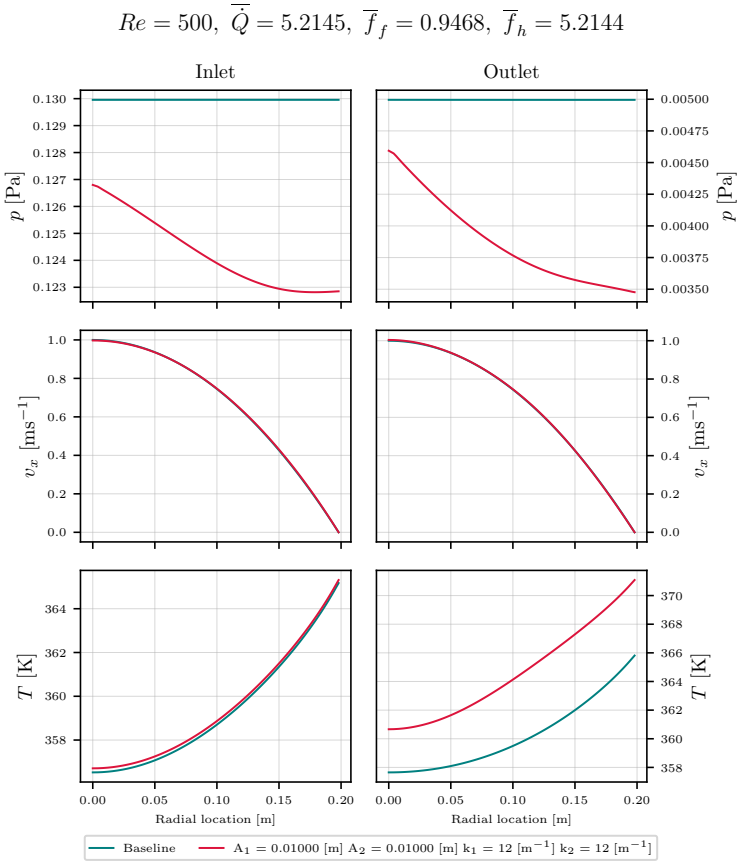


Figure 7.134: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 500$.

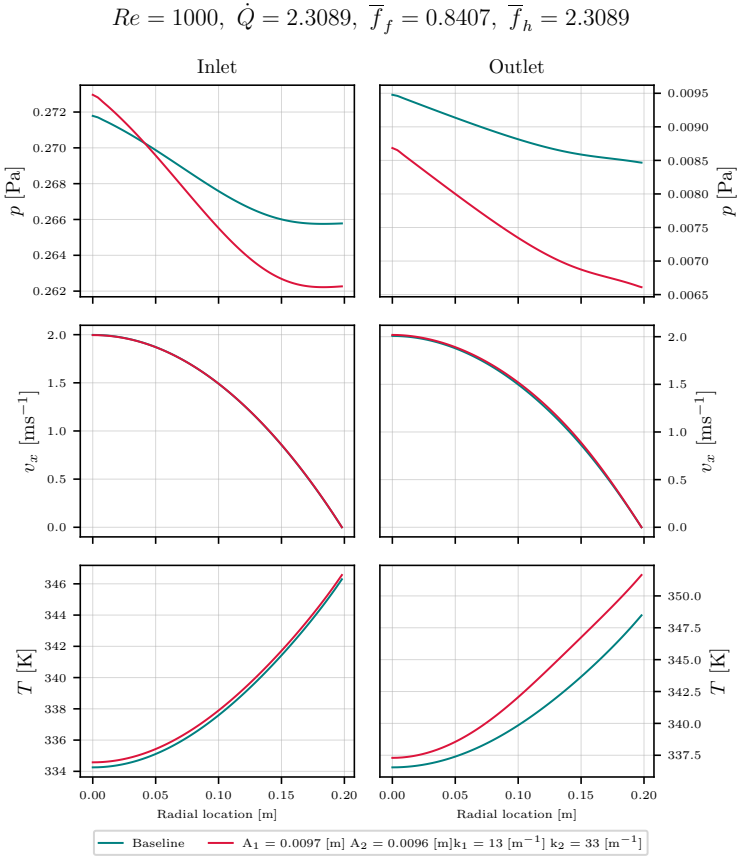


Figure 7.135: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface obtained with the M-GP surrogate model at $Re = 1000$.

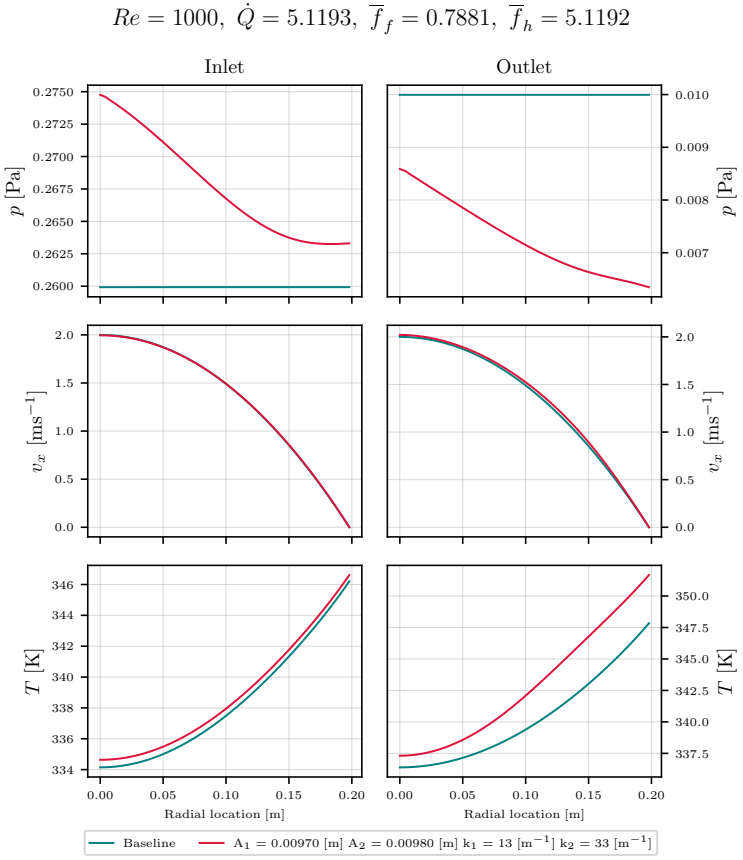


Figure 7.136: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 1000$.

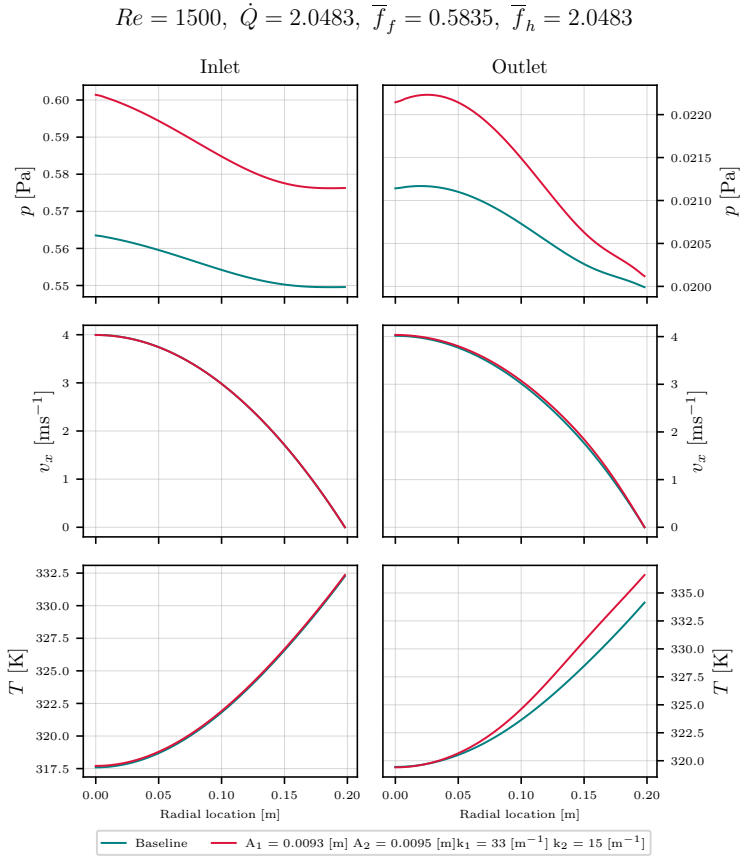


Figure 7.137: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface obtained with the M-GP surrogate model at $Re = 1500$.

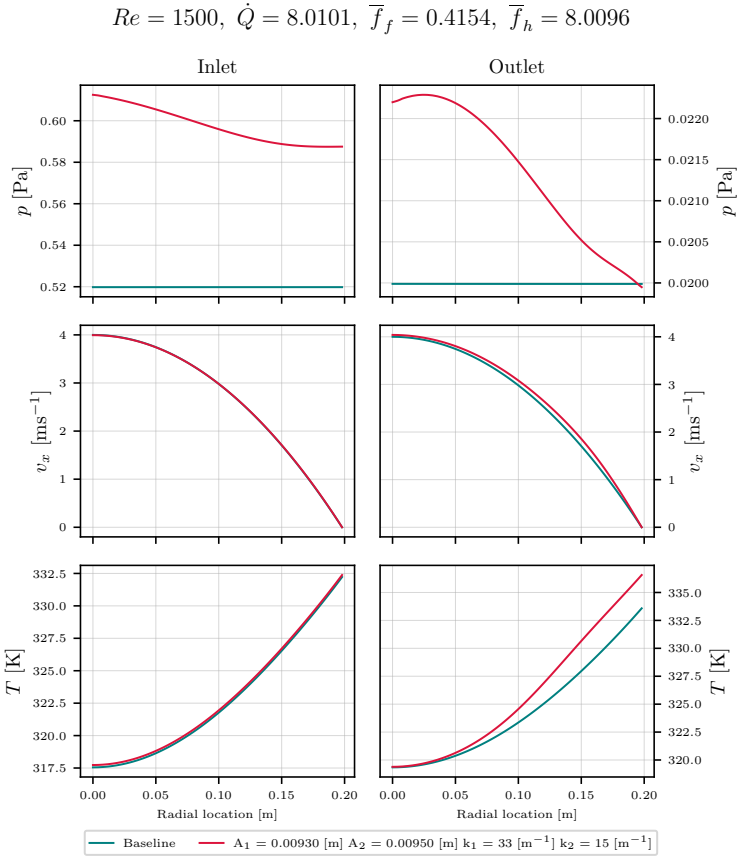


Figure 7.138: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 1500$.

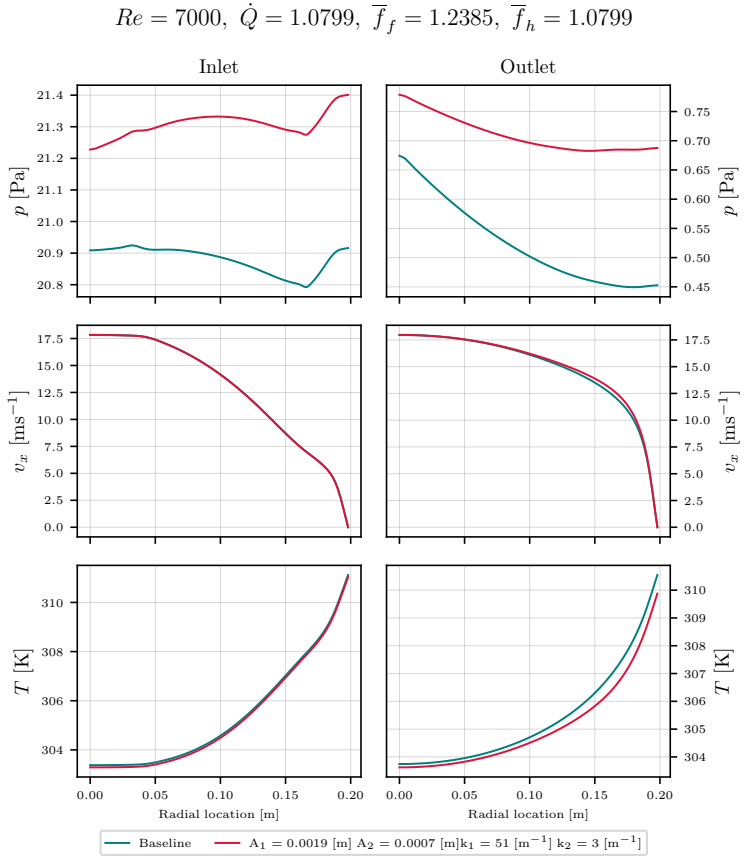


Figure 7.139: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface obtained with the M-GP surrogate model at $Re = 7000$.

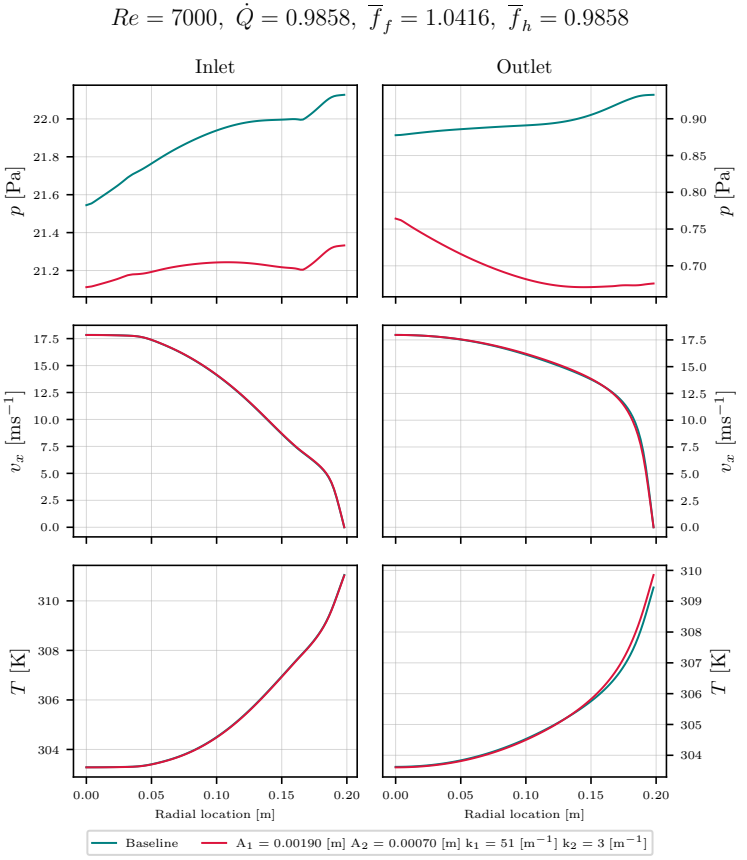


Figure 7.140: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 7000$.

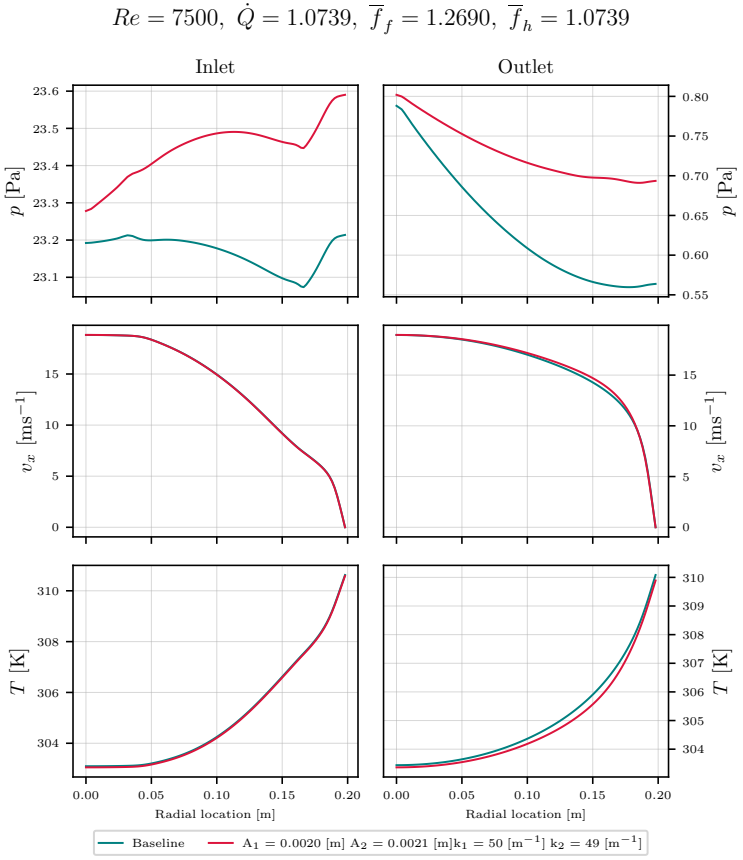


Figure 7.141: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface obtained with the M-GP surrogate model at $Re = 7500$.

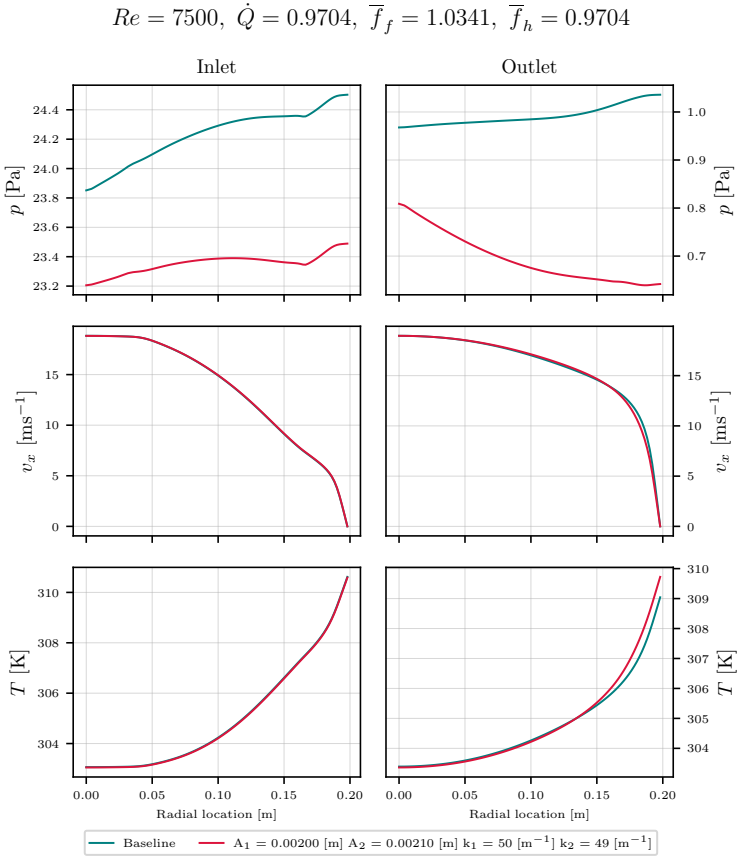


Figure 7.142: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 7500$.

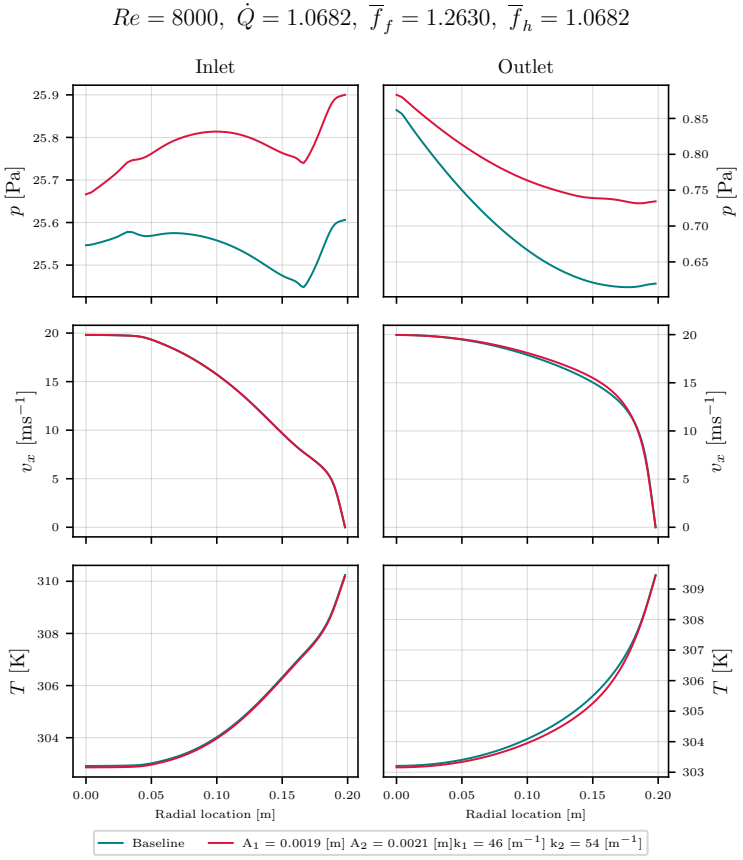


Figure 7.143: THP optimisation benchmark of the surrogate models - Profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface obtained with the M-GP surrogate model at $Re = 8000$.

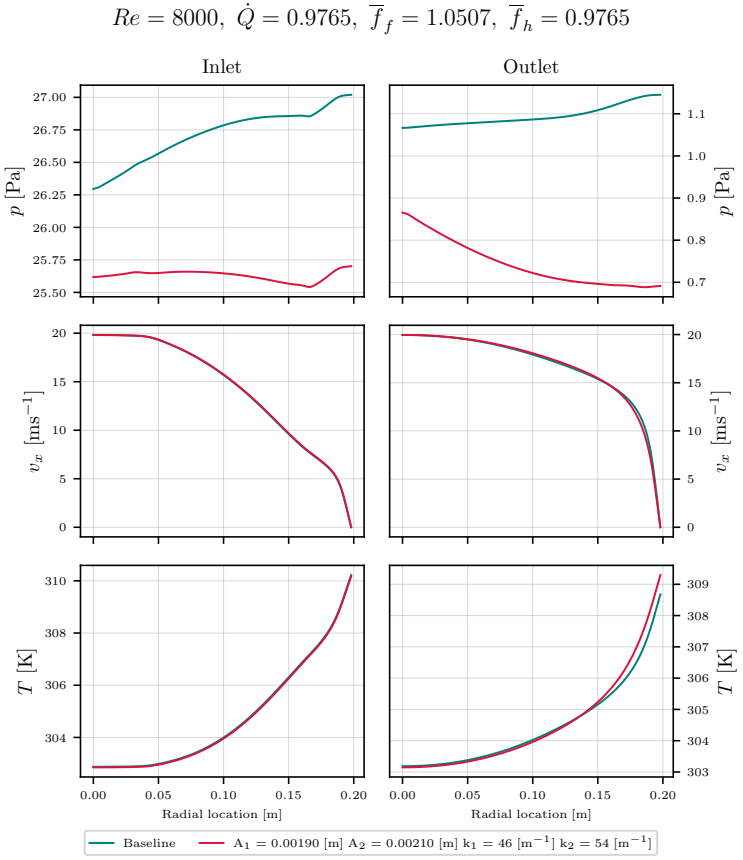


Figure 7.144: THP optimisation benchmark of the surrogate models - High fidelity verification of profile plots of –from top to bottom– pressure p, stream-wise velocity v_x and temperature T fields of 2D flow over a rough surface at $Re = 8000$.

7.4.4 Discussion and findings

From the previous observations, optimal geometries were found at the parametric space boundaries as in the high fidelity study from Section 6.3, demonstrating the models' response to the THP optimisation problem, as stated in Objective VII.VI. This suggests that a higher density mapping is required outside this parameters to find potential maxima $\dot{Q}_m$ values. Particularly for turbulence regimes, this gives an idea of the parametric region where $\dot{Q}_m$ optimisation might be found in Re values where there was none with high fidelity data or the current surrogate models –since the inverse problem with these is limited to the parametric region to avoid extrapolation–. This idea is tied to the surrogate model predictions at $Re = \{7500,\ 7000\}$, which yielded geometries with overestimated $\dot{Q}_m$, likely due to the extrapolation close to the baseline case –$\mathrm{A}_i = 0.0\mathrm{m}$ and $\mathrm{k}_i = 0.0\mathrm{m}^{-1}$ are used as the lower limit, when it should be $\mathrm{A}_i = 0.002\mathrm{m}$ and $\mathrm{k}_i = 12\mathrm{m}^{-1}$–, a shortcoming exposed in accordance to Objective VII.VIII. Adding high fidelity data within this region and adjusting the lower limit of the inverse problem will potentially increase the accuracy of the surrogate models and optimisation search process.

The overestimated values of $\overline{\dot{Q}}$ from MRBF at $Re = 1000$ are related to over-fitting towards $\mathrm{k}_i \approx 0\mathrm{m}^{-1}$, where extrapolation is visible in Figure 7.145, while MNN at $Re = 500$ are related to over-fitting in between $\mathrm{k}_i = [0,\ 20]$, where slight non-linearities are visible in Figure 7.146. A second MNN model yielded results that agree with the rest of surrogate models at $Re = 500$ as seen in Figures 7.147, with a better function along the non-linearities visible in Figure 7.148. The results of the first model are shown as part of the shortcomings to address from Objective VII.VIII.

The inaccurate $\dot{Q}_0$ results from MGP are related to the baseline values being smoothed out in favour of a linear function for the kernel, reducing over-fitting in the parametric region excluding the baseline, observed in Figures 7.111 and 7.112 from Section 7.3. To continue with shortcomings from Objective VII.VIII, this translates as a less accurate reading at the baseline –probably considered an outlier by the model–, but the feature values are closer to the high fidelity data $\dot{Q}_m$ within the parameter space, suggesting a better fit within that region. The inaccuracies are reflected in the temperature and pressure profiles, where we observed very good accuracy from the rough surface prediction but extremely poorly portrayed baseline variables, which yield unreliable normalised results $\overline{\dot{Q}}$, $\overline{f}_f$ and $\overline{f}_h$. Regardless of these errors, MGP gave a geometry with great optimisation at $Re = 1500$, where MRBF was most misleading, which demonstrates Objective VII.VII for MGP.

Overall, MNN yielded verified geometries for $Re = 8000$, $Re = 1000$ and $Re = 1500$ with optimised $\dot{Q}_m$ compared to any from Section 6.3, and MGP did for $Re = 1000$ and $Re = 1500$ in spite of the estimation errors, proving the enhancement these models bring to the optimisation process, satisfying Objective VII.VII.

7.5 Summary and conclusion

This chapter presented the architecture benchmark of the ML algorithms Radial Basis Function (RBF) interpolation, Neural Networks (NN) and Gaussian Processes (GP) applied to the momentum and heat transfer optimisation problem. To analyse the sensitivity of

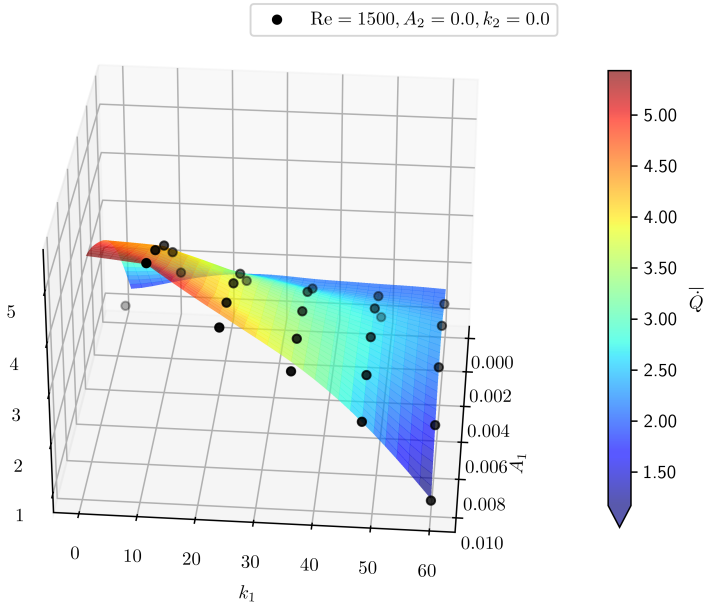


Figure 7.145: THP optimisation benchmark of the surrogate models - $\overline{\dot{Q}} = \dot{Q}_m/\dot{Q}_0$ prediction surface vs high fidelity data points from the $\boldsymbol{\omega}_4$ feature RBF trained with a validation split of 35% for $Re = 1500$.

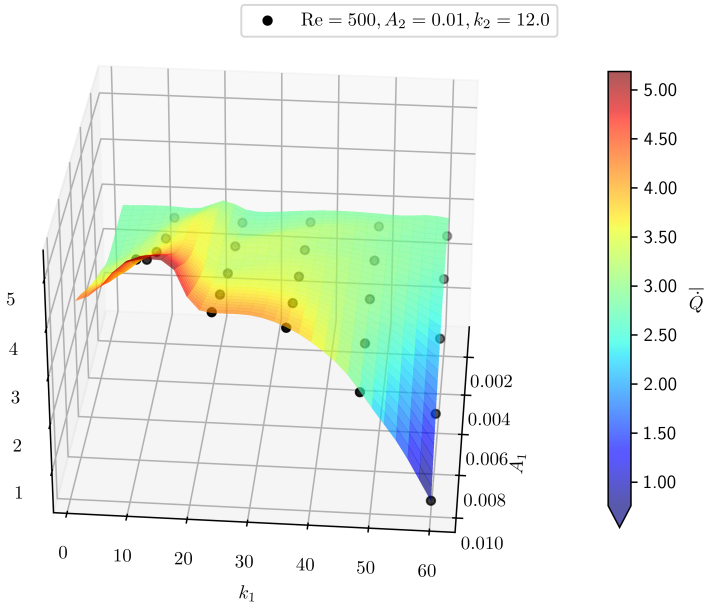


Figure 7.146: THP optimisation benchmark of the surrogate models - $\overline{\dot{Q}} = \dot{Q}_m/\dot{Q}_0$ prediction surface vs high fidelity data points from the $\boldsymbol{\omega}_4$ feature NN trained with a validation split of 35% for $Re = 500$.

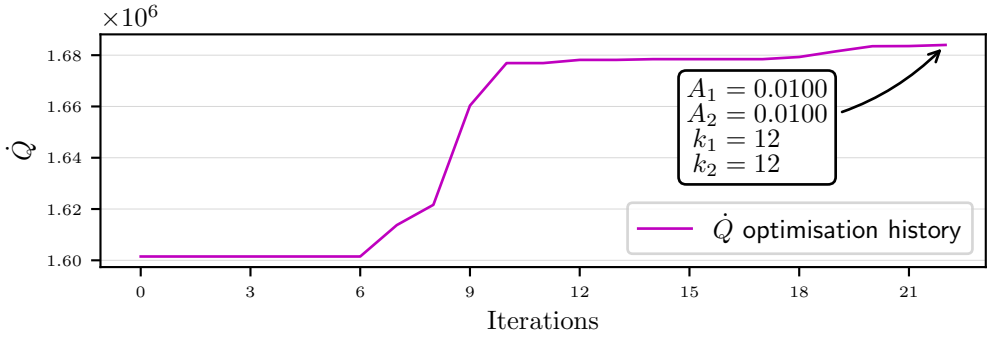


Figure 7.147: THP optimisation benchmark of the surrogate models - History plot of two randomly initialised MNN $\dot{Q}$ prediction evolution within the optimal parameter search made through the heuristic method `differential_evolution` from ScyPy package.

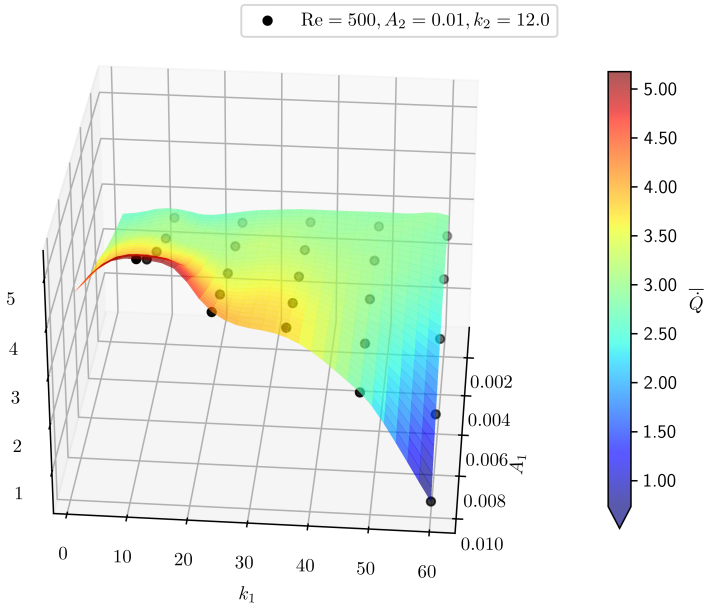


Figure 7.148: THP optimisation benchmark of the surrogate models - $\overline{\dot{Q}} = \dot{Q}_m/\dot{Q}_0$ prediction surface vs high fidelity data points from the $\boldsymbol{\omega}_4$ feature NN –secondary model– trained with a validation split of 35% for $Re = 500$.

each model to varying input-output dimensions, the ML algorithm models were evaluated for increasingly complex problems with $\boldsymbol{\omega}_2$, $\boldsymbol{\omega}_3$, $\boldsymbol{\omega}_4$ and $\boldsymbol{\omega}_5$ input features including amplitudes and wavenumbers of a harmonic function as shape parameters, and *Re* as flow parameter. Multiple output dimensions were tested as well, with three possible matrices: mesh data points as the spatial $\mathbf{x}^{\Gamma_f}$, directly the net power output as the lumped $\mathbf{x}^{\dot{Q}}$, and the Proper Orthogonal Decomposition (POD) used to reduce the dimension of the variable profile data points as the modal $\mathbf{x}^{\mathbf{P}}$. It is concluded that, excluding RBF, the modal proves to have the best accuracy, with consistency regardless of the random initialisation across architectures and low absolute error numbers Er_r. The RBF as an interpolation tool is extremely powerful, but prone to over-fitting when left with no smoothing hyper-parameters. Between the NN and GP, the latter is more computationally expensive but comes with the added capability of *confidence region*. The ML-based THP optimisation is presented as an addition to the high-fidelity data based optimisation raised in Chapter 6. The interest of finding new regions and values outside the high fidelity found maxima relies on the possibility to increase the density of simulation data *a posteriori*. The insight the surrogate models give helps finding the regions to study further with high-fidelity models.

Section 7.2 addressed Objectives VII.I and VII.I, presenting the method to obtain low-rank matrices that represent the high-dimension datasets of flow variables from different rough surface geometries interacting with the flow in a lower dimension. An eigen-analysis through SVD is used to perform and evaluate the dimensionality reduction of flow variables $\mathbf{D}\left(\boldsymbol{\omega}_{\mathrm{N}_i}\right)$ on various independent shape and flow features –$\boldsymbol{\omega}_2$ to $\boldsymbol{\omega}_5$–. It was found that the data has high correlation between shape parameters, where the tolerance cut-off at $\boldsymbol{\omega}_2$ and $\boldsymbol{\omega}_4$ was σ_5 to σ_{15}. The feature correlation with *Re* is low, where the tolerance cut-off is between σ_{10} and σ_{20} for $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$ to produce reliable reconstructions. This leads us to conclude that the reduced matrix rank scales up faster with poorly correlated features.

Section 7.3 presents a performance assessment for RBF, NN and GP algorithms with multiple input features and output dimensions based on 6 characteristics: convergence, consistency, over-fitting, data and computational cost, and complexity sensitivity. In agreement with Objectives VII.III, VII.IV and VII.V, each model is described and evaluated independently for each set of features $\boldsymbol{\omega}_{\mathrm{N}_i}$ and output dimensions $\mathbf{x}^{\mathrm{N}_x}$ with variations in architecture, where neither of the models were found suitable to approach the problem with $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$. However, the assessment for $\boldsymbol{\omega}_4$ gave insight on the advantages of each algorithm.

RBF has the lowest computational cost by several orders of magnitude both in the training and the prediction stages, and even though it is a linear system, it generates accurate results for turbulent regimes where $\overline{\dot{Q}} = [0, 1]$ with a Linear kernel. NN approximates accurately the non-linear behaviour in laminar regimes where $\overline{\dot{Q}} = [-12, 8]$ with 3 layers and 128 neurons, where the low-rank output computed in Section 7.2, modal output $\mathbf{x}^{\mathbf{P}}$, has low CPU times compared to the flow variables from the spatial output $\mathbf{x}^{\Gamma_f}$ while still retaining such information. GP models give a region of confidence with their prediction, showing the areas in the parameter space where the model function is poorly correlated. This insight can be used to understand the data cost and sensitivity of any model, to potentially reduce ill-posedness and over-fitting. Overall, the Matern kernel has the lowest CPU time and highest accuracy for the GP models. From the results of modal (M-) NN, each algorithm with their respective *best* architecture parameters use the output $\mathbf{x}^{\mathbf{P}}$ to approach the THP optimisation problem in Section 7.4.

The study addresses Objectives VII.VI, VII.VII and VII.VIII, exposing advantages and shortcomings of using surrogate models to aid in the optimal parameter search. Similarly to Section 6.3, optimal geometries were found at the parametric space boundaries, where extrapolation was found for the lower limits of the shape parameters –baseline values $\mathrm{A}_i = 0.0\mathrm{m}$ and $\mathrm{k}_i = 0.0\mathrm{m}^{-1}$ are used as the lower limit, when it should be $\mathrm{A}_i = 0.002\mathrm{m}$ and $\mathrm{k}_i = 12\mathrm{m}^{-1}$–. We discuss that a higher density mapping and adjustment of the lower limit for the inverse problem might increase the surrogate models' accuracy and optimisation search process.
Both MNN and MRBF have problems with over-fitting close to the highest $\dot{Q}_m$, overestimating optimal values when searching for roughness parameters. MGP on the other hand, has inaccurate $\dot{Q}_0$ results since the baseline values are considered outliers in favour of reducing over-fitting in the parametric region and producing a more linear function. The models' inaccuracies are reflected in the prediction of flow variables with rough surfaces for MNN and MRBF, and the baseline for MGP, which demonstrates Objective VII.VIII. However, MNN and MGP predicted geometries with optimised $\dot{Q}_m$ compared to any from Section 6.3, at $Re = \{1000,\ 8000\}$ from MNN and at $Re = 1500$ from GP, satisfying Objective VII.VII.
Overall, the outcomes from this chapter are

Out VII.I. Low-rank matrix $\mathbf{P}$ representation of the flow data from the high fidelity models $\mathbf{D}(\boldsymbol{\omega}_{\mathrm{N}_i})$ through the appropriate truncation of σ_{N_i} modes.

Out VII.II. Insight on the non-linearities the flow parameter Re adds to the matrix $\mathbf{D}(\boldsymbol{\omega}_{\mathrm{N}_i})$ when introduced as an independent variable into the input features $\boldsymbol{\omega}_{\mathrm{N}_i}$ alongside the shape parameters amplitude A_i and wavenumber k_i, by an eigen-analysis of $\mathbf{D}(\boldsymbol{\omega}_{\mathrm{N}_i})$ when looking for a low-rank matrix representation of the matrix.

Out VII.III. In depth analysis of surrogate modelling performance by examining the characteristics of convergence, consistency, over-fitting, data cost, computational cost and complexity sensitivity in the three algorithms of interest: RBF, NN and GP.

Out VII.IV. Report on the impact that the correlation between features $\boldsymbol{\omega}_{\mathrm{N}_i}$ and amount of measurements $\mathbf{x}^{\mathrm{N}_x}$ of the matrix $\mathbf{D}_{\mathrm{N}_i}^{\mathrm{N}_x}(\boldsymbol{\omega}_{\mathrm{N}_i})$ has on each surrogate models' performance.

Out VII.V. Insight on the required architecture complexity for high non-linearities in the input features $\boldsymbol{\omega}_{\mathrm{N}_i}$ and how this limits the capacity of all algorithms to predict the influence of geometry variation on the thermo-hydraulic behaviour of the flow when Re is introduced as an independent variable in the surrogate model.

Out VII.VI. Obtained various optimised geometries from the prediction of surrogate models, 3 of which were verified through high fidelity modelling to have THP higher than the baseline and any case studied prior to applying the inverse problem.

Out VII.VII. Report on the found shortcomings of surrogate models in predicting flow variables to approach the THP optimisation problem, namely over-fitting from NN and RBF, and over-linearisation from GP.

CHAPTER 8

Concluding remarks

"It's the questions we can't answer that teach us the most. They teach us how to think. If you give a man an answer, all he gains is a little fact. But give him a question and he'll look for his own answers."

— Patrick Rothfuss, The wise man's fear, 2011

8.1 Findings overview and outcomes

This project presents the exploratory research of momentum and heat transfer optimisation in pipe flows with rough surfaces based on simplified biomimetic structures. To address this problem, the general objectives stipulated in Chapter 1 are recapitulated

Obj I. Design open-source high fidelity computational models of steady-state, two-dimensional pipe cross-section flows in a conjugate heat transfer system with cold flow.

Obj II. Verification and validation of the numerical approach against analytical data and previous research studies.

These first two objectives are addressed in Section 6.2 and Appendix C.2. OpenFOAM v2212 is adopted to approach the discretisation of the pipe cross-section model with Finite Volume Method formulation, where the conjugate heat transfer is handled by the solver `chtMultiRegionSimpleFoam`. An $h-$refinement study is presented in Section 6.2 to verify the selection of OpenFOAM numerical schemes and mesh independence of the model. The analytical solution to the Hagen-Poiseuille law and Nusselt thermal correlations are presented in Appendix C.2 to validate the flow variable fields from the numerical model.
The L^2 norm error from 3 levels of mesh refinement with ratio r= 2 of the numerical solution to $Re = \{500,\ 8000\}$ flows have rates of convergence between 1 and 3, generally expected for 1st and 2nd order schemes. Laminar temperature, at rates under 1, has a thermal behaviour as expected in a fully developed thermal layer in a cylindrical pipe with constant wall temperature from Appendix C.2. The mean temperature in the flow increases in value due to the interaction with the wall and has a time-dependent exponential decay, reducing the convergence of temperature in steady-state analysis.
The overall rates decay with increasing geometric complexity, but are still within the expected values around 2, and suggest results are mesh independent. The results are in agreement with the analytical Hagen-Poiseuille law and Nu algebraic predictions for laminar regimes, and on turbulent flows, the Nu are under the values suggested in literature [137] due to the developing boundary layer. The $1-$level mesh has been selected as a base for both low and high Re regimes to properly represent the smaller-scale features of surface roughness.

Obj III. Explore parametrised geometries based on simplified biomimetic structures on the wetted surface of the pipe and their influence on heat transfer and flow behaviour.

Obj IV. Gain insight on the thermal and hydraulic mechanisms of the flow when influenced by different topologies.

These objectives are discussed in Section 6.3, presenting the analysis of different flow regimes at $Re = [500,\ 8000]$ over rough surfaces. The flow has a wide variety of reactions to rough surfaces depending on the geometric parameters, which in the case of this study include the amplitude A_i and wavenumber k_i of a harmonic function 5.1. From the equations of dissipation rate (2.52) and advected heat flux (2.54) and the results from Section 6.3, we observe f_h is dominant over f_f when studying $\dot{Q}$, but both depend on the velocity profile changes of the flow. Accordingly, increasing the wetted surface area and average velocity resulted in the optimisation of $\dot{Q}$ in both laminar and turbulent regimes. The main flow behaviour observed

includes an increase in turbulent mixing, recirculation eddies in-between geometric elements and separation. However, each regimes seems to benefit from opposite spectrums of the parametric space examined in this study.

Sharp elevation changes sparsely located on the surface enhance $\overline{\dot{Q}}$ in all the laminar regimes evaluated $Re = [500,\ 1500]$, where deep structures trap eddies that enhance thermal interaction with low disruption to the main flow in laminar regimes, resulting in high heat transfer and low dissipation. The disruption looks like transition to turbulence reported in literature [92, 94, 115, 116], since the velocity shifts slightly towards a logarithmic profile rather than the parabolic from laminar flows, albeit transition was not modelled property due to the lack of a turbulence model under $Re = 2000$. In turbulent regimes the dissipation from the disruption of the boundary layer leads to velocity decay, reducing the advected heat flux. Therefore, turbulence gains from dense elements with low elevation, where the disruption to the logarithmic layer is minor, although enhancement was only found in $Re = \{7000, 8000\}$ with an increase in dissipation rate. This same behaviour was observed by Stalio and Nobile [29], Benhalilou and Kasagi [104] and Jin [102], who observed either no enhancement at low Pr or enhancement with drag penalty in turbulent flow at $Pr = 100$. Wiesche [103] reports enhancement within the drag reduction regime at $Pr = 0.01$, contradicting the previous findings. Note that cross-stream vorticity is not modelled in a two-dimensional domain, potentially reducing the thermal gain from these structures. In these geometries there are no trapped eddies but the structures induce separation in the valleys and early reattachment just before the peaks, increasing the boundary layer thickness and the velocity outside the boundary layer.

A geometry with dense, deep elements has a negative impact on $\overline{\dot{Q}}$ for both regimes, where two trapped eddies form in the valleys instead of one, and act as an insulation layer rather than enhance the thermal interaction due to separation between them, the wall and the main flow. Different trapped eddy structures were studied by Nikuradse [94] and Perry [113], and Soleimani [23] summarises the insulation mechanism of these structures.

Obj V. Identify the advantages and drawbacks of alternative ML techniques like Radial Basis Function (RBF) interpolation, Neural Networks (NN), and Gaussian Processes (GP) when employed to evaluate the Thermo-Hydraulic Performance (THP) optimisation in pipe flows.

The Machine Learning (ML) algorithms' assessment presented in Chapter 7 comprises the resolution of this last objective. The algorithms are evaluated through 6 different characteristics: convergence, consistency, over-fitting, data cost, computational cost and complexity sensitivity, where different architectures and input-output dimensions provide the match to contrast with. The different input dimensions $\boldsymbol{\omega}_{\mathrm{N}_i}$ are based on the shape and flow parameters, A_i, k_i and Re, with variation in the use of Re –$\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$– and single or double harmonics –$\boldsymbol{\omega}_2$ and $\boldsymbol{\omega}_4$–. The output dimensions $\mathbf{x}_{\mathrm{N}_i}$ depend on the treatment of flow variable profiles at the inlet and outlet, where either the data is used raw –temperature, pressure and velocity profiles–, reduced via Proper Orthogonal Decomposition (POD) or the dissipation rate and advected heat flux are computed to train the models.

POD is used as a dimensionality reduction technique in Section 7.2 to extract the main features of the flow variables and reduce the cost and complexity of the problem for the surrogate models. It was found that for $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$, the problem is highly non-linear since the cut-off of

most prominent POD modes required to be $\sigma_2 0$ compared to σ_5 without the Re as a feature. The algorithms evaluated are RBF, NN and GP, none of which were suitable to approach the optimisation problem with $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$, at least with the assessed architectures. The main drawback found is that the models are heavily over-fitting at $Re = 500$ due to the extreme changes observed at laminar regimes described in Section 6.3, and the low correlation of flow and shape parameters found with the eigen-analysis. However, $\boldsymbol{\omega}_4$ has reliable approximations from most models, and each algorithm had a characteristic that makes them advantageous for different situations.

RBF is a computationally cheap linear system of equations, with enough accuracy for the low feature problems. These models are consistent regardless of random initial weight coefficients and convergence to loss values in the order of the machine precision. however, they struggle with non-linear behaviour in $Re = 500$ –$\overline{\dot{Q}} = [-12, 8]$–, which is well addressed by NN models. However, NNs are more computationally expensive, and this cost increases exponentially with output dimension and architecture complexity. The output yielded by POD dimension reduction $\mathbf{x}^{\mathbf{P}}$ from Section 7.2 had low prediction errors at $Re = 500$, and CPU times close to the smallest output $\mathbf{x}^{\dot{Q}}$ while providing the flow variable information from $\mathbf{x}^{\Gamma_f}$. Even though most NNs were not as consistent as RBF, $\mathbf{x}^{\mathbf{P}}$ had low variation with random initial weight coefficients. Additionally, NN has the potential to approach the problem with input features $\boldsymbol{\omega}_5$ if trained for longer or with deeper architectures.

The GP is an algorithm with cubic computational complexity, unfeasible for large datasets and additional features. However, the region of confidence predicted by these models gives insight into the function behaviour, showing areas where data refinement is required or ill-posedness is happening. This characteristic alongside the good consistency of the models establish GP models as a preliminary tool for function behaviour assessment, parametric region exploration and high fidelity data ill-posedness.

This assessment resulted in the selection of output $\mathbf{x}^{\mathbf{P}}$ as an accurate model for flow variables with lower computational cost, and input $\boldsymbol{\omega}_4$ as the maximum features to use for reliable predictions with the current architectures of RBF –Linear kernel–, NN –3 layers, 128 neurons each– and GP –Matern kernel–. The predictions are used to find an optimal geometry in Section 7.4 with the help of Scypy's `differential_evolution`. The parameters from the obtained geometries were verified through high fidelity simulations, which resulted in 3 geometries with higher $\overline{\dot{Q}}$ than any found within the original parameters analysed in Section 6.3. For $Re = 8000$ and $Re = 1000$, NN gave accurate predictions of $\overline{\dot{Q}} = 1.0052$ and $\overline{\dot{Q}} = 5.8179$, while GP predicted $\overline{\dot{Q}} = 8.1066$ for $Re = 1500$. No other geometries tested gave better results than the previously obtained or the baseline. Both laminar regimes are in the lower dissipation rate region, but $Re = 8000$ still yielded higher dissipation rate than the baseline. A concern surges about the use of a heuristic method for the inverse problem, since this techniques output sparse results and often needed several restarts to produce a result in agreement with surface plots from the models predictions and the parametric space study.

In the interest to address these objectives, the open-source software Hammerhead was developed from scratch within this project as an ML-aided THP optimisation toolkit. Hammerhead, described in Chapter 5, is an application that automates the exploratory study of momentum and heat transfer optimisation in pipe flows with rough surfaces through high fidelity

simulations and ML support through the functions

- **High-fidelity database population.** Shape parametrisation, body fitted mesh generation and multi-model simulation running in OpenFOAM.
- **Tensor data process.** High fidelity data sample is transformed into input feature tensors $\boldsymbol{\omega}_{\mathrm{N}_i}$ and output tensors $\mathbf{x}^{\mathrm{N}_x}$.
- **Surrogate model training.** This uses the tensors and modular architectures for each algorithm.
- **Optimal parameters' search.** Predictions from the trained surrogate models are used with a heuristic method to find the shape parameters with *optimal* thermo-hydraulic impact on the flow.
- **Plotting and data visualisation.** This give insight on the flow variables' profiles, surrogate model's characteristics –errors, loss, architecture benchmark, predictions– and optimisation process search.

The general outcomes from Chapters 6 and 7, as well as the Hammerhead software described in Chapter 5 are

Out I. A robust OpenFOAM mesh for exploratory research of thermal and hydraulic optimisation in pipe flows over rough surfaces, verified for a flow regime range of $Re = \{500,\ 8000\}$, with validated fluid flow and heat transfer based on the Hagen-Poiseuille law and thermally fully developed laminar flow analysis.

Out II. In depth analysis of the influence that surface roughness has in fluid flow and heat transfer within laminar and turbulent flow regimes over an extensive database of parametrised geometries, gathering statistics that inform about the best parametric regions for optimisation in each regime and the mechanisms behind enhancement.

Out III. Low-rank matrix $\mathbf{P}$ representation of the flow data from the high fidelity models $\mathbf{D}(\boldsymbol{\omega}_{\mathrm{N}_i})$ through the appropriate truncation of σ_{N_i} modes.

Out IV. Insight on the non-linearities the flow parameter Re adds to the matrix $\mathbf{D}(\boldsymbol{\omega}_{\mathrm{N}_i})$ when introduced as an independent variable into the input features $\boldsymbol{\omega}_{\mathrm{N}_i}$ alongside the shape parameters amplitude A_i and wavenumber k_i, by an eigen-analysis of $\mathbf{D}(\boldsymbol{\omega}_{\mathrm{N}_i})$ when looking for a low-rank matrix representation of the matrix.

Out V. Report on the surrogate modelling performance and impact of the correlation between features $\boldsymbol{\omega}_{\mathrm{N}_i}$ and amount of measurements $\mathbf{x}^{\mathrm{N}_x}$ of the matrix $\mathbf{D}_{\mathrm{N}_i}^{\mathrm{N}_x}(\boldsymbol{\omega}_{\mathrm{N}_i})$ on each model trained with RBF, NN and GP algorithms through a complexity sensitivity analysis.

Out VI. Verification of the surrogate models' optimisation prediction through high fidelity modelling, resulting in 3 geometries with THP higher than the baseline and any case studied prior to applying the inverse problem.

Out VII. Design and development of an open-source, robust and easy-to-use toolkit for the semi-automation of both high fidelity and surrogate modelling for the MOO problem of THP in cooling systems.

8.2 General conclusions

As an exploratory research, the project led to insightful information about the impact of rough surfaces on flow behaviour and the potential these structures have for THP optimisation, even within turbulence regimes at the higher end of Re values examined. Overall, biomimetic structures based on the skin of fast swimming sharks enhance turbulent mixing, increase the wetted surface area and disrupt the boundary layer –and the velocity profile with it–, subsequently affecting the advected heat, convection and the energy generation/dissipation rates. However, optimisation over the baseline cases in turbulence regimes was marginal. This has the possibility to improve with higher Re values or by changing the Pr, as observed in literature [29, 102–104], yet another promising approach arises with a slight boost in Re compared to the baseline case. This is usually done by rising pumping power, although it can be done by reducing the effective hydraulic diameter, either through surface roughness or not, or by changing the fluid. For this method, the idea is to find a roughness geometry in the new Re that does not reduce significantly the heat advected, but has massive reduction in dissipation rate –observed in $Re = 8000$, with $\overline{f}_f = 0.0357$– and compare the performance with the original Re.
Regardless, the study of parametric regions on such a big dataset gives qualitative reports on flow mechanisms, the effect –positive or negative– that the shape parameters have on the flow and how that leads to optimisation, which differs for laminar and turbulence regimes. This serves as a basis for a systematic study on geometric parametrisation for surface roughness, where further exploration into areas of potential optimisation found here –since optimum values were measured at the boundaries– expands on the parametric analysis. The insight this puts forward has the potential to be used in cooling systems' development and efficiency studies, particularly in fields where optimisation or heat transfer enhancement is essential, like in nuclear fusion research. The open-source Hammerhead software fills in this necessity, with which further exploration in flow and heat transfer over rough surfaces can be carried out with low user cost and minor adjustments to the source code if additional shape and flow parameters are required. Furthermore, the metrics used in surface roughness interactions with fluid flow differ in each study –heat coefficients, Nu, advected heat, mean temperature, dissipation rate, drag, pressure drop, friction factor–, which promotes the usage of Hammerhead due to its modular capabilities and makes the analysis within this research relevant for overall energy efficiency.

8.3 Future work

The thesis opened up a number of prospective research directions introduced below.

- This study needs to be extended to a three-dimensional domain, to exhibit the cross-stream thermal and hydraulic behaviour of flow interacting with rough surfaces, especially within transition and turbulent regimes. This would help study the mechanisms seen in literature [1, 23, 29, 32, 91, 98, 102, 104], where shark riblets have been found to lift the eddies formed in the viscous sublayer in continuous, stream-wise biomimetic configurations; as well as provide insight on turbulent mixing enhanced heat transfer. The $h-$refinement verification of the laminar regime for flow in an axi-symmetric domain

is in Appendix C.3 as an example of this problem extended towards three-dimensions, maintaining the double harmonic function as the parametric study.

- Further research on higher Re is required to fully confirm the flow mass and heat transfer optimisation within the turbulent regimes. This can be done by either refining the mesh or introducing wall functions for Re values where $5 \leq y^+ \leq 30$ is true.
- Explore different Pr values and include these in the parametric research. Several studies [29, 102, 104] report better enhancement in higher Pr, giving this dimensionless parameter a heavy influence in the thermal interaction of flow over rough surfaces.
- Approach the high fidelity modelling with higher order schemes, Large Eddy Simulations or Direct Numerical Simulations to compute flow and thermal behaviour more accurately.
- Expand the shape parametric space towards different amplitudes and wavenumbers. This is required to observe the momentum and heat transfer optimisation outside of the studied region within laminar or turbulent regimes, since the optimal values were found on the boundaries. Accordingly, the complexity of the harmonic function can be increased to adjust to the three-dimensional behaviour of flow and produce channel-like domains and shark skin mimicry surfaces, which have been observed in literature to reduce pressure loss while increasing the wetted surface area [1, 23, 29, 32, 91, 98, 102, 104].
- Assessment of Neural Network architectures with higher neuron counts to increase the accuracy of models with complex input dimension problems $\boldsymbol{\omega}_3$ and $\boldsymbol{\omega}_5$. Overall, the study of low correlation in high-dimension input problems needs to be address for the currently studied models with different architectures, as well as new surrogate models [55, 65, 68, 70, 71, 73, 125, 139–141].
- Finally, generalisation of the Hammerhead framework towards ML-aided shape parameter optimisation based on high fidelity data for any engineering application, where the framework is agnostic of the problem, software and data shape.

APPENDIX A

Mathematical foundations

"Calculus is the outcome of a dramatic intellectual struggle which has lasted for twenty-five hundred years."

— Richard Courant, 1888-1972

A.1 Vector and tensor calculus

Let $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$ be the second-order orthonormal Euclidean basis in $\mathbb{R}^3$, such that generic vector $\boldsymbol{v} \in \mathbb{R}^3$, second-order tensor $\boldsymbol{\tau} \in \mathbb{R}^3 \times \mathbb{R}^3$ can be expressed as, assuming Einstein's notation

$$\boldsymbol{v} = v_i \boldsymbol{e}_i \,, \qquad \boldsymbol{\tau} = \tau_{ij} \boldsymbol{e}_i \otimes \boldsymbol{e}_j \tag{A.1}$$

The product of tensors encompasses a number of operations: scalar or dot product $\cdot$, vector or cross product $\times$, outer or dyadic product $\otimes$ and contraction product $:$. To introduce these operations, consider vectors $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$, and second-order tensors $\boldsymbol{A}$, $\boldsymbol{B}$, then the definitions above are used to compute

$$\boldsymbol{u} \cdot \boldsymbol{v} = u_i \, v_i \,, \tag{A.2a}$$
$$\boldsymbol{u} \times \boldsymbol{v} = \varepsilon_{ijk} \, u_j \, v_k \, \boldsymbol{e}_i \,, \tag{A.2b}$$
$$(\boldsymbol{u} \otimes \boldsymbol{v})_{ij} = u_i \, v_j \, \boldsymbol{e}_i \otimes \boldsymbol{e}_j \,, \tag{A.2c}$$
$$(\boldsymbol{u} \times \boldsymbol{v}) \, \boldsymbol{w} = \varepsilon_{ijk} \, u_j \, v_k \, w_i \,, \tag{A.2d}$$
$$\boldsymbol{A} : \boldsymbol{B} = A_{ij} \, B_{ij} \,, \tag{A.2e}$$
$$\boldsymbol{A}\boldsymbol{u} = A_{ij} \, u_j \, \boldsymbol{e}_i \,, \tag{A.2f}$$
$$(\boldsymbol{A} \otimes \boldsymbol{B})_{ijkl} = A_{ij} \, B_{kl} \, \boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l \,, \tag{A.2g}$$
$$\mathrm{tr}\,(\boldsymbol{A}) = \boldsymbol{A} : \boldsymbol{I} = A_{ij} \, \delta_{ij} \,. \tag{A.2h}$$

where ε_{ijk}, $\boldsymbol{I}$ and δ_{ij} are the Levi-Civita permutation coefficient, identity second-order tensor and Kronecker delta given by

$$\varepsilon_{ijk} = \begin{cases} +1 & \text{if } (i,j,k) \text{ is } (1,2,3),\ (2,3,1) \text{ or } (3,1,2)\,, \\ -1 & \text{if } (i,j,k) \text{ is } (3,2,1),\ (1,3,2) \text{ or } (2,1,3)\,, \\ 0 & \text{if } i=j \text{ or } j=k,\ k=i\,. \end{cases} \tag{A.3a}$$

$$\boldsymbol{I} = \delta_{ij} \boldsymbol{e}_i \otimes \boldsymbol{e}_j \tag{A.3b}$$

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j\,, \\ 1 & \text{if } i = j\,. \end{cases} \tag{A.3c}$$

Some properties of the operations above are presented in the following set of equations

$$\boldsymbol{u} \cdot (\boldsymbol{A}\boldsymbol{v}) = \boldsymbol{v} \cdot \left(\boldsymbol{A}^T \boldsymbol{u}\right) = \boldsymbol{A} : (\boldsymbol{u} \otimes \boldsymbol{v}) \,, \tag{A.4a}$$
$$\boldsymbol{u} \times \boldsymbol{v} = -\boldsymbol{\varepsilon} : (\boldsymbol{u} \otimes \boldsymbol{v}) \,, \tag{A.4b}$$
$$(\boldsymbol{u} \otimes \boldsymbol{v}) \, \boldsymbol{w} = (\boldsymbol{w} \cdot \boldsymbol{v}) \, \boldsymbol{u} \,, \tag{A.4c}$$
$$(\boldsymbol{u} \otimes \boldsymbol{v})^T = (\boldsymbol{v} \otimes \boldsymbol{u}) \,, \tag{A.4d}$$
$$\boldsymbol{A} \, (\boldsymbol{u} \otimes \boldsymbol{v}) = (\boldsymbol{A}\boldsymbol{u} \otimes \boldsymbol{v}) \,, \tag{A.4e}$$

$$(\boldsymbol{u} \otimes \boldsymbol{v})\, \boldsymbol{A} = \left(\boldsymbol{u} \otimes \boldsymbol{A}^T \boldsymbol{v}\right) , \tag{A.4f}$$

$$\mathrm{tr}\,(\boldsymbol{u} \otimes \boldsymbol{v}) = \boldsymbol{u} \cdot \boldsymbol{v} , \tag{A.4g}$$

$$\boldsymbol{A} : \boldsymbol{B} = \mathrm{tr}\left(\boldsymbol{A}^T \boldsymbol{B}\right) = \mathrm{tr}\left(\boldsymbol{B}\boldsymbol{A}^T\right) = \mathrm{tr}\left(\boldsymbol{B}^T \boldsymbol{A}\right) = \mathrm{tr}\left(\boldsymbol{A}\boldsymbol{B}^T\right) . \tag{A.4h}$$

A.2 Differential operators

The *del* operator, denoted by ∇, is a differential operator defined by

$$\nabla = \frac{\partial}{\partial x_i} \boldsymbol{e}_i . \tag{A.5}$$

The *gradient* operator is a vector differential that acts on a scalar field $f : \mathbb{R}^3 \to \mathbb{R}$, to produce a vector field $\nabla f : \mathbb{R}^3 \to \mathbb{R}^3$

$$\nabla f = \frac{\partial f}{\partial x_i} \boldsymbol{e}_i . \tag{A.6}$$

The resulting vector field represents the direction of magnitude of the steepest increase of the scalar field $f(\boldsymbol{x})$ at each point $\boldsymbol{x}$. The gradient can be visualised as a vector field, where all the arrows point in the direction of maximum change. Similarly, the gradient of a vector field $\boldsymbol{v} : \mathbb{R}^3 \to \mathbb{R}^3$ can be defined to produce the second order tensor $\nabla \boldsymbol{v} : \mathbb{R}^3 \to \mathbb{R}^3 \times \mathbb{R}^3$ as

$$\nabla \boldsymbol{v} = \frac{\partial v_j}{\partial x_i} \boldsymbol{e}_j \otimes \boldsymbol{e}_i . \tag{A.7}$$

The *divergence* operator, denoted by $\nabla\cdot$, is a vector differential operator that acts on a vector field $\boldsymbol{v} : \mathbb{R}^3 \to \mathbb{R}^3$ to produce a scalar field $\nabla \cdot \boldsymbol{v} : \mathbb{R}^3 \to \mathbb{R}$ as

$$\nabla \cdot \boldsymbol{v} = \mathrm{tr}\,(\nabla \boldsymbol{v}) = \nabla \boldsymbol{v} : \boldsymbol{I} = \frac{\partial v_i}{\partial x_i} , \tag{A.8}$$

with the use of (A.2h). Geometrically, the divergence measures the extent to which a vector field $\boldsymbol{v}(\boldsymbol{x})$ is *sourcing* or *sinking* at a given point in space. A positive divergence means the vector field is *sourcing*, whereas a negative divergence means it is *sinking*. Similarly, the divergence of a second order tensor field $\nabla \boldsymbol{A} : \mathbb{R}^3 \to \mathbb{R}^3 \times \mathbb{R}^3$ can be defined as $\nabla \cdot \boldsymbol{A} : \mathbb{R}^3 \to \mathbb{R}^3$ given by

$$\nabla \cdot \boldsymbol{A} = \nabla \boldsymbol{A} : \boldsymbol{I} = \frac{\partial A_{ik}}{\partial x_k} \boldsymbol{e}_i . \tag{A.9}$$

The *curl* operator, denoted by $\nabla\times$, is a vector differential operator that acts on a vector field $\boldsymbol{v} : \mathbb{R}^3 \to \mathbb{R}^3$ to produce a vector field $\nabla \times \boldsymbol{v} : \mathbb{R}^3 \to \mathbb{R}^3$ as

$$\nabla \times \boldsymbol{v} = -\boldsymbol{\varepsilon} : \nabla \boldsymbol{v} = \varepsilon_{ijk} \frac{\partial v_j}{\partial x_i} \boldsymbol{e}_k . \tag{A.10}$$

The resulting vector field represents the rotational behaviour of the original vector field $\boldsymbol{v}(\boldsymbol{x})$ at each point $\boldsymbol{x}$ in space. Geometrically, the curl represents the tendency of a vector field to rotate about a given point. The direction and magnitude of the resulting vector field represents

the axis and magnitude of rotation of the original vector field. With the above definitions at hand, it is possible to show that the following useful identities hold

$$\nabla \times (\nabla f) = 0\,, \tag{A.11a}$$
$$\nabla \cdot (\nabla \times \boldsymbol{v}) = 0\,, \tag{A.11b}$$
$$\nabla (f\boldsymbol{v}) = f\nabla \boldsymbol{v} + \boldsymbol{v} \otimes \nabla f\,, \tag{A.11c}$$
$$\nabla \cdot (f\boldsymbol{v}) = f\nabla \cdot \boldsymbol{v} + \boldsymbol{v} \cdot \nabla f\,, \tag{A.11d}$$
$$\nabla (\boldsymbol{v} \cdot \boldsymbol{u}) = (\nabla \boldsymbol{v})^T \boldsymbol{u} + (\nabla \boldsymbol{u})^T \boldsymbol{v}\,, \tag{A.11e}$$
$$\nabla \cdot (\boldsymbol{v} \otimes \boldsymbol{u}) = \boldsymbol{v}\nabla \cdot \boldsymbol{u} + \boldsymbol{v} \cdot (\nabla \boldsymbol{u})\,, \tag{A.11f}$$
$$\nabla \cdot \left(\boldsymbol{A}^T \boldsymbol{v}\right) = \boldsymbol{A} : \nabla \boldsymbol{v} + \boldsymbol{v} \cdot \nabla \cdot \boldsymbol{A}\,, \tag{A.11g}$$
$$\nabla \cdot (f\boldsymbol{A}) = f\nabla \cdot \boldsymbol{A} + \boldsymbol{A}\nabla f\,. \tag{A.11h}$$

which state that the gradient of a field is always curl free –i.e. irrotational or conservative– and the curl of a field is always divergence free –i.e. solenoidal–.

A.3 Field theorems

Let us consider a domain Ω bounded by the surface boundary $\partial\Omega$ and outward unit normal vector $\boldsymbol{n}_{\partial\Omega}(x)$. For scalar and vector fields $\phi : \mathbb{R}^3 \to \mathbb{R}$ and $\boldsymbol{v} : \mathbb{R}^3 \to \mathbb{R}^3$, respectively, **Gauss' theorem** establishes that within a closed surface, the gradient of a field integrated over its volume can be obtained from the field's net flux integral through the boundary in the direction of the outward normal

$$\int_\Omega \nabla f \; d\Omega = \int_{\partial\Omega} f\boldsymbol{n}_{\partial\Omega} \; dS\,, \qquad \int_\Omega \nabla \boldsymbol{v} \; d\Omega = \int_{\partial\Omega} \boldsymbol{v} \otimes \boldsymbol{n}_{\partial\Omega} \; dS\,. \tag{A.12}$$

The **Divergence theorem** states that within a closed surface, the divergence of a field integrated over its volume can be related to the field's net flux integral projected on the bounding surface in the direction of the outward normal

$$\int_\Omega \nabla \cdot \boldsymbol{v} \; d\Omega = \int_{\partial\Omega} \boldsymbol{v} \cdot \boldsymbol{n}_{\partial\Omega} \; dS\,, \qquad \int_\Omega \nabla \cdot \boldsymbol{\tau} \; d\Omega = \int_{\partial\Omega} \boldsymbol{\tau}\boldsymbol{n}_{\partial\Omega} \; dS\,. \tag{A.13}$$

Let $\phi = \phi(t, \boldsymbol{x})$ be any property of the fluid in an time-dependent moving control volume $\Omega^*(t)$, with an integral form

$$\int_{\Omega^*(t)} \phi(t, \boldsymbol{x}) \; d\Omega\,, \tag{A.14}$$

where $\boldsymbol{x}$ is a point in the control volume. The **Reynold's transport theorem** states that the derivative of this integral can be expressed as the variation of $\phi = \phi(t, \boldsymbol{x})$ in time integrated over its volume and the net flux integral through the bounding surface in the direction of the outward normal

$$\frac{d}{dt} \int_{\Omega^*(t)} \phi(t) \; d\Omega = \int_\Omega \frac{\partial \phi}{\partial t} \; d\Omega + \int_{\partial\Omega} \phi \boldsymbol{v} \cdot \boldsymbol{n}_{\partial\Omega} \; dS\,, \tag{A.15}$$

where $\boldsymbol{v}$ represents the underlying velocity. This can be transformed into a volume integral with the Gauss theorem

$$\begin{aligned} \frac{d}{dt}\int_{\Omega(t)} \phi\,(t)\; d\Omega &= \int_{\Omega} \left(\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \boldsymbol{v}) \right) d\Omega \\ &= \frac{d}{dt}\int_{\Omega} \phi \, d\Omega + \int_{\Omega} \nabla \cdot (\phi \boldsymbol{v})\; d\Omega \,. \end{aligned} \tag{A.16}$$

APPENDIX B

Fundamentals of fluid mechanics

"The purpose of computing is insight, not numbers."

— C. Hastings Jr, 1955

B.1 Dimensionless numbers

The following are some useful dimensionless numbers that result from writing the conservation equations in non-dimensional forms. These are used in parametric studies for engineering problems [45] since they govern the natural behaviour of flow. The **Reynolds number** Re is a measure of the relative ratio of advection to diffusion momentum fluxes

$$Re = \frac{\rho \overline{v} L}{\mu}\,, \tag{B.1}$$

where $\overline{v}$ is the free-stream velocity of the flow and L a reference length, typically stream-wise length of the geometry in external flows or hydraulic diameter in internal flows. For flow in a pipe of diameter D, Re conveys the flow regime as laminar $Re_D \leq 2300$ and turbulent $Re_D \geq 2900$. The Prandtl number Pr defines the ratio of momentum to thermal diffusivity

$$Pr = \frac{\mu \varsigma_p}{k} = \frac{\nu}{\alpha}\,, \tag{B.2}$$

where $\alpha = k/\rho$, ν, k, ς_p and μ are the fluid's thermal diffusivity, kinematic viscosity, thermal conductivity, specific heat at constant pressure and dynamic viscosity, respectively. This number describes the ratio in size of the thermal to hydrodynamic boundary layers, where $Pr < 1$ indicates the thermal layer is larger than the hydrodynamic boundary layer, and $Pr = 1$ means both layers coincide. The **Mach number** M defines the velocity ratio of the speed of sound and an object moving through fluid

$$M = \frac{|\boldsymbol{v}|}{a}\,, \tag{B.3}$$

where $|\boldsymbol{v}|$ is the local magnitude of velocity and a the speed of sound. The **dimensionless wall distance** y^+ is computed as

$$y^+ = \frac{v_* y}{\nu}\,, \tag{B.4}$$

where $v_* = \sqrt{\tau_w/\rho}$ is the friction velocity at the nearest wall, y is the distance to the nearest wall, ν is the local kinematic viscosity, τ_w is the shear stress at the wall and ρ is the density of the fluid. This is often used in boundary layer theory, the law of the wall and to compute the necessary grid spacing close to the wall in a mesh to maintain the boundary layer definition.

B.2 Reynolds Averaged Navier-Stokes

Considering ϕ and φ as flow field variables in time t and position $\boldsymbol{x}$, with a mean or *average value* $\overline{\phi}$ and $\overline{\varphi}$, and turbulent value or *fluctuating components* ϕ' and φ', respectively, so that the following additive decomposition applies

$$\phi = \overline{\phi} + \phi'\,, \qquad \varphi = \overline{\varphi} + \varphi'\,. \tag{B.5}$$

Depending on the dependent variable interval where the averaging is performed, a different computation of $\overline{\phi}$ is required. Some relevant types of averaging are presented in Table B.1.

Time averaging	$\overline{\phi}(\boldsymbol{x}) = \lim_{T\to\infty} \frac{1}{T} \int_t^{t+T} \phi(\boldsymbol{x}, t)\, dt$
Space averaging	$\overline{\phi}(t) = \lim_{\Omega\to\infty} \frac{1}{\Omega} \int_\Omega \phi(\boldsymbol{x}, t)\, d\Omega$

Table B.1: Averaging computation of the dependent flow field variable $\overline{\phi}$ depending on the independent variable. In time averaging, T represents the time interval, associated with steady turbulent flows. For space averaging, Ω represents the volume interval, usually analogous to homogenous turbulence.

The Reynolds-averaged Navier-Stokes (RANS) equations are derived through time averaging. Some rules needed to derive these equations include [45]

$$\begin{aligned} \overline{\phi'} &= 0, \\ \overline{\phi} &= \overline{\overline{\phi}}\,, \\ \overline{\nabla\phi} &= \nabla\overline{\phi}\,, \\ \overline{\phi+\varphi} &= \overline{\phi} + \overline{\varphi}\,, \\ \overline{\overline{\phi}\varphi} &= \overline{\phi}\,\overline{\varphi}\,, \\ \overline{\overline{\phi}\varphi'} &= 0\,, \\ \overline{\phi\varphi} &= \overline{\phi}\,\overline{\varphi} + \overline{\phi'\varphi'}\,. \end{aligned} \tag{B.6}$$

B.3 Turbulence modelling

Kolmogorov postulated the "energy cascade" concept, describing turbulence as composed of different sized eddies[1] possessing a certain amount of energy depending on their dimensions [45]. The largest eddies are known as the integral length scales, proportional to the size of the geometry involved. These break up in a chain process where they transfer their energy until the smallest possible size is reached, the Kolmogorov scale. These have a characteristic length and time scales, η and t_η, respectively [45]

$$\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{\frac{1}{4}} \qquad t_\eta = \left(\frac{\nu}{\varepsilon}\right)^{\frac{1}{2}}, \tag{B.7}$$

where ν is the molecular kinematic viscosity and ε the average rate of dissipation of turbulent kinetic energy. Based on (B.7) the direct numerical solution of the Navier-Stokes equation for turbulent flows is limited by a time step $\Delta t < t_\eta$, and an element size $\Delta x < \eta$, resulting in a computationally demanding approach called Direct Numerical Simulation (DNS) [45]. To reduce the large computational cost of DNS, statistical averaging can be applied to approximate the random fluctuations of turbulent flows. The Reynolds Averaged Navier-Stokes (RANS) equations are time-averaged, decomposing the flow variables into time-mean value and

[1] Swirling of a fluid, developing reverse current, usually associated with turbulent flows.

fluctuating components to substitute into the original equations. Nevertheless, the non-linear terms in momentum conservation lead to the appearance of momentum fluxes [45, 142]. The RANS approach intents to model all scales of turbulent flow to address this closure problem with the help of averaging rules in Section B.2

$$\phi = \overline{\phi} + \phi' \,. \tag{B.8}$$

By substituting (B.8) into the incompressible continuity, momentum and energy equations, and assuming a Newtonian fluid, the definitions transform in [45, 143]

$$\nabla \cdot (\rho \overline{\boldsymbol{v}}) = 0 \,, \tag{B.9}$$

$$\frac{\partial}{\partial t}(\rho \overline{\boldsymbol{v}}) + \nabla \cdot (\rho \overline{\boldsymbol{v}} \otimes \overline{\boldsymbol{v}}) = -\nabla \overline{p} + \nabla \cdot (\overline{\boldsymbol{\tau}} - \rho \overline{\boldsymbol{v}' \otimes \boldsymbol{v}'}) + \overline{\rho \boldsymbol{g}} \,, \tag{B.10}$$

$$\frac{\partial}{\partial t}(\rho \overline{E}) + \nabla \cdot (\rho \overline{E} \overline{\boldsymbol{v}}) = -\nabla \cdot \overline{q_s} - \nabla \cdot (\overline{p \boldsymbol{v}} + \overline{p' \boldsymbol{v}'}) + \nabla \cdot (\overline{\boldsymbol{\tau} \boldsymbol{v}} - \rho \overline{E' \boldsymbol{v}'}) + \rho (\overline{\boldsymbol{g} \cdot \boldsymbol{v}}) + \overline{q_\Omega} \,, \tag{B.11}$$

where the averaged total and the turbulent kinetic energy, $\overline{E}$ and k, respectively, are given by [143]

$$\overline{E} \equiv \overline{e} + \frac{1}{2} \overline{\boldsymbol{v}} \cdot \overline{\boldsymbol{v}} + k \,, \qquad k \equiv \frac{1}{2} \overline{\boldsymbol{v}' \cdot \boldsymbol{v}'} \,. \tag{B.12}$$

The additional averaged products of the fluctuating components introduce the Reynolds stress tensor $\boldsymbol{\tau}^R = -\rho \overline{\boldsymbol{v}' \otimes \boldsymbol{v}'}$, and turbulent heat fluxes $\boldsymbol{q}^R = -\rho \overline{E' \boldsymbol{v}'}$, into the momentum and energy equations, respectively [45]. Based on the Boussinesq hypothesis, the non-linear fluctuating components are expressed as [45, 142]

$$\boldsymbol{\tau}^R = 2 \mu_t \boldsymbol{d} - \frac{2}{3} (\rho k + \mu_t \mathrm{tr} \boldsymbol{d}) \boldsymbol{I} \,, \tag{B.13}$$

which for incompressible flows it reduces to

$$\boldsymbol{\tau}^R = 2 \mu_t \boldsymbol{d} - \frac{2}{3} \rho k \boldsymbol{I} \,, \tag{B.14}$$

where $\boldsymbol{d} = \frac{1}{2}(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T)$, μ_t is the turbulent eddy viscosity. For incompressible flows the term $-\frac{2}{3}\rho k I$ it becomes [45]

$$p \leftarrow p + \frac{2}{3} \rho k I \,. \tag{B.15}$$

Therefore, μ_t and k, are computed by using turbulence modelling equations. These intend to express μ_t in terms of a velocity and length scales, $\sqrt{k}$ and l, such that [45]

$$\mu_t = \rho l \sqrt{k}. \tag{B.16}$$

Likewise, the turbulent thermal fluxes are calculated in analogy with Fourier's law [45]

$$\boldsymbol{q}^R = \alpha_t \nabla T \,, \tag{B.17}$$

where α_t is the turbulent thermal diffusivity, also evaluated by the turbulence model of choice. The most commonly used are two-equation models, amongst which the k - ε model of Jones and Launder [45, 142] and the k - ω model of Wilcox [45, 142] are most popular and have undergone modifications and improvements.

B.4 Incompressible model of specific heat capacity

With the assumption of incompressibility, the definition of specific heat capacity at constant pressure and constant volume take the same value ($\varsigma_p = \varsigma_v$). To prove this concept, we define ς_p and ς_v from Maxwell's relations as

$$\varsigma_p = \left.\frac{dh}{dT}\right|_p , \tag{B.18}$$

$$\varsigma_v = \left.\frac{de}{dT}\right|_V , \tag{B.19}$$

Then to calculate the derivative of the definition of ς_p from (B.18) we write

$$\begin{aligned} \varsigma_p = \left.\frac{dh}{dT}\right|_p &= \left.\frac{de}{dT}\right|_p + p \underbrace{\left.\frac{dV}{dT}\right|_p}_{=V\alpha} + V \underbrace{\left.\frac{dp}{dT}\right|_p}_{=0} \\ &= \left.\frac{de}{dT}\right|_p + pV\alpha , \end{aligned} \tag{B.20}$$

where $\alpha := \frac{1}{V}\left.\frac{dV}{dT}\right|_p$ is the expansion coefficient. By assuming $e\,(V,T)$

$$\begin{aligned} \left.\frac{de}{dT}\right|_p &= \left.\frac{de}{dV}\right|_T \underbrace{\left.\frac{dV}{dT}\right|_p}_{=V\alpha} + \underbrace{\left.\frac{de}{dT}\right|_V}_{=\varsigma_v} \left.\frac{dT}{dT}\right|_p \\ &= \left.\frac{de}{dV}\right|_T V\alpha + \varsigma_v , \end{aligned} \tag{B.21}$$

and by assuming $e\,(V,S)$

$$\begin{aligned} \left.\frac{de}{dT}\right|_p &= \left[T \left.\frac{dS}{dV}\right|_T - p \underbrace{\left.\frac{dV}{dV}\right|_T}_{=1} \right] V\alpha + \varsigma_v \\ &= \left[T \left.\frac{dS}{dV}\right|_T - p \right] V\alpha + \varsigma_v \\ &= \left[T \underbrace{\left.\frac{dV}{dT}\right|_p}_{:=\alpha V} \underbrace{\left.\frac{dp}{dV}\right|_T}_{:=\frac{1}{\beta V}} - p \right] V\alpha + \varsigma_v \\ &= \left[T\frac{\alpha}{V} - p \right] V\alpha + \varsigma_v , \end{aligned} \tag{B.22}$$

where $\beta = \frac{1}{V}\frac{dV}{dp}\Big|_T$ is the compressibility of the fluid, and $\frac{\alpha}{\beta} = \frac{dp}{dT}\Big|_V = \frac{dS}{\mathrm{d}V}\Big|_T$ according to Maxwell's relations [84, 85]. Thus,

$$\varsigma_p - \varsigma_v = \frac{\alpha^2}{\beta} TV \,. \tag{B.23}$$

By assuming incompressibility, $\alpha = 0$, and hence $\varsigma_p = \varsigma_v$.

APPENDIX C

High fidelity model additional material

"Dedication is a talent all on its own."

— Full Metal Alchemist: Brotherhood

C.1 k - ω SST turbulence model

The **Shear Stress Transport (SST)** model [45, 134] is a modification of the Baseline (BSL) model developed by Menter [134], which combines the $k-\varepsilon$ and $k-\omega$ to take advantage of the performance near the boundary layer edge and free stream region from the first, as well as the robustness for near wall surfaces flows and weak adverse pressure situations of the latter. A brief exposition of these three models is presented to understand the overall operation of the $k-\omega$ SST. The $\boldsymbol{k-\varepsilon}$ model is based in the Boussinesq approximation formulating the μ_t and k_t as

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon}\,, \tag{C.1}$$

$$\alpha_t = \frac{C_p \mu_t}{Pr_t}\,, \tag{C.2}$$

where ε is the dissipation rate of turbulence kinetic energy per unit mass due to viscous stress [45]

$$\varepsilon = \frac{1}{2}\frac{\mu}{\rho}\overline{(\nabla \boldsymbol{v}' + (\nabla \boldsymbol{v}')^T) : (\nabla \boldsymbol{v}' + (\nabla \boldsymbol{v}')^T)}\,, \tag{C.3}$$

This model computes k and ε using

$$\frac{\partial}{\partial t}(\rho k) + \nabla \cdot (\rho k \boldsymbol{v}) = \nabla \cdot (\mu_{eff,k} \nabla k) + (\boldsymbol{\tau^R} : \nabla \boldsymbol{v}) - \rho\varepsilon\,, \tag{C.4}$$

$$\frac{\partial}{\partial t}(\rho \varepsilon) + \nabla \cdot (\rho \varepsilon \boldsymbol{v}) = \nabla \cdot (\mu_{eff,\varepsilon} \nabla \varepsilon) + C_{\varepsilon 1}\frac{\varepsilon}{k}(\boldsymbol{\tau^R} : \nabla \boldsymbol{v}) - C_{\varepsilon 2}\rho\frac{\varepsilon^2}{k}\,, \tag{C.5}$$

where

$$\mu_{eff,k} = \mu + \frac{\mu_t}{\sigma_k}\,, \qquad \mu_{eff,\varepsilon} = \mu + \frac{\mu_t}{\sigma_\varepsilon}\,, \tag{C.6}$$

and the model constants are assigned with the values

$$C_{\varepsilon 1} = 1.44\,,\ C_{\varepsilon 2} = 1.92\,,\ C_\mu = 0.09\,,\ \sigma_k = 1.0\,,\ \sigma_\varepsilon = 1.3\,,\ Pr_t = 0.9\,. \tag{C.7}$$

The $\boldsymbol{k-\omega}$ model computes the rate at which turbulence kinetic energy is converted into internal thermal energy per unit volume and time ω instead of ε due to its capacity of predicting separated flows. This specific turbulence dissipation ω is defined by [45]

$$\omega = \frac{\varepsilon}{C_{mu} k}\,. \tag{C.8}$$

The model formulates the μ_t and k_t as

$$\mu_t = \rho \frac{k}{\omega}\,, \tag{C.9}$$

$$\alpha_t = \frac{\mu_t}{Pr_t}\,, \tag{C.10}$$

The conservation equations are written as [45]

$$\frac{\partial}{\partial t}(\rho k) + \nabla \cdot (\rho k \boldsymbol{v}) = \nabla \cdot (\mu_{eff,k} \nabla k) + (\boldsymbol{\tau^R} : \nabla \boldsymbol{v}) - \rho\beta^* k\omega\,, \tag{C.11}$$

$$\frac{\partial}{\partial t}(\rho\omega) + \nabla\cdot(\rho\omega\boldsymbol{v}) = \nabla\cdot(\mu_{eff,\omega}\nabla\omega) + C_{\alpha 1}\frac{\omega}{k}(\boldsymbol{\tau^R}:\nabla\boldsymbol{v}) - C_{\beta 1}\rho\omega^2\,, \tag{C.12}$$

where

$$\mu_{eff,k} = \mu + \frac{\mu_t}{\sigma_{k_1}}\,, \qquad \mu_{eff,\omega} = \mu + \frac{\mu_t}{\sigma_{\omega_1}}\,, \tag{C.13}$$

and the model constants are assigned with the values

$$C_{\alpha 1} = 5/9\,,\ C_{\beta 1} = 0.075\,,\ \beta^* = 0.09\,,\ \sigma_{k_1} = 2.0\,,\ \sigma_{\omega_1} = 2.0\,,\ Pr_t = 0.9\,. \tag{C.14}$$

The **Baseline (BSL)** model multiplies the original $k-\omega$ equations with a blending function and the transformation of the $k-\varepsilon$ model to a $k-\omega$ formulation given by [45, 134]

$$\frac{\partial}{\partial t}(\rho k) + \nabla\cdot(\rho k\boldsymbol{v}) = \nabla\cdot(\mu_{eff,k}\nabla k) + (\boldsymbol{\tau^R}:\nabla\boldsymbol{v}) - \rho\beta^* k\omega\,, \tag{C.15}$$

$$\frac{\partial}{\partial t}(\rho\omega) + \nabla\cdot(\rho\omega\boldsymbol{v}) = \nabla\cdot(\mu_{eff,\omega}\nabla\omega) + C_{\alpha 2}\frac{\omega}{k}(\boldsymbol{\tau^R}:\nabla\boldsymbol{v}) - C_{\beta 2}\rho\omega^2 + 2\sigma_{\omega_2}\frac{\rho}{\omega}\nabla k\cdot\nabla\omega\,, \tag{C.16}$$

where

$$\mu_{eff,k} = \mu + \frac{\mu_t}{\tilde{\sigma}_k}\,, \qquad \mu_{eff,\omega} = \mu + \frac{\mu_t}{\tilde{\sigma}_\omega}\,, \tag{C.17}$$

The model uses the equations of the $k-\omega$ for μ_t and α_t given by (C.9) and (C.10), respectively, while the equation used for ω is

$$\frac{\partial}{\partial t}(\rho\omega) + \nabla\cdot(\rho\omega\boldsymbol{v}) = \nabla\cdot(\mu_{eff,\omega}\nabla\omega) + \tilde{C}_\alpha\frac{\omega}{k}(\boldsymbol{\tau^R}:\nabla\boldsymbol{v}) - C_\beta\rho\omega^2 + 2(1-F_1)\sigma_{\omega_2}\frac{\rho}{\omega}\nabla k\cdot\nabla\omega\,, \tag{C.18}$$

with the constants are assigned with the values

$$\begin{aligned} C_{\alpha 1} = 0.5976\,,\ C_{\beta 1} = 0.075\,,\ \beta^* = 0.09\,,\ \sigma_{k_1} = 2.0\,,\ \sigma_{\omega_1} = 2.0\,,\\ C_{\alpha 2} = 0.4404\,,\ C_{\beta 2} = 0.0828\,,\ \sigma_{k_2} = 1.0\,,\ \sigma_{\omega_2} = 0.856\,,\ Pr_t = 0.9\,. \end{aligned} \tag{C.19}$$

where F_1 is the blending function in the form of

$$F_1 = tan(\gamma_1^4)\,, \tag{C.20}$$

where the solution depends on the variables and on the distance $d_\perp$

$$\begin{aligned} \gamma_1 &= \min\left(\max\left(\frac{\sqrt{k}}{\beta^*\omega(d_\perp)}, \frac{500\boldsymbol{v}}{(d_\perp)^2\omega}\right), \frac{4\rho\sigma_\omega k}{CD_{k\omega}(d_\perp^2)}\right)\,,\\ CD_{k\omega} &= \max\left(2\rho\sigma_{\omega_2}\frac{1}{\omega}\nabla k\cdot\nabla\omega,\ 10^{-10}\right)\,. \end{aligned} \tag{C.21}$$

The first modification to the BSL yielded by the $\boldsymbol{k-\omega}$ **SST** model is to satisfy the Bradshaw assumption. This states that the boundary layer's principal shear stress and turbulent kinetic energy are linearly related [45, 134]:

$$\boldsymbol{\tau}_{xy} = \rho a_1 k\,, \tag{C.22}$$

while the principal shear stress for two-equation models is computed by

$$\boldsymbol{\tau}_{xy} = \mu_t \Omega = \rho \sqrt{\frac{\text{Production of } k}{\text{Dissipation of } k}} a_1 k \,, \tag{C.23}$$

where Ω is the vorticity. The turbulent viscosity μ_t is modified due to this ratio of production to dissipation of kinetic energy, since in flows with adverse pressure gradient it can be much larger than one, violating Broadshaw's hypothesis [45, 134]

$$\mu_t = \frac{\rho a_1 k}{\max(a_1 \omega,\, \sqrt{2} S_t F_2)} \,, \tag{C.24}$$

where $a_1 = 0.31$, S_t is the magnitude of the strain rate defined by

$$S_t = \frac{1}{2}(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T) \,, \tag{C.25}$$

and F_2 is given by

$$F_2 = \tanh(\gamma_2^2) \text{with } \gamma_2 = \max\left(2\frac{\sqrt{k}}{\beta^* \omega (d_\perp)}, \frac{500\boldsymbol{v}}{(d_\perp)^2 \omega} \right) . \tag{C.26}$$

The same blending technique as the BSL model is adopted. The second modification is to the term of turbulence kinetic energy production $P_k = (\boldsymbol{\tau}^{\boldsymbol{R}} : \nabla \boldsymbol{v})$ in the k equation, replaced by

$$P_k = \min(\boldsymbol{\tau}^{\boldsymbol{R}} : \nabla \boldsymbol{v},\, C_1 \varepsilon) \,, \tag{C.27}$$

where ε is obtained from (C.8), and the blending function F_1 is calculated as the BSL model via (C.20) and (C.21) using the following model constants [45, 134]:

$$\begin{aligned} C_{\alpha 1} = 0.5532 \,,\, C_{\beta 1} = 0.075 \,,\, \beta^* = 0.09 \,,\, \sigma_{k_1} = 2.0 \,,\, \sigma_{\omega_1} = 2.0 \,,\, C_1 = 10 \,, \\ C_{\alpha 2} = 0.4403 \,,\, C_{\beta 2} = 0.0828 \,,\, \sigma_{k_2} = 1.0 \,,\, \sigma_{\omega_2} = 1.186 \,,\, Pr_t = 0.9 \,. \end{aligned} \tag{C.28}$$

C.2 Classical laminar flow over 2D pipe cross-section

This section presents the solution of laminar flow in a pipe cross-section in 2D for each of the flow variable fields of interest. This flow is virtually considered a cylindrical pipe flow, and analytical data is used to validate the accuracy and convergence of the proposed numerical solution. Initial and boundary conditions are as described in Section 6.2, and the relevant physical and thermal properties of both fluid and solid regions are listed in Table C.1. For the velocity and pressure fields, an analytical solution ϕ_A is computed based on the momentum and Hagen-Poiseuille equations. In the analytical solution of steady-state, incompressible, fully developed laminar flow, the momentum equation (2.18) can be written as

$$\mu \frac{\partial^2 v_x}{\partial y^2} = -\frac{dp}{dx} \,, \tag{C.29}$$

Parameter	Value	Parameter	Value
ρ	1^* [kg/m^2]	T_f/T_s	0.6667
Re	500	k_f/k_s	0.2414
Pr	0.2414	$\varsigma_{p,f}/\varsigma_{p,s}$	5230

Table C.1: Classical laminar flow over 2D pipe cross-section - Physical parameters for the two-dimensional laminar flow over a flat plate example. *The density is considered constant for an incompressible problem.

where pressure is a function of x coordinates only, and v_x is a function of y coordinates only. Now let us introduce the Hagen-Poiseuille law for a laminar flow in a cylindrical pipe of constant cross-section

$$\frac{dp}{dx} = \frac{\Delta p}{L} = \frac{8\mu \dot{V}}{\pi r^4} = \frac{2\mu v_{x,max}}{r^2} = G\,, \tag{C.30}$$

where $\dot{V} = \overline{v}_x \frac{A}{2}$, is the volumetric flow rate, A the area of pipe entrance[1] and $\overline{v}_x = \frac{1}{2} v_{x,max}$ the average stream-wise velocity, with $v_{x,max}$ as the maximum velocity at the pipeline centre. When solving (C.29) for v_x by introducing (C.30), it yields

$$v_x\,(y) = -\frac{G}{2\mu} y^2 + c_1 y + c_2\,. \tag{C.31}$$

At $y = 0$, the velocity is at the pipeline centre $v_x(0) = v_{x,max}$ and so $c_2 = v_{x,max}$. At $y = r$ where r is the maximum radius of the cylindrical pipe, $v_x(r) = 0$, and $c_1 = \frac{Gr}{2\mu} - v_{x,max}$. The parabolic velocity profile in laminar flows is then obtained as

$$v_x\,(y) = v_{x,max}\left[1 - \left(\frac{y}{r}\right)^2\right]\,. \tag{C.32}$$

Likewise, when solving for p it yields

$$p\,(x) = -Gx + c\,. \tag{C.33}$$

At $x = L$, $p(L) = 0$, which gives $c = GL$, and the pressure equation then becomes

$$p\,(x) = 2\mu v_{x,max} \frac{(L - x)}{r^2}\,. \tag{C.34}$$

Note that the Hagen-Poiseuille does not quantify the energy equation from (2.39). The validation of heat transfer requires the theory of thermally fully developed laminar flow, used

[1]Since the domain of study is half of the 2D cross-section of a pipe, the volumetric flow rate of half the entrance is considered.

to compare the heat transfer coefficients. To do so, the Nusselt number Nu represents the ratio of total heat transfer to conductive heat transfer at the wall [138]

$$Nu = \frac{hD}{k_f}\,, \tag{C.35}$$

where h is the convective heat transfer coefficient of the flow, $D = 2r$ is the diameter representing the characteristic length of a cylindrical pipe, and k_f is the thermal conductivity of the fluid. Since the coefficient is unknown algebraically, this equation is typically expressed as a function of Pr and Re, and empirical correlations for a variety of geometries arise. The ones used in this example apply to a cylindrical pipe with constant temperature at the wall with either fully developed laminar flow –$Re \leq 2300$– [144]

$$\begin{aligned} Nu &= 3.66 + \frac{0.0668\left(\frac{D}{L}\right) Re\ Pr}{1 + 0.04\left[(D/L)\, Re\ Pr\right]^{2/3}}\,, \\ &\approx 3.66 \quad \text{when } Pr \geq 0.7\,, \\ &\approx 5.78 \quad \text{when } Pr = 0\,, \end{aligned} \tag{C.36}$$

fully developed transition flow –$2300 \leq Re \leq 3200$–

$$\begin{aligned} Nu &= 2.2407a^4 - 29.499a^3 + 142.32a^2 - 292.51a + 219.88\,, \\ a &= \left(\frac{Re}{1000}\right) \quad \text{when } Pr \geq 0.7\,, \end{aligned} \tag{C.37}$$

and lastly fully developed turbulent flow –$Re \geq 3200$–

$$Nu = \frac{\left(\frac{f}{8}\right)(Re - 1000)\, Pr}{1 + 12.7\left(\frac{f}{8}\right)^{1/2}\left(Pr^{2/3} - 1\right)} \quad \text{when } Pr \geq 0.5\,. \tag{C.38}$$

These equations have been developed and verified by Abraham [137], Churchill and Gnielinski [138]. The simulation values computed by (C.35) need to be congruent with the algebraic values of Nu from (C.36), (C.37) and (C.38), and the development of local Nu across the pipe length must follow similar correlations to the profiles from Abraham [137].
The numerical solution is taken from the example on flow over a smooth surface at $Re = 500$ in Section 6.2. The *ideal* mesh spacing in the near wall region is obtained for a dimensionless wall distance $y^+ = 1$ for an $Re = 9400$ with a $\Delta y = 7.14e^{-4}$, since this refinement is the *modular* base grid within *Hammerhead*'s software basis to account for both low to high Re regimes and complex geometry features.
The relative error $\mathrm{Er}_\phi = \frac{\mathrm{abs}(\phi_A - \phi)}{\phi_A}$ for velocity and pressure radial and axial profiles, respectively, are shown in Figure C.1. These show an average error for both variables very close to zero. Figure C.2 shows the expected values of Nu from the empirical correlations alongside the algebraic predictions and numerical results for $Re = \{500,\ 8000\}$. For $Re = 500$, the predicted and simulated values fall within the expected range, while for $Re = 500$ both predicted and simulated values fall under the expected. However, the simulated value is within the predicted and expected, where the expected is computed for $Pr \geq 0.5$, which leads us to believe it

maintains a good level of accuracy. Figure C.3 shows the local Nu developing across the axial dimensionless location x/r, where it looks like $Re = 500$ reaches a maximum value and starts to decay, while $Re = 8000$ continues to rise. $Re = 500$ has a behaviour similar to what Abraham [137] reports, but $Re = 8000$ seems to be thermally underdeveloped flow, which is congruent with the fixed laminar profile at the inlet boundary condition. Overall, the results seem valid for the heat and mass transfer study.

C.3 Axisymmetric flow

Based on the problem posed in Chapter 6.2, an axisymmetric domain Ω consists of a section based on an angular $\theta = 20$ measurement or *wedge* with internal flow Ω_f in contact with a

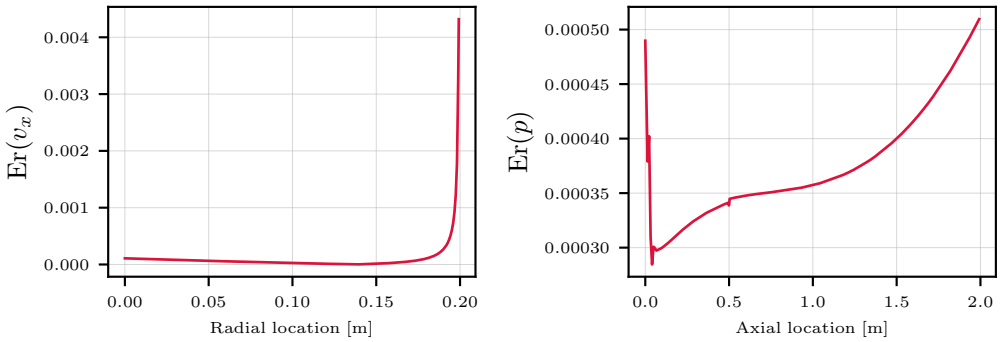


Figure C.1: Classical laminar flow over 2D pipe cross-section - Relative error of –left– streamwise velocity v_x and –right– pressure p fields of the numerical solution at $Re = 500$ compared to the analytical Hagen-Poiseuille flow.

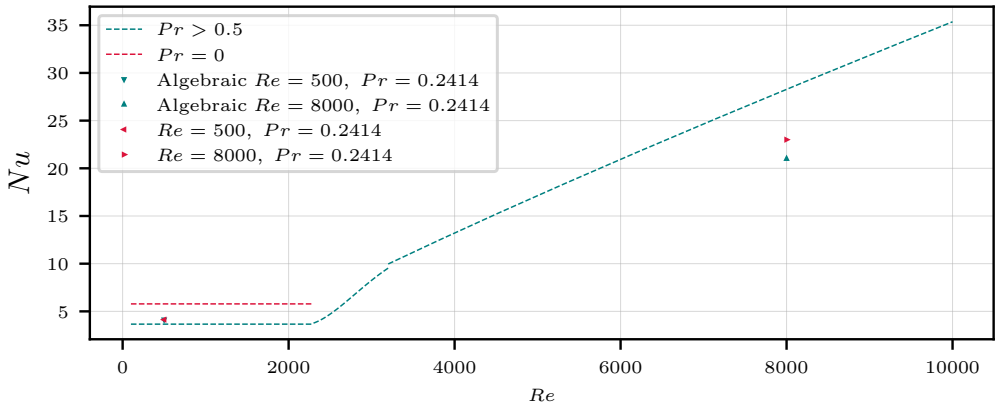


Figure C.2: Classical laminar flow over 2D pipe cross-section - Comparison of fully developed Nusselt numbers for the $Re = 500$ and $Re = 8000$ cases against the empirical correlations from [137].

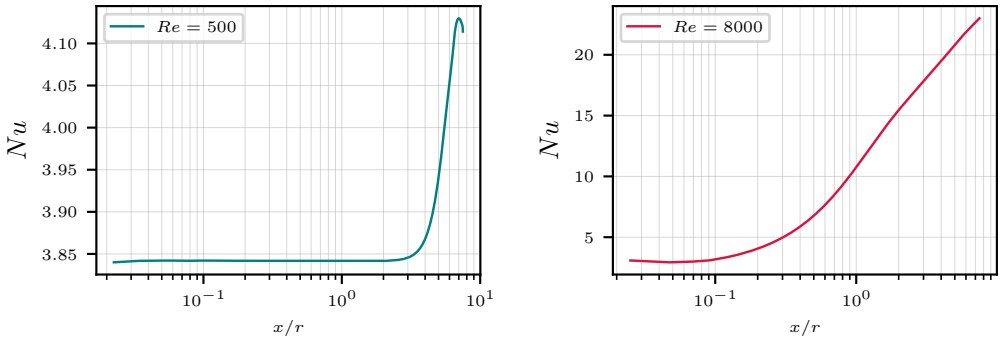


Figure C.3: Classical laminar flow over 2D pipe cross-section - Comparison of fully developed Nusselt numbers for $Re = \{500,\ 8000\}$ against the empirical correlations from [137].

solid arc Ω_s such that $\Omega = \Omega_f \bigcup \Omega_s$. This model is considered *rotationally* symmetric along the stream-wise centre line of the pipe. Even though usually axisymmetric flow is examined in cylindrical coordinates (r, θ, z), OpenFOAM's discretisation is based on Cartesian. The surface in contact with the fluid Γ_{sf} is parametrised using a simplified harmonic function representing the shark skin inspired biomimetic structure described in Section 2.5.2 or in cylindrical coordinates for the axisymmetric problem when the original pipe radius is $r = 0.2$ m

$$y\left(z\right) = r + \sum_{i=1}^{\mathrm{N_h}} \mathrm{A}_i \left(\cos\left(\pi \mathrm{k}_i z\right) + 1\right) , \tag{C.39}$$

where $\mathrm{N_h}$ is the number of harmonics to implement, A_i is the amplitude and k_i is the wavenumber. The parameters described in Chapter 6.2 are used in this example.
The Reynolds $Re = 500,\ 8000$ from Table 6.1 will be used for the laminar and turbulent conditions, respectively. Grid levels are shown in Figure C.5. The grid parameters for each $l-$refinement level mesh and quality are listed in Table C.2 shared across the axisymmetric

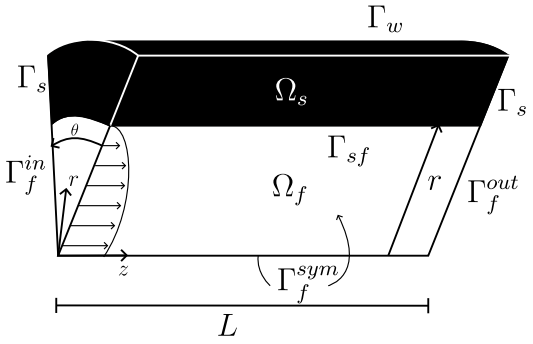


Figure C.4: Flow over *smooth surface* and *micro-structured surface* - Domains of internal flow in a pipe represented by 2D flow over –left– a *smooth* surface plate and over –right– a *micro-structured* surface plate.

$l-$level	h [m]	Δy	$y-$cell count	$x-$cell count	$y-$expansion ratio
3	0.04	$5.53e^{-3}$	25	275	1.02
2	0.02	$2.19e^{-3}$	50	550	1.02
1	0.01	$7.14e^{-4}$	100	1100	1.02
0	0.005	$2.25e^{-4}$	200	2200	1.01

	Maximum non-orthogonality [°]		
$l-$level	$A_1 = 0$ [m]	$A_1 = 0.001$ [m]	$A_1 = 0.01$ [m]
3	0	28.1	43.6
2	0	28.9	44.6
1	0	29.3	44.9
0	0	29.3	45.1

Table C.2: Axisymmetric flow over *smooth* and *micro-structured* surfaces - Mesh refinement parameters and maximum non orthogonality for the two-dimensional flow over *smooth surface* and *micro-structured* flat plate examples.

examples. It is worth noting that $r_h = 2$, and all grids lie well within acceptable maximum non-orthogonal values. The error values $||\varepsilon_r||_{L^2}$ computed are presented against h in Figures C.6 and C.7, while the measured rates of convergence of each flow variable field for the tested shape parameters are provided in Table C.3. From the relation in (6.5) the numerical results have a rate of convergence approaching the expected behaviour of 1st and 2nd numerical schemes, except for temperature in turbulent regimes. The rate of convergence does not seem to decay as much as in the flat plate case with more complex geometries, yet turbulence does impact negatively the rate of convergence.

Amplitude A_1 [m]	q_{v_x}	q_{v_y}	q_{v_z}	q_T	q_p
0.0	1.595	2.4	1.165	1.58	2.38
0.001	1.27	1.75	1.81	1.535	2.3
0.01	1.485	1.95	2.02	1.52	2.785
Amplitude A_1 [m]	q_{v_x}	q_{v_y}	q_{v_z}	q_T	q_p
0.0	1.3	1.715	1.81	0.65	1.4

Table C.3: Axisymmetric flow over *smooth* and *micro-structured* surfaces - Expected and average measured rates of convergence of the numerical solution for each flow variable field within –from top to bottom– laminar and turbulent regimes.

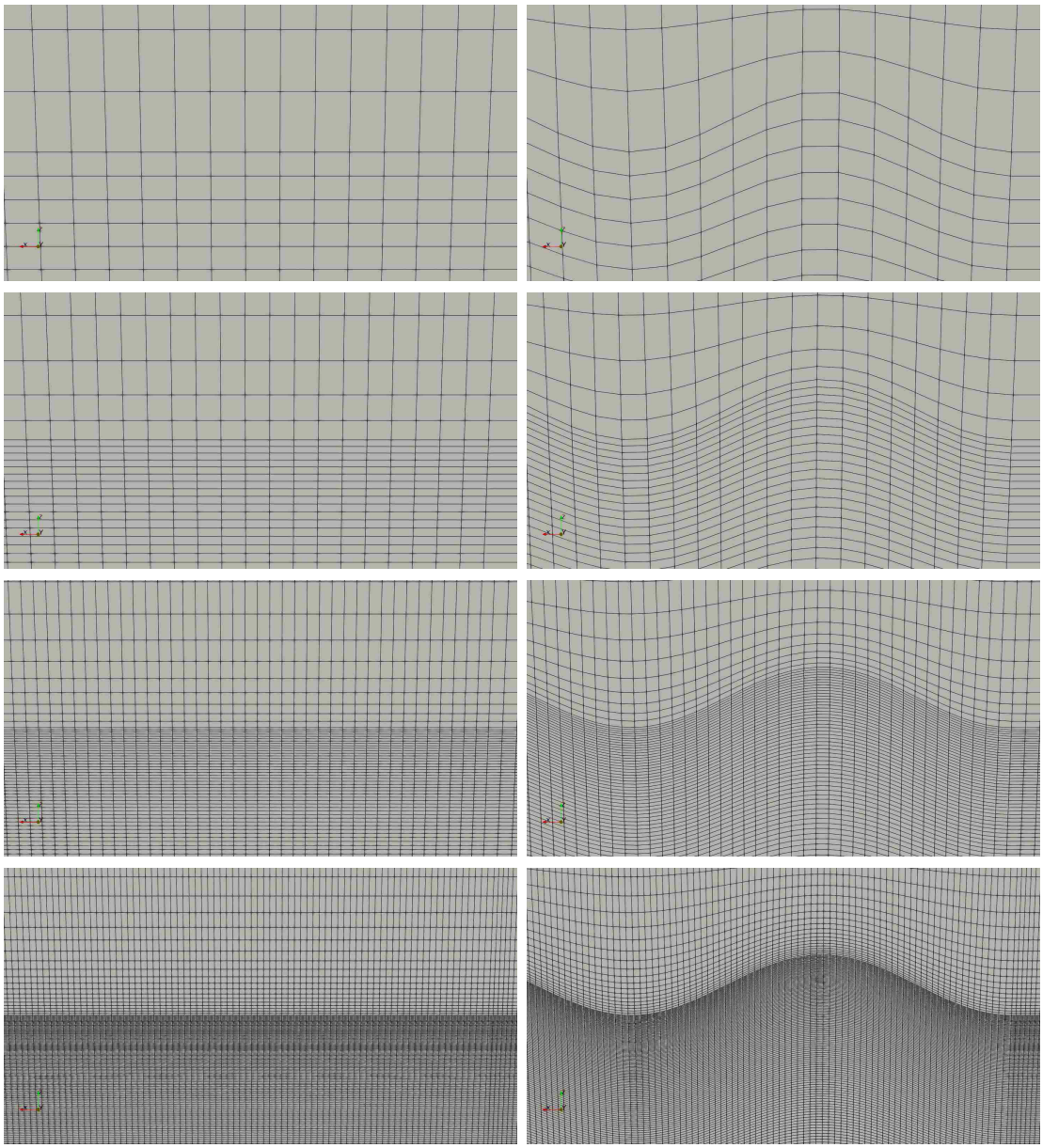


Figure C.5: Axisymmetric flow over *smooth* and *micro-structured* surfaces - Near-wall close up of the axisymmetric domain discretisation representing internal flow in a pipe for the grid spacing of –from top to bottom– h = 0.04, h = 0.02, h = 0.01 and h = 0.005 for –left to right– $A_1 = 0$ and $A_1 = 0.01$.

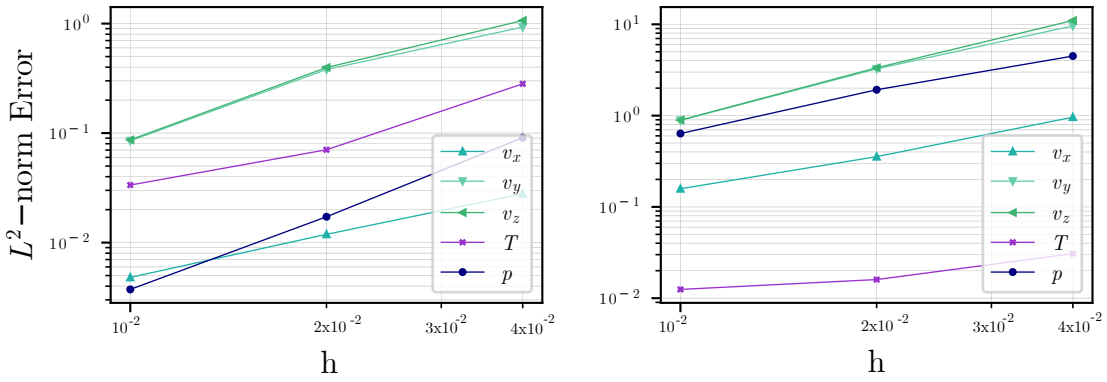


Figure C.6: Axisymmetric flow over *smooth* and *micro-structured* surfaces - Relative norm L^2 error $||\varepsilon_r||_{L^2}$ against h for the surface geometries of $A_1 = 0.0$ m in –left to right– laminar and turbulent regimes.

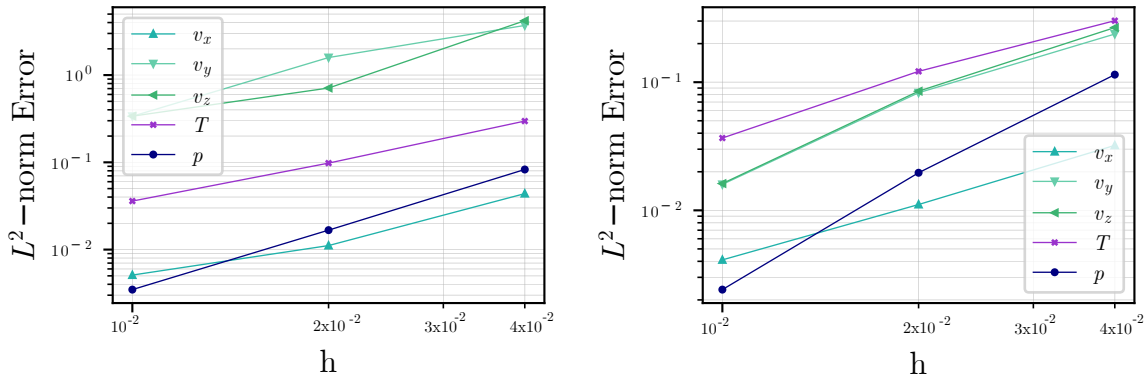


Figure C.7: Axisymmetric flow over *smooth* and *micro-structured* surfaces - Relative norm L^2 error $||\varepsilon_r||_{L^2}$ against h for the surface geometries of –left to right– $A_1 = 0.001$ m and $A_1 = 0.01$ m in laminar regime.

APPENDIX D

Hammerhead software additional material

"If at first you don't succeed; call it version 1.0."

— Unknown

D.1 SciPy's differential evolution

DE mutation strategies	
rand1	$\boldsymbol{x}' = \boldsymbol{x}_{r_0} + F \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2})$
rand2	$\boldsymbol{x}' = \boldsymbol{x}_{r_0} + F \cdot (\boldsymbol{x}_{r_1} + \boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3} - \boldsymbol{x}_{r_4})$
best1	$\boldsymbol{x}' = \boldsymbol{x}_0 + F \cdot (\boldsymbol{x}_{r_0} - \boldsymbol{x}_{r_1})$
best2	$\boldsymbol{x}' = \boldsymbol{x}_0 + F \cdot (\boldsymbol{x}_{r_0} + \boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3})$
currenttobest1	$\boldsymbol{x}' = \boldsymbol{x}_i + F \cdot (\boldsymbol{x}_0 - \boldsymbol{x}_i + \boldsymbol{x}_{r_0} - \boldsymbol{x}_{r_1})$
randtobest1	$\boldsymbol{x}' = \boldsymbol{x}_{r_0} + F \cdot (\boldsymbol{x}_0 - \boldsymbol{x}_{r_0} + \boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2})$

Table D.1: Mutation strategy functions available in the Scipy DE package within Python [145, 146]. The crossover probability is based on a binomial bin or exponential exp distribution, and is added to the name of the mutation strategy in the package (e.g. rand1bin).

The package used within Hammerhead for global optimisation is based on the differential evolution (DE) method in ScyPy [145]. This is an *evolutionary algorithm* that iteratively mutates a candidate solution with regard to a given criterion. The method is metaheuristic and non gradient-based so it doesn't guarantee finding an optimal solution but it can be used in noisy and non-continuous problems. DE arrives at an optimal result through trial candidate solutions using 3 main steps: mutation, crossover and selection. In the mutation stage, an initial random population of $\mathrm{N_p}$ solutions is generated within the D dimensional control parameter space, where each population member $\boldsymbol{x}_i$, $i = 1, 2, 3, ..., \mathrm{N_p}$ is set as the target vector at generation G, and a new mutant vector $\boldsymbol{x}'$ is created with a *mutation strategy* [146]. There are a variety of mutation strategies available in ScyPy presented in Table D.1, where $\boldsymbol{x}_0$ is the best solution of the current generation G, F is the mutation parameter, and r_1, r_2, r_3, r_4 and r_5 are randomly chosen integers from $[1,\ \mathrm{N_p}]$, different from the current index i.
The crossover stage combines $\boldsymbol{x}'$ and $\boldsymbol{x}_0$ into a trial candidate vector $\boldsymbol{U}'_i = u_{i1}, u_{i2}, ..., u_{iD}$, $i = 1, 2, 3, ..., \mathrm{N_p}$. The components are obtained from the crossover probability C to ensure $\boldsymbol{U}'_i$ contains at least 1 component from $\boldsymbol{x}'$, using the rule

$$u_{ij} = \begin{cases} x'_{ij}, & \text{if} \mathrm{rand}_i \leq C \quad \text{or} j = \mathrm{mbr}_i \\ x_{ij}, & \text{otherwise} \end{cases}, \tag{D.1}$$

where rand_i is a random real number in either a binomial or exponential distribution, and mbr_i is a random integer in $[1,\ D]$. In the selection stage, if the trial candidate $\boldsymbol{U}'_i$ is outside of the control parameter space boundaries, a new trial solution is computed from a random sample within bounds. Then the objective function value is evaluated with $\boldsymbol{U}'_i$ and $\boldsymbol{x}_i$, where the best fit is kept in the next generation population. This procedure is repeated for all members of the population to obtain the next generation, until the criterion for the final global optimal solution has been satisfied or the maximum amount of iterations has been reached.

D.2 Gaussian process preliminary study

Parameter	
Kernels [132]	{RBF Kernel (RBFK), RQ Kernel (RQK), Linear Kernel (LK) Matern Kernel (MK)}

Table D.2: GP preliminary study - Numerical parameters for the Gaussian process benchmark study. $^{*}\nu$ is required for the Matern kernel, the rest of kernels ignore this parameter.

The description of a GP model and the training process is given in Section 4.5. The assessment of Gaussian Process (GP) models presented in Section 7.3 is based on a preliminary study of a broader set of kernels based on low dimension input features $\boldsymbol{\omega}_4$ and output matrix $\mathbf{x}^{\dot{Q}}$. This study serves as a reference starting point for the implementation of RBF Kernel (RBFK) and Matern Kernel (MK) in the main assessment from Section 7.3. This was carried out to reduce the CPU time spent in the assessment of other kernels with increasing set sizes due to the memory and computational expense of the covariance matrix with a larger dataset and input features $\boldsymbol{\omega}_{\mathrm{N}_i}$. The characteristics are addressed are the ones presented in Section 7.3. The marginal log-likelihood $\mathrm{L_J}$ is the performance index of the GP training but a loss based on J is computed to address convergence. The GP models' benchmark for $\mathbf{y}_4^{\dot{Q}}\left(\boldsymbol{\omega}_{\mathrm{N}_i}, \mathbf{x}^{\mathrm{N}_x}\right)$, trained with validation set of 95% are shown in Figures D.1 to D.5. Consistency is not addressed in this preliminary study, and since the training set size if fixed at 95% of the dataset of $\boldsymbol{\omega}_4$, data sensitivity isn't considered either.

- **Convergence.** From Figure D.1, the kernel has some impact on J but all kernels reached values around $\mathrm{J}(\mathbf{y}_i^{\dot{Q}}) = 10^{-1}$. The Matern kernel is consistently in the lower range of values, however, the Linear and RBF kernels almost reach $\mathrm{J}(\mathbf{y}_i^{\dot{\Gamma}_f}) = 10^{-2}$ in $Re = 500$ and $Re = 8000$, respectively.

- **Over-fitting.** The errors $\mathrm{Er}(\mathbf{y}_4^{\dot{Q}})$ in Figures D.2 to D.5 show a density distribution in different ranges for each kernel. The higher error density area is in the range of $\mathrm{Er}(\mathbf{y}_4^{\dot{Q}}) = [10^{-1}, 10^{0}]$ for $Re = 5000$ with the Linear kernel, while the lowest is $\mathrm{Er}(\mathbf{y}_4^{\dot{Q}}) \approx 10^{-3}$ for $Re = 8000$ with the Matern kernel. The RBF, RQ and Linear kernels have a maximum error value from outliers over $\mathrm{Er}(\mathbf{y}_4^{\dot{Q}}) = 10^{1}$, while the Matern stays closer to $\mathrm{Er}(\mathbf{y}_4^{\dot{Q}}) = 10^{0}$. When observing the confidence region, the Matern kernel has a large region compared to the rest of kernels even though it has the most accurate predictions. On the other hand, the Linear kernel has very small confidence regions but the largest errors. RQ and RBF have similar behaviour regarding this.

- **Computational cost.** Most kernels have consistent CPU execution times, where Matern kernel takes $t(\mathbf{y}_4^{\dot{Q}}) \approx 14$ s, the RQ kernel takes $t(\mathbf{y}_4^{\dot{Q}}) \approx 18$ s and the Linear kernel takes $t(\mathbf{y}_4^{\dot{Q}}) \approx 12$ s. The RBF kernel did not yield consistent CPU times, with a range as

$t(\mathbf{y}_4^{\dot{Q}}) \approx [14,\ 23]$ s Note that the RBF kernel on $\mathbf{y}_4^{\Gamma_f}$ with a training set of 50% the complete dataset had a CPU time of $t \approx 1\ 200$ s, and the RQ kernel was not able to finalise training without errors on output sizes larger than $\mathbf{x}^{\dot{Q}}$.

The Matern and RBF kernels have the highest accuracy, with the Matern having the most consistent and lower CPU times of the two. In addition to the training errors yielded by the RQ kernel and the extremely low accuracy of the Linear kernel, the characteristics discussed in this preliminary study justify the selection of Matern and RBF kernels for the GP complexity analysis in Section 7.3.

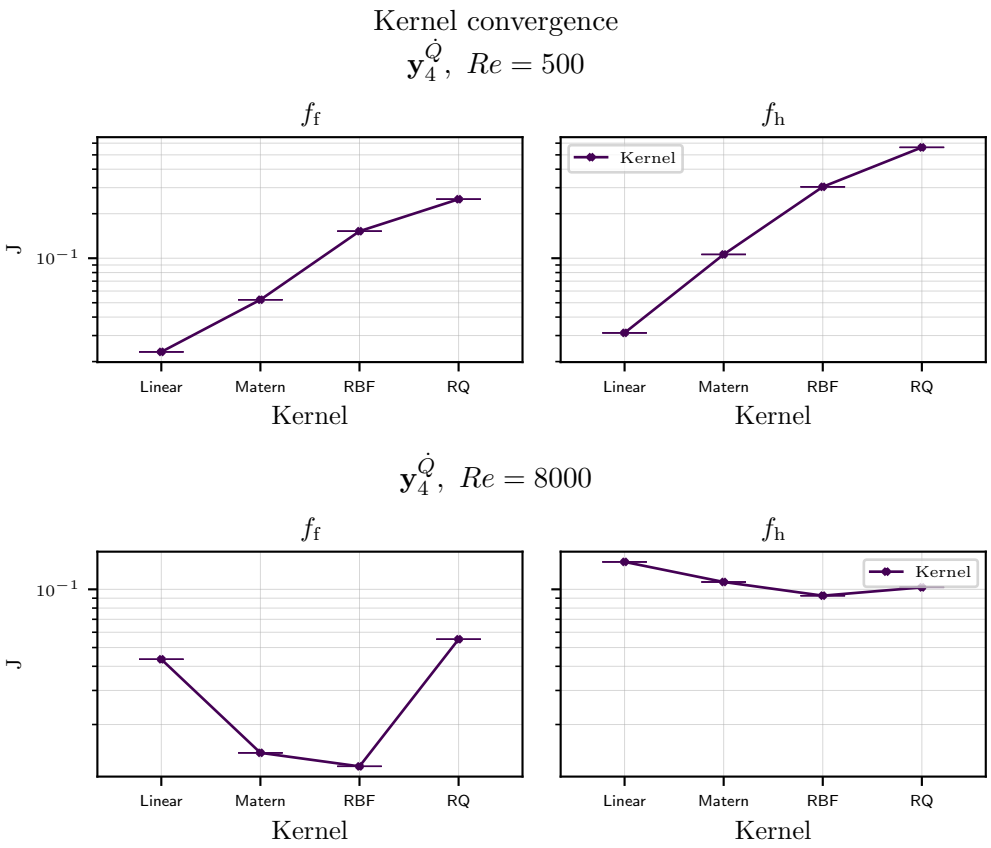


Figure D.1: L-GP benchmark - $\mathrm{J}(\mathbf{y}_4^{\dot{Q}})$ against kernels of dataset $\boldsymbol{\omega}_4$ feature GP trained with a validation split of 35%. The Lumped output data the GP is trained on is based off –left– advected heat flux (2.54) and –right– dissipation rate (2.52).

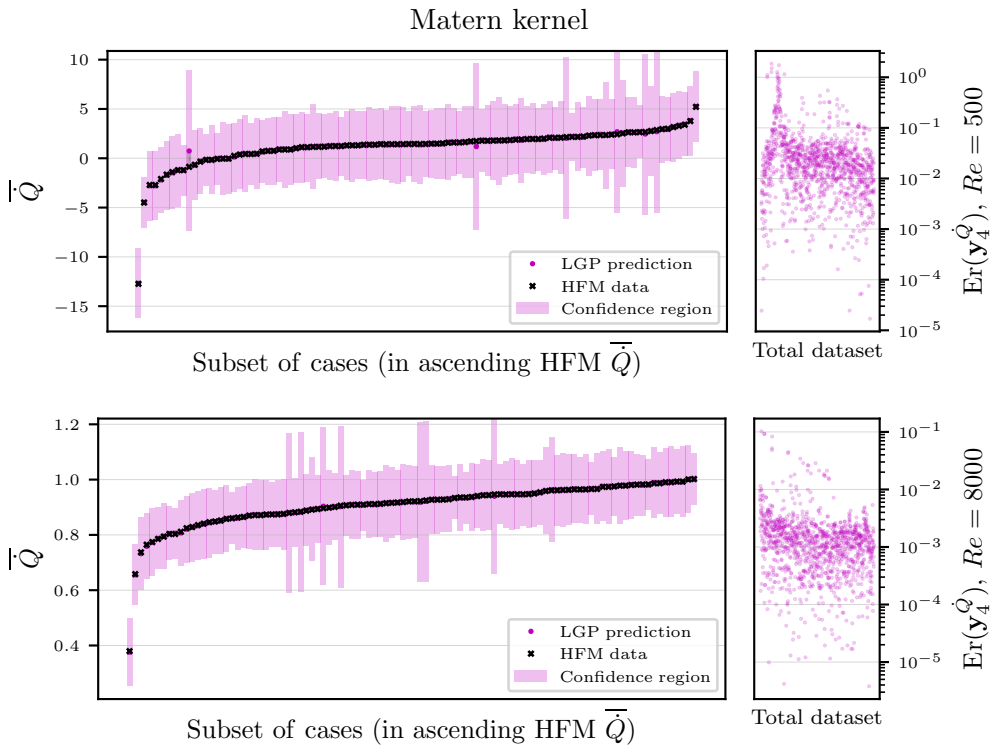


Figure D.2: L-GP kernel preliminary benchmark - Relative error $\mathrm{Er}(\mathbf{y}_5^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature GP trained with a Matern kernel with a validation split of 95% vs high fidelity data.

Figure D.3: L-GP kernel preliminary benchmark - Relative error $\mathrm{Er}(\mathbf{y}_5^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature GP trained with a RBF kernel with a validation split of 95% vs high fidelity data.

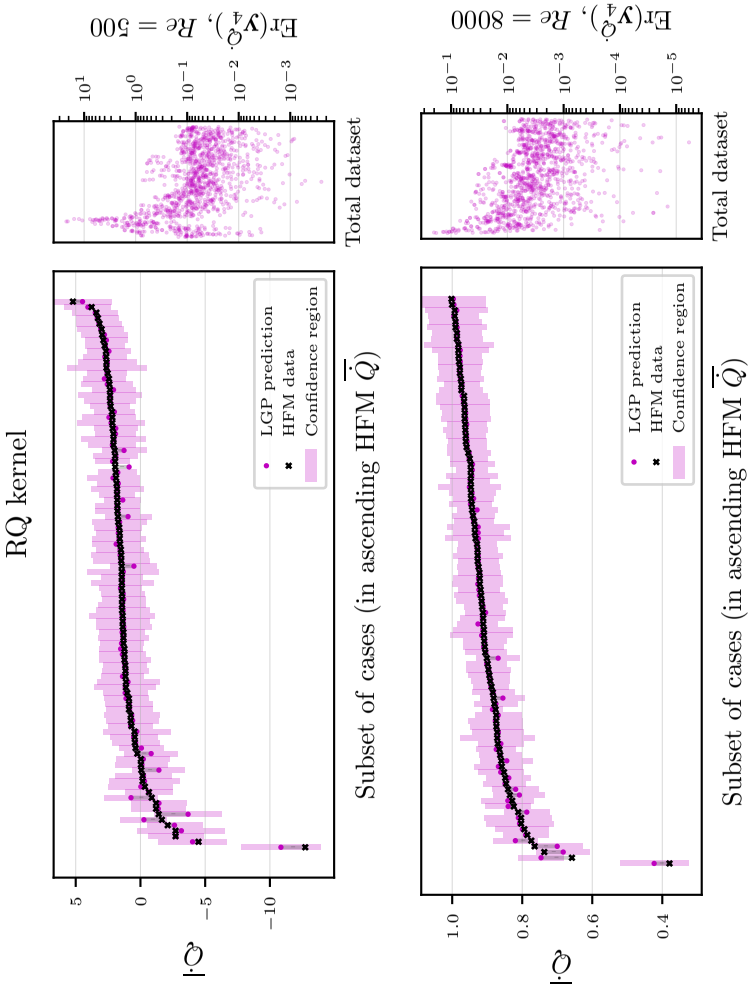


Figure D.4: L-GP kernel preliminary benchmark - Relative error $\mathrm{Er}(\mathbf{y}_5^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature GP trained with a RQ kernel with a validation split of 95% vs high fidelity data.

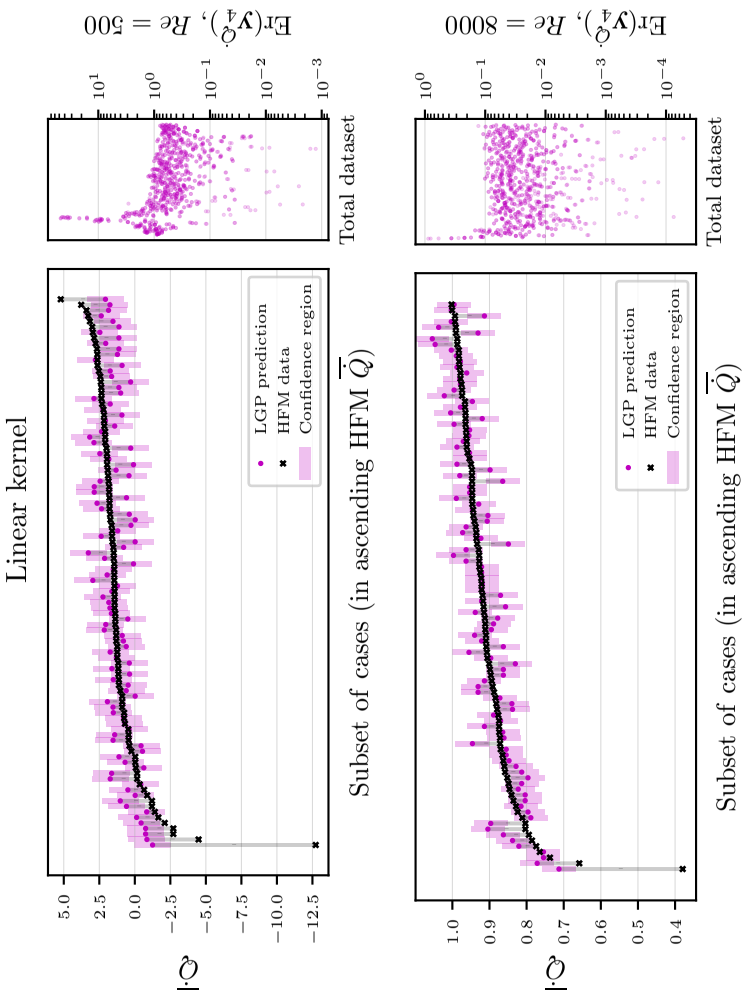


Figure D.5: L-GP kernel preliminary benchmark - Relative error $\mathrm{Er}(\boldsymbol{y}_5^{\dot{Q}})$ of independent cases prediction from the $\boldsymbol{\omega}_4$ feature GP trained with a Linear kernel with a validation split of 95% vs high fidelity data.

D.3 Python listings for surrogate model training

The model training is done through different functions for each algorithm. Since RBF is loaded from SciPy libraries, training is done via their modules and no class is required. For NN and GP, the training functions call the class, which although very simple, is unique to Hammerhead. The NN base uses a combination of linear and Sigmoid functions for each layer, and uses the Sequential model from PyTorch to built up the layer-neurons modules as shown in D.1. GP, similarly to RBF, uses `GPytorch`'s library of kernel functions, and has a base architecture shown in Listing D.2, so if any user input is not found in the library, it will lead to an error. The training functions are presented in Listings D.3, D.4 and D.5.

```
class Network(torch.nn.Module):
def __init__(self,
featureSize,
outputSize,
neurons,
layers):
super().__init__()
inputs = [featureSize] + [neurons for _ in range(layers + 1)]
outputs = [neurons for _ in range(layers + 1)] + [outputSize]

linearSteps = [torch.nn.Linear(i, o) for i, o in zip(inputs, outputs)]
sigmoidStep = torch.nn.Sigmoid()

archLayers = sum([[linearStep, sigmoidStep]
for linearStep in linearSteps],
[])[:-1]
self.model = torch.nn.Sequential(*archLayers)

def forward(self, x):
return self.model(x)
```

Listing D.1: General form of the NN class developed with PyTorch used within Hammerhead. Linear and Sigmoid functions are used for the hidden layers, which are stacked together in a list in the variable `archLayers`. `Sequential` adds the layers in the order they are passed in the constructor from the list `archLayers`. The function `forward()` is the computation performed for every NN call, where `x` represents the input features.

```
from gpytorch.means import ConstantMean
from gpytorch.kernels import ScaleKernel
from gpytorch.dristributions import MultitaskMultivariateNormal,
 MultivariateNormal

class Kriging(gpytorch.models.ExactGP):
def __init__(self,
outputTrain,
likelihood,
kernelGP,
featureSize,
outputSize):
super().__init__(xTrain, yTrain, likelihood)
self.mean = ConstantMean(batch_shape=torch.Size([dimensionSize]))
self.covar = ScaleKernel(kernelGP(nu=0.5, batch_shape=torch.Size(
[dimensionSize])),
batch_shape=torch.Size([dimensionSize]))
def forward(self, x):
return MultitaskMultivariateNormal.from_batch_mvn(
MultiVariateNormal(self.mean(x),self.covar(x)))
```

Listing D.2: General form of the GP class developed with GPyTorch used within Hammerhead. The `self.mean` and `self.covar` are GP model mean and kernel functions, `outputTrain` is the output results training set, the `kernelGP` is the user input architecture kernel. The function `forward()` computes multivariate normal random variable from a distribution based on the model mean and covariance, called every GP call where `x` represents the input features.

```
def RBF(kernelRBF,
        xtrain,
        outoutTrain)
    rbfi = RBFInterpolator(xTrain,
                           outputTrain,
                           kernel=kernelRBF,
                           smoothing=1e-2,
                           epsilon=3)
    lossFunc = torch.nn.MSELoss()
    lTrain = lossFunc(torch.from_numpy(rbfi(xTrain)), outputTrain)
```

Listing D.3: General form of the RBF function training from SciPy used within Hammerhead. RBFInterpolator is imported from `scipy.interpolate`, `xTrain` refers to the input features training set, `outputTrain` is the output results training set, the `kernelRBF` is the user input architecture kernel and the rest of parameters are smoothing values. To call RBF, rbfi is used.

```
def NN(featureSize,
       outputSize,
       layers,
       neurons,
       xTrain,
       outputTrain,
       xValid,
       outputValid,
       maxEpochs = int(9e4),
       lossTarget = 1e-6):
    network = Network(featureSize, outputSize, neurons, layers)

    optimizer = torch.optim.AdamW(network.parameters(), lr=1e-2)
    lossFunc = torch.nn.MSELoss()
    lossTrainList, lossValidList = [], []

    for epoch in range(maxEpochs):
        yPred = network(xTrain)
        lTrain = lossFunc(yPred, outputTrain)
        lossTrainList.append(lTrain.item())
        v = network(xValid)
        lVal = lossFunc(v, outputValid)
        lossValidList.append(lVal.item())
        optimizer.zero_grad()
        lTrain.backward()
        optimizer.step()
        if lossTrainList[-1] < lossTarget:
            break
```

Listing D.4: General form of the NN training function developed with PyTorch used within Hammerhead. `featureSize` refers to the amount of input features, `xTrain` refers to the input features training set, `outputTrain` is the output results training set, `xValid` refers to the input features validation set, `outputValid` is the output results validation set, `maxEpoch` sets the maximum amount of epoch to use for training and `lossTarget` gives a Loss target to stop the training early when reached.

```
from gpytorch import likelihoods, constraints, mlls, settings, metrics
def GP(featureSize,
       outputSize,
       kernelGP,
       xTrain,
       outputTrain,
       epochs = int(7e3),
       lossTarget = 1e-4):
    likelihood = likelihoods.MultitaskGaussianLikelihood(
                 num_tasks=dimensionSize,
                 noise_constraint=constraints.Interval(1e-5,1e-3))
    model = Kriging(xTrain, outputTrain, likelihood,
                    kernel, featureSize, dimensionSize)

    model.train()
    likelihood.train()

    optimizer = torch.optim.RAdam(model.parameters(),
                                  lr=1e-2, eps=1e-8, weight_decay=1e-2,
                                  decoupled_weight_decay=True)
    mll = mlls.ExactMarginalLogLikelihood(likelihood, model)

    lossTrainList = []
    epoch = 0
    with settings.lazily_evaluate_kernels(state=True),
         settings.fast_computations(covar_root_decomposition=True,
                                    log_prob=True, solves=False):
    while epoch < epochs:
        epoch += 1
      optimizer.zero_grad()
      distTrain = model(xTrain)
      mseTrain = metrics.mean_squared_error(distTrain,
                                            outputTrain,
                                            squared=True)
      output = model(xTrain)
      loss = -mll(output, outputTrain)
      loss.backward()
      optimizer.step()
      lossTrainList.append(max(mseTrain).item())
```

Listing D.5: General form of the GP training function developed with GPyTorch used within Hammerhead. `featureSize` refers to the amount of input features, `outputSize` refers to the dimension of the output results, `outputTrain` is the output results training set, `xTrain` refers to the input features training set, `outputTrain` is the output results training set, `kernelGP` is the user input architecture kernel, `outputValid` is the output results validation set, `epochs` sets the maximum amount of epoch to use for training and `lossTarget` gives a Loss target to stop the training early when reached.

Bibliography

[1] G. D. Bixler and B. Bhushan, "Fluid drag reduction with shark-skin riblet inspired microstructured surfaces", *Advanced Functional Materials*, vol. 23, pp. 4507–4528, 36 Sep. 2013. DOI: 10.1002/adfm.201203683.

[2] M. D. Buhmann, *Radial Basis Functions*, 1st ed. Cambridge University Press, Jul. 2003. DOI: 10.1017/CBO9780511543241.

[3] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. Springer New York, 2009, vol. 27, 83–85. DOI: 10.1007/978-0-387-84858-7.

[4] J. Wang, "An intuitive tutorial to gaussian process regression", *Computing in Science & Engineering*, vol. 25, pp. 4–11, 4 Jul. 2023. DOI: 10.1109/MCSE.2023.3342149.

[5] Y. C. Liang, H. P. Lee, S. P. Lim, W. Z. Lin, K. H. Lee and C. G. Wu, "Proper orthogonal decomposition and its applications—part i: Theory", *Journal of Sound and Vibration*, vol. 252, pp. 527–544, 3 May 2002. DOI: 10.1006/jsvi.2001.4041.

[6] *Iter tokamak*. Available: https://www.iter.org/ (visited on 15/07/2021).

[7] J. Wesson and D. J. Campbell, *Tokamaks*, 3rd. Claredon Press, 2004, pp. –749.

[8] J. Wesson, *The Science of JET*. Jet Joint Undertaking, 1999, p. 178.

[9] K. Hesch, L. V. Boccaccini and R. Stieglitz, "Blankets – key element of a fusion reactor – functions, design and present state of development", *Kerntechnik*, vol. 83, pp. 241–250, 3 Jun. 2018. DOI: 10.3139/124.110923.

[10] P. Brans, *Towards demo — what will the blanket teach us?*, Sep. 2019. Available: https://www.iter.org/newsline/-/3323 (visited on 15/07/2021).

[11] D. Jackson *et al.*, "A review of fusion breeder blanket technology, part 1", Canadian Fusion Fuels Technology Project, Tech. Rep., Jan. 1985, p. 309.

[12] F. Edemetti, E. Martelli, A. Tassone, G. Caruso and A. D. Nevo, "Demo wcll breeding zone cooling system design: Analysis and discussion", *Fusion Engineering and Design*, vol. 146, pp. 2632–2638, October 2018 Sep. 2019. DOI: 10.1016/j.fusengdes.2019.04.063.

[13] H. Sadeghi, R. Amrollahi, M. Zare and S. Fazelpour, "Design and simulation of a blanket module with high efficiency cooling system of tokamak focused on demo reactor", *Nuclear Engineering and Technology*, vol. 52, pp. 323–327, 2 Feb. 2020. DOI: 10.1016/j.net.2019.07.019.

[14] P. Chiovaro *et al.*, "Investigation of the demo wcll breeding blanket cooling water activation", *Fusion Engineering and Design*, vol. 157, p. 111 697, September 2019 Aug. 2020. DOI: 10.1016/j.fusengdes.2020.111697.

[15] F. Edemetti, P. Micheli, A. D. Nevo and G. Caruso, "Optimization of the first wall cooling system for the demo wcll blanket", *Fusion Engineering and Design*, vol. 161, p. 111 903, July Dec. 2020. DOI: 10.1016/j.fusengdes.2020.111903.

[16] I. Fernández-Berceruelo *et al.*, "Thermal-hydraulic design of a dcll breeding blanket for the eu demo", *Fusion Engineering and Design*, vol. 124, pp. 822–826, Nov. 2017. DOI: 10.1016/j.fusengdes.2017.03.108.

[17] J. Shimwell *et al.*, "Multiphysics analysis with cad-based parametric breeding blanket creation for rapid design iteration", *Nuclear Fusion*, vol. 59, p. 046 019, 4 Apr. 2019. DOI: 10.1088/1741-4326/ab0016.

[18] G. Marck, M. Nemer and J.-L. Harion, "Topology optimization of heat and mass transfer problems: Laminar flow", *Numerical Heat Transfer, Part B: Fundamentals*, vol. 63, pp. 508–539, 6 Jun. 2013. DOI: 10.1080/10407790.2013.772001.

[19] C. Othmer, "A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows", *International Journal for Numerical Methods in Fluids*, vol. 58, pp. 861–877, 8 Nov. 2008. DOI: 10.1002/fld.1770.

[20] V. Subramaniam, T. Dbouk and J.-L. Harion, "Topology optimization of conjugate heat transfer systems: A competition between heat transfer enhancement and pressure drop reduction", *International Journal of Heat and Fluid Flow*, vol. 75, pp. 165–184, January Feb. 2019. DOI: 10.1016/j.ijheatfluidflow.2019.01.002.

[21] A. Ghasemi and A. Elham, "Multi-objective topology optimization of pin-fin heat exchangers using spectral and finite-element methods", *Structural and Multidisciplinary Optimization*, vol. 64, pp. 2075–2095, 4 Oct. 2021. DOI: 10.1007/s00158-021-02964-6.

[22] P. Huang and M. Pan, "Secondary heat transfer enhancement design of variable cross-section microchannels based on entransy analysis", *Renewable and Sustainable Energy Reviews*, vol. 141, p. 110 834, September 2020 May 2021. DOI: 10.1016/j.rser.2021.110834.

[23] S. Soleimani and S. Eckels, "A review of drag reduction and heat transfer enhancement by riblet surfaces in closed and open channel flow", *International Journal of Thermofluids*, vol. 9, p. 100 053, Feb. 2021. DOI: 10.1016/j.ijft.2020.100053.

[24] D. W. Bechert, M. Bruse, W. Hage, J. G. T. V. D. Hoeven and G. Hoppe, "Experiments on drag-reducing surfaces and their optimization with an adjustable geometry", *Journal of Fluid Mechanics*, vol. 338, pp. 59–87, May 1997. DOI: 10.1017/S0022112096004673.

[25] D. W. Bechert, M. Bruse and W. Hage, "Experiments with three-dimensional riblets as an idealized model of shark skin", *Experiments in Fluids*, vol. 28, pp. 403–412, 5 May 2000. DOI: 10.1007/s003480050400.

[26] L. Wen, J. C. Weaver and G. V. Lauder, "Biomimetic shark skin: Design, fabrication and hydrodynamic function", *Journal of Experimental Biology*, vol. 217, pp. 1656–1666, 10 May 2014. DOI: 10.1242/jeb.097097.

[27] H. A. Abdulbari, H. D. Mahammed and Z. B. Y. Hassan, "Bio-inspired passive drag reduction techniques: A review", *ChemBioEng Reviews*, vol. 2, pp. 185–203, 3 Jun. 2015. DOI: 10.1002/cben.201400033.

[28] C. Yu *et al.*, "Bio-inspired drag reduction: From nature organisms to artificial functional surfaces", *Giant*, vol. 2, p. 100 017, July Jun. 2020. DOI: 10.1016/j.giant.2020.100017.

[29] E Stalio and E Nobile, "Direct numerical simulation of heat transfer over riblets", *International Journal of Heat and Fluid Flow*, vol. 24, pp. 356–371, 3 Jun. 2003. DOI: 10.1016/S0142-727X(03)00004-3.

[30] D.-G. Lee and Y.-J. Kim, "Effects of biomimetic riblet configurations on the drag reduction", *DEStech Transactions on Environment, Energy and Earth Science*, pp. 1–5, eccsd Mar. 2017. DOI: 10.12783/dteees/eccsd2016/5830.

[31] M. M. Rahman, "Measurements of heat transfer in microchannel heat sinks", *International Communications in Heat and Mass Transfer*, vol. 27, pp. 495–506, 4 May 2000. DOI: 10.1016/S0735-1933(00)00132-9.

[32] P. Li, D. Guo and X. Huang, "Heat transfer enhancement, entropy generation and temperature uniformity analyses of shark-skin bionic modified microchannel heat sink", *International Journal of Heat and Mass Transfer*, vol. 146, p. 118 846, Jan. 2020. DOI: 10.1016/j.ijheatmasstransfer.2019.118846.

[33] F. Hernández *et al.*, "A new hcpb breeding blanket for the eu demo: Evolution, rationale and preliminary performances", *Fusion Engineering and Design*, vol. 124, pp. 882–886, Nov. 2017. DOI: 10.1016/j.fusengdes.2017.02.008.

[34] C. Alberghi, L. Candido, R. Testoni, M. Utili and M. Zucchetti, "Magneto-convective effect on tritium transport at breeder unit level for the wcll breeding blanket of demo", *Fusion Engineering and Design*, vol. 160, p. 111 996, September 2019 Nov. 2020. DOI: 10.1016/j.fusengdes.2020.111996.

[35] S. Smolentsev *et al.*, "Mhd thermohydraulics analysis and supporting r&d for dcll blanket in the fnsf", *Fusion Engineering and Design*, vol. 135, pp. 314–323, Oct. 2018. DOI: 10.1016/j.fusengdes.2017.06.017.

[36] Y. Huang, M. Tillack, N. Ghoniem, J. Blanchard, L. El-Guebaly and C. Kessel, "Multiphysics modeling of the fw/blanket of the u.s. fusion nuclear science facility (fnsf)", *Fusion Engineering and Design*, vol. 135, pp. 279–289, Oct. 2018. DOI: 10.1016/j.fusengdes.2017.07.005.

[37] J.-C. Jaboulay, G. Aiello, J. Aubert, A. Morin and M. Troisne, "Nuclear analysis of the hcll blanket for the european demo", *Fusion Engineering and Design*, vol. 124, pp. 896–900, Nov. 2017. DOI: 10.1016/j.fusengdes.2017.01.050.

[38] J. Aubert *et al.*, "Status of the eu demo hcll breeding blanket design development", *Fusion Engineering and Design*, vol. 136, pp. 1428–1432, May Nov. 2018. DOI: 10.1016/j.fusengdes.2018.04.133.

[39] T. Zhang, X. Yang and X. Wang, "Level set-based topology optimization for thermal-fluid system based on the radial basis functions", *Applied Mathematical Modelling*, vol. 113, pp. 144–159, Jan. 2023. DOI: 10.1016/j.apm.2022.09.005.

[40] J. Alexandersen and C. S. Andreasen, "A review of topology optimisation for fluid-based problems", *Fluids*, vol. 5, p. 29, 1 Mar. 2020. DOI: 10.3390/fluids5010029.

[41] D. Tuckerman and R. Pease, "High-performance heat sinking for vlsi", *IEEE Electron Device Letters*, vol. 2, pp. 126–129, 5 May 1981. DOI: 10.1109/EDL.1981.25367.

[42] C. J. Lloyd *et al.*, "Hydrodynamic efficiency in sharks: The combined role of riblets and denticles", *Bioinspiration & Biomimetics*, vol. 16, p. 046 008, 4 Jul. 2021. DOI: 10.1088/1748-3190/abf3b1.

[43] D. M. Causon and C. G. Mingham, *Introductory Finite Difference Methods for PDEs*. Ventus Publishing ApS, 2010.

[44] D. M. Causon, C. G. Mingham and L. Qian, *Introductory Finite Volume Methods for PDEs*. Ventus Publishing ApS, 2011.

[45] F. Moukalled, L. Mangani and M. Darwish, *The Finite Volume Method in Computational Fluid Dynamics*. Springer International Publishing, 2016, vol. 113, pp. –791. DOI: 10.1007/978-3-319-16874-6.

[46] I. Nouzil, A. Eltaggaz, I. Deiab and S. Pervaiz, "Numerical cfd-fem model for machining titanium ti-6al-4v with nano minimum quantity lubrication: A step towards digital twin", *Journal of Materials Processing Technology*, vol. 312, p. 117 867, Mar. 2023. DOI: 10.1016/j.jmatprotec.2023.117867.

[47] R. Molinaro, J.-S. Singh, S. Catsoulis, C. Narayanan and D. Lakehal, "Embedding data analytics and cfd into the digital twin concept", *Computers & Fluids*, vol. 214, p. 104 759, Jan. 2021. DOI: 10.1016/j.compfluid.2020.104759.

[48] G. Aversano, M. Ferrarotti and A. Parente, "Digital twin of a combustion furnace operating in flameless conditions: Reduced-order model development from cfd simulations", *Proceedings of the Combustion Institute*, vol. 38, pp. 5373–5381, 4 2021. DOI: 10.1016/j.proci.2020.06.045.

[49] F. Shao and Y. Wang, "Intelligent overall planning model of underground space based on digital twin", *Computers and Electrical Engineering*, vol. 104, p. 108 393, Dec. 2022. DOI: 10.1016/j.compeleceng.2022.108393.

[50] Y. Liu *et al.*, "Digital twin of the atmospheric turbulence channel based on self-supervised deep learning algorithm", *Physics Letters A*, vol. 481, p. 128 992, Sep. 2023. DOI: 10.1016/j.physleta.2023.128992.

[51] *What is a digital twin? - digital twin technology explained - aws*, 2024. Available: https://aws.amazon.com/what-is/digital-twin/ (visited on 20/10/2024).

[52] K. Balla, R. Sevilla, O. Hassan and K. Morgan, "An application of neural networks to the prediction of aerodynamic coefficients of aerofoils and wings", *Applied Mathematical Modelling*, vol. 96, pp. 456–479, Aug. 2021. DOI: 10.1016/j.apm.2021.03.019.

[53] J. Weiss, "A tutorial on the proper orthogonal decomposition", in *AIAA Aviation 2019 Forum*, American Institute of Aeronautics and Astronautics, Jun. 2019, pp. 1–21. DOI: 10.2514/6.2019-3333.

[54] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning*. Cambridge University Press, May 2014, pp. 1–397. DOI: 10.1017/CBO9781107298019.

[55] Y. Wang, B. Yu, Z. Cao, W. Zou and G. Yu, "A comparative study of pod interpolation and pod projection methods for fast and accurate prediction of heat transfer problems", *International Journal of Heat and Mass Transfer*, vol. 55, pp. 4827–4836, 17-18 Aug. 2012. DOI: 10.1016/j.ijheatmasstransfer.2012.04.053.

[56] T. Bui-Thanh, M. Damodaran and K. Willcox, "Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition", *AIAA Journal*, vol. 42, pp. 1505–1516, 8 Aug. 2004. DOI: 10.2514/1.2159.

[57] A. Chatterjee, "An introduction to the proper orthogonal decomposition", *Current Science*, vol. 78, pp. 808–817, 7 Apr. 2000.

[58] M. Seoane, P. D. Ledger, A. J. Gil, S. Zlotnik and M. Mallett, "A combined reduced order-full order methodology for the solution of 3d magneto-mechanical problems with application to magnetic resonance imaging scanners", *International Journal for Numerical Methods in Engineering*, vol. 121, pp. 3529–3559, 16 Aug. 2020. DOI: 10.1002/nme.6369.

[59] F. Ballarin, A. D'Amario, S. Perotto and G. Rozza, "A pod-selective inverse distance weighting method for fast parametrized shape morphing", *International Journal for Numerical Methods in Engineering*, vol. 117, pp. 860–884, 8 Feb. 2019. DOI: 10.1002/nme.5982.

[60] R. Löhner, H. Antil, H. Tamaddon-Jahromi, N. K. Chakshu and P. Nithiarasu, "Deep learning or interpolation for inverse modelling of heat and fluid flow problems?", *International Journal of Numerical Methods for Heat & Fluid Flow*, vol. 31, pp. 3036–3046, 9 Aug. 2021. DOI: 10.1108/HFF-11-2020-0684.

[61] C. E. Rasmussen and C. K. I. Williams, "Gaussian processes for machine learning", in *Adaptive Computation and Machine Learning*. 2006, vol. 11, pp. 32–46.

[62] O. Bousquet, U. Luxburg and G. Ratsch, *Advanced Lectures on Machine Learning*, O. Bousquet, U. von Luxburg and G. Rätsch, Eds. Springer Berlin Heidelberg, 2004, vol. 3176, p. 240. DOI: 10.1007/b100712.

[63] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer (India) Private Limited, 2013, vol. 2, pp. –738.

[64] N. J. Nilsson, "Introduction to machine learning, an early draft of a proposed textbook", Ph.D. dissertation, Stanford University, 1998, pp. 387–99.

[65] K. Linka, M. Hillgärtner, K. P. Abdolazizi, R. C. Aydin, M. Itskov and C. J. Cyron, "Constitutive artificial neural networks: A fast and general approach to predictive data-driven constitutive modeling by deep learning", *Journal of Computational Physics*, vol. 429, p. 110 010, Mar. 2021. DOI: 10.1016/j.jcp.2020.110010.

[66] A. H. Monahan, "Nonlinear principal component analysis by neural networks: Theory and application to the lorenz system", *Journal of Climate*, vol. 13, pp. 821–835, 4 Feb. 2000. DOI: 10.1175/1520-0442(2000)013<0821:NPCABN>2.0.CO;2.

[67] J. Hesthaven and S. Ubbiali, "Non-intrusive reduced order modeling of nonlinear problems using neural networks", *Journal of Computational Physics*, vol. 363, pp. 55–78, Jun. 2018. DOI: 10.1016/j.jcp.2018.02.037.

[68] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces", *Journal of Global Optimization*, vol. 21, pp. 345–383, 4 2001. DOI: 10.1023/A:1012771025575.

[69] C. Latanoiotis, D. Wicaksono, S. Marelli and B. Sudret, "Uqlab user manual - kriging (gaussian process modeling)", Chair of Risk, Safety and Uncertainty Quantification, ETH, Tech. Rep., 2022.

[70] S. Zheng, S. Li, C. Hu and W. Li, "Performance optimization of latent heat storage device based on surrogate-assisted multi-objective evolutionary algorithm and cfd method", *Journal of Energy Storage*, vol. 99, p. 113 338, Oct. 2024. DOI: 10.1016/j.est.2024.113338.

[71] B. G. Knight and K. J. Maki, "Fast-running cfd calculations of the onr tumblehome performing maneuvers in calm water and in waves using a gaussian process regression propeller and rudder model", *Ocean Engineering*, vol. 303, p. 117 671, Jul. 2024. DOI: 10.1016/j.oceaneng.2024.117671.

[72] Y. Duan *et al.*, "Using a gaussian process regression inspired method to measure agreement between the experiment and cfd simulations", *International Journal of Heat and Fluid Flow*, vol. 80, p. 108 497, Dec. 2019. DOI: 10.1016/j.ijheatfluidflow.2019.108497.

[73] A. Coppedè, S. Gaggero, G. Vernengo and D. Villa, "Hydrodynamic shape optimization by high fidelity cfd solver and gaussian process based response surface method", *Applied Ocean Research*, vol. 90, p. 101 841, Sep. 2019. DOI: 10.1016/j.apor.2019.05.026.

[74] J. H. Spurk and N. Aksel, *Fluid Mechanics*, 2nd ed. Springer Berlin Heidelberg, 2008, p. 531. DOI: 10.1007/978-3-540-73537-3.

[75] L. D. Landau and E. M. Lifshitz, *Fluid Mechanics*, 2nd ed. Elsevier, 1987, vol. 6, pp. –539. DOI: 10.1016/C2013-0-03799-1.

[76] C. Hirsch, *Numerical Computation of Internal and External Flows*, 2nd ed. Elsevier, 2007, vol. 1, pp. –656. DOI: 10.1016/B978-0-7506-6594-0.X5037-1.

[77] I. G. Currie and I. Currie, *Fundamental Mechanics of Fluids*, 3rd ed. CRC Press, Dec. 2002, p. 525. DOI: 10.1201/9781482275889.

[78] H. K. Versteeg and W Malalasekera, *An Introduction to Computational Fluid Dynamics: the finite volume method*, 2nd ed. Pearson Prentice Hall, 2007, pp. –503.

[79] J. N. Reddy and D. Gartling, *The Finite Element Method in Heat Transfer and Fluid Dynamics*, 3rd ed., J. N. Reddy, Ed. CRC Press, Apr. 2010, pp. –524. DOI: 10.1201/9781439882573.

[80] J. Bonet, A. J. Gil and R. D. Wood, *Nonlinear Solid Mechanics for Finite Element Analysis: Dynamics*. Cambridge University Press, Mar. 2021, pp. –348. DOI: 10.1017/9781316336083.

[81] O. C. Zienkiewicz and R. L. Taylor, *The finite element method: Solid Mechanics*, 5th ed. Butterworth-Heinemann, 2000, pp. –459.

[82] G. Bar-Meir, *Basics of Fluid Mechanics*, 0.2.4. Potto Project, 2010, pp. –280. DOI: 10.5281/zenodo.8184518.

[83] J. Darong, "An hp-finite element computational framework for nonlinear magneto-fluid problems including magnetostriction", Ph.D. dissertation, Swansea University, 2015.

[84] A. B. Pippard, *Elements of Classical Thermodynamics: For Advanced Students of Physics*, 5th ed. Cambridge University Press, 1966, pp. –165.

[85] *Thermodynamic potentials*, 2022.

[86] *Heat transfer: Conservation of energy*, Jun. 2018. Available: https://www.comsol.com/multiphysics/heat-transfer-conservation-of-energy (visited on 01/02/2024).

[87] P. Renze and K. Akermann, "Simulation of conjugate heat transfer in thermal processes with open source cfd", *ChemEngineering*, vol. 3, p. 59, 2 Jun. 2019. DOI: 10.3390/chemengineering3020059.

[88] W. H. Rohsenow, J. P. Hartnett and Y. I. Cho, *Handbook of heat transfer*, 3rd ed., W. H. Rohsenow, J. P. Hartnett and Y. I. Cho, Eds. McGraw-Hill, 1998, pp. –1520.

[89] T. V. Karman, H. L. Dryden and H. S. Taylok, *Turbulent Flows and Heat Transfer*, C. C. Lin, Ed. Princeton University Press, 1959, vol. 5, pp. –549.

[90] M. Pietropaoli, F. Montomoli and A. Gaymann, "Three-dimensional fluid topology optimization for heat transfer", *Structural and Multidisciplinary Optimization*, vol. 59, pp. 801–812, 3 Mar. 2019. DOI: 10.1007/s00158-018-2102-4.

[91] B. Dean and B. Bhushan, "Shark-skin surfaces for fluid-drag reduction in turbulent flow: A review", *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, pp. 4775–4806, 1929 Oct. 2010. DOI: 10.1098/rsta.2010.0201.

[92] M. Kadivar, D. Tormey and G. McGranaghan, "A review on turbulent flow over rough surfaces: Fundamentals and theories", *International Journal of Thermofluids*, vol. 10, p. 100 077, May 2021. DOI: 10.1016/j.ijft.2021.100077.

[93] A. A. Townsend, *The Structure of Turbulent Shear Flow*, 2nd ed. Cambridge Univesity Press, 1999.

[94] J Nikuradse, "Laws of flow in rough pipes", National Advisory Committee for Aeronautics, Tech. Rep., Nov. 1950.

[95] B. Dean and B. Bhushan, "The effect of riblets in rectangular duct flow", *Applied Surface Science*, vol. 258, pp. 3936–3947, 8 Feb. 2012. DOI: 10.1016/j.apsusc.2011.12.067.

[96] Y. C. Jung and B. Bhushan, "Biomimetic structures for fluid drag reduction in laminar and turbulent flows", *Journal of Physics: Condensed Matter*, vol. 22, p. 035 104, 3 Jan. 2010. DOI: 10.1088/0953-8984/22/3/035104.

[97] S. Martin and B. Bhushan, "Fluid flow analysis of a shark-inspired microstructure", *Journal of Fluid Mechanics*, vol. 756, pp. 5–29, Oct. 2014. DOI: 10.1017/jfm.2014.447.

[98] H. Choi, P. Moin and J. Kim, "Direct numerical simulation of turbulent flow over riblets", *Journal of Fluid Mechanics*, vol. 255, pp. 503–539, Oct. 1993. DOI: 10.1017/S0022112093002575.

[99] M. J. Walsh and L. M. Weinstein, "Drag and heat-transfer characteristics of small longitudinally ribbed surfaces", *AIAA Journal*, vol. 17, pp. 770–771, 7 Jul. 1979. DOI: 10.2514/3.61216.

[100] A. M. Lindemann, "Turbulent reynolds analogy factors for nonplanar surface microgeometries", *Journal of Spacecraft and Rockets*, vol. 22, pp. 581–582, 5 Sep. 1985. DOI: 10.2514/3.25782.

[101] K.-S Choi and D. M. Orchard, "Turbulence management using riblets for heat and momentum transfer", *Experimental Thermal and Fluid Science*, vol. 15, pp. 109–124, 1997.

[102] Y. Jin and H. Herwig, "Turbulent flow and heat transfer in channels with shark skin surfaces: Entropy generation and its physical significance", *International Journal of Heat and Mass Transfer*, vol. 70, pp. 10–22, Mar. 2014. DOI: 10.1016/j.ijheatmasstransfer.2013.10.063.

[103] S. aus der Wiesche, "Heat transfer and drag reduction in flows over riblet mounted surfaces", in *Heat Transfer: Volume 1*, ASMEDC, Jan. 2003, pp. 449–457. DOI: 10.1115/HT2003-47356.

[104] M. Benhalilou and N. Kasagi, "Numerical prediction of heat and momentum transfer over micro-grooved surface with a nonlinear k–ϵ model", *International Journal of Heat and Mass Transfer*, vol. 42, pp. 2525–2541, 14 Jul. 1999. DOI: 10.1016/S0017-9310(98)00348-2.

[105] B. Dai, M. Li and Y. Ma, "Effect of surface roughness on liquid friction and transition characteristics in micro- and mini-channels", *Applied Thermal Engineering*, vol. 67, pp. 283–293, 1-2 Jun. 2014. DOI: 10.1016/j.applthermaleng.2014.03.028.

[106] P Bradshaw, "Turbulent secondary flows", *Annual Review of Fluid Mechanics*, vol. 19, pp. 53–74, 1 Jan. 1987. DOI: 10.1146/annurev.fl.19.010187.000413.

[107] D. Willingham, W. Anderson, K. T. Christensen and J. M. Barros, "Turbulent boundary layer flow over transverse aerodynamic roughness transitions: Induced mixing and flow characterization", *Physics of Fluids*, vol. 26, 2 Feb. 2014. DOI: 10.1063/1.4864105.

[108] J. M. Barros and K. T. Christensen, "Observations of turbulent secondary flows in a rough-wall boundary layer", *Journal of Fluid Mechanics*, vol. 748, R1–R13, 2 Apr. 2014. DOI: 10.1017/jfm.2014.218.

[109] W. Anderson, J. M. Barros, K. T. Christensen and A. Awasthi, "Numerical and experimental study of mechanisms responsible for turbulent secondary flows in boundary layer flows over spanwise heterogeneous roughness", *Journal of Fluid Mechanics*, vol. 768, pp. 316–347, 2015. DOI: 10.1017/jfm.2015.91.

[110] S. Martin and B. Bhushan, *Modeling and optimization of shark-inspired riblet geometries for low drag applications*, Jul. 2016. DOI: 10.1016/j.jcis.2016.04.019.

[111] K. Okabayashi, K. Hirai, S. Takeuchi and T. Kajishima, "Direct numerical simulation of turbulent flow above zigzag riblets", *AIP Advances*, vol. 8, 10 Oct. 2018. DOI: 10.1063/1.5049714.

[112] H. Benschop and W.-P. Breugem, "Drag reduction by herringbone riblet texture in direct numerical simulations of turbulent channel flow", *Journal of Turbulence*, vol. 18, pp. 717–759, 8 Aug. 2017. DOI: 10.1080/14685248.2017.1319951.

[113] A. E. Perry, W. H. Schofield and P. N. Joubert, "Rough wall turbulent boundary layers", *Journal of Fluid Mechanics*, vol. 37, pp. 383–413, 2 Jun. 1969. DOI: 10.1017/S0022112069000619.

[114] N. R. Vadlamani, P. G. Tucker and P. Durbin, "Distributed roughness effects on transitional and turbulent boundary layers", *Flow, Turbulence and Combustion*, vol. 100, pp. 627–649, 3 Apr. 2018. DOI: 10.1007/s10494-017-9864-4.

[115] E. Dick and S. Kubacki, "Transition models for turbomachinery boundary layer flows: A review", *International Journal of Turbomachinery, Propulsion and Power*, vol. 2, p. 4, 2 Apr. 2017. DOI: 10.3390/ijtpp2020004.

[116] Z. Yang, "On bypass transition in separation bubbles: A review", *Propulsion and Power Research*, vol. 8, pp. 23–34, 1 Mar. 2019. DOI: 10.1016/j.jppr.2018.12.004.

[117] T. C. Corke, A. Bar-Sever and M. V. Morkovin, "Experiments on transition enhancement by distributed roughness", *The Physics of Fluids*, vol. 29, pp. 3199–3213, 10 Oct. 1986. DOI: 10.1063/1.865838.

[118] *Openfoam*. Available: https://www.openfoam.com/ (visited on 12/10/2022).

[119] H. G. Weller, G. Tabor, H. Jasak and C. Fureby, "A tensorial approach to computational continuum mechanics using object-oriented techniques", *Computers in Physics*, vol. 12, pp. 620–631, 6 Nov. 1998. DOI: 10.1063/1.168744.

[120] T. Holzmann, *Mathematics, Numerics, Derivations and OpenFOAM: The Basics for Numerical simulations*, Release 7.0. Holzmann CFD, 2020. DOI: 10.13140/RG.2.2.27193.36960.

[121] *Chtmultiregionfoam*, Apr. 2019.

[122] *4.3 mesh generation with the blockmesh utility*. Available: https://www.openfoam.com/documentation/user-guide/4-mesh-generation-and-conversion/4.3-mesh-generation-with-the-blockmesh-utility (visited on 08/07/2024).

[123] M. T. Hagan, H. B. Demuth, M. H. Beale and O. D. Jesús, *Neural Network Design*, 2nd ed. Martin Hagan, 2014, pp. –800.

[124] P. Langley, "The changing science of machine learning", *Machine Learning*, vol. 82, pp. 275–279, 3 Mar. 2011. DOI: 10.1007/s10994-011-5242-y.

[125] V. Buljak, *Inverse Analyses with Model Reduction*, J. B. K, Ed. Springer Berlin Heidelberg, 2012, pp. 1–2443. DOI: 10.1007/978-3-642-22703-5.

[126] D. W. Toit, "Radial basis function interpolation", Ph.D. dissertation, University of Stellenbosch, Mar. 2008, pp. –58.

[127] *Rbfinterpolator — scipy v1.14.1 manual*. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.RBFInterpolator.html (visited on 23/12/2024).

[128] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer New York, 2000, p. 188. DOI: 10.1007/978-1-4757-3264-1.

[129] J. R. Koza, F. H. Bennett, D. Andre and M. A. Keane, "Automated design of both the topology and sizing of analog electrical circuits using genetic programming", in *Artificial Intelligence in Design '96*. Springer Netherlands, 1996, pp. 151–170. DOI: 10.1007/978-94-009-0279-4_9.

[130] X. Wu *et al.*, "Top 10 algorithms in data mining", *Knowledge and Information Systems*, vol. 14, pp. 1–37, 1 Jan. 2008. DOI: 10.1007/s10115-007-0114-2.

[131] C. Cortes and V. Vapnik, "Support-vector networks", *Machine Learning*, vol. 20, pp. 273–297, 3 Sep. 1995. DOI: 10.1007/BF00994018.

[132] G. P. Cornellius, *Gpytorch kernels*, 2023. Available: https://docs.gpytorch.ai/en/latest/kernels.html (visited on 05/12/2024).

[133] *Github - dani-darko/hammerhead*. Available: https://github.com/Dani-Darko/Hammerhead/tree/main.

[134] F. Menter, "Zonal two equation k-wturbulence models for aerodynamic flows", in *23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference*, vol. 93, American Institute of Aeronautics and Astronautics, Jul. 1993. DOI: 10.2514/6.1993-2906.

[135] F. R. Menter, "Two-equation eddy-viscosity turbulence models for engineering applications", *AIAA Journal*, vol. 32, pp. 1598–1605, 8 Aug. 1994. DOI: 10.2514/3.12149.

[136] A. N. Colli and J. M. Bisang, "A cfd study with analytical and experimental validation of laminar and turbulent mass-transfer in electrochemical reactors", *Journal of The Electrochemical Society*, vol. 165, E81–E88, 2 Jan. 2018. DOI: 10.1149/2.0971802jes.

[137] J. P. Abraham, E. M. Sparrow and J. C. Tong, "Heat transfer in all pipe flow regimes: Laminar, transitional/intermittent, and turbulent", *International Journal of Heat and Mass Transfer*, vol. 52, pp. 557–563, 3-4 Jan. 2009. DOI: 10.1016/j.ijheatmasstransfer.2008.07.009.

[138] F. Incropera, D. P. Dewitt, T. Bergman and A. Lavine, "Internal flow", in *Fundamentals of Heat and Mass Transfer*, 6th ed. John Wiley & Sons, Inc., 2007.

[139] C Shu, H Ding and K. Yeo, "Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible navier–stokes equations", *Computer Methods in Applied Mechanics and Engineering*, vol. 192, pp. 941–954, 7-8 Feb. 2003. DOI: 10.1016/S0045-7825(02)00618-7.

[140] Y. Wang, Y. Ma and H. Chao, "Machine learning and computational fluid dynamics based optimization of finned heat pipe radiator performance", *Journal of Building Engineering*, vol. 78, p. 107 612, Nov. 2023. DOI: 10.1016/j.jobe.2023.107612.

[141] A. Mukhtar, A. S. H. M. Yasir and M. F. M. Nasir, "A machine learning-based comparative analysis of surrogate models for design optimisation in computational fluid dynamics", *Heliyon*, vol. 9, e18674, 8 Aug. 2023. DOI: 10.1016/j.heliyon.2023.e18674.

[142] D. C. Wilcox, *Turbulence Modelling for CFD*, 2nd ed. DCW Industries, 1994, p. 460.

[143] *Favre averaged navier-stokes equations*, 2005. Available: https://www.cfd-online.com/Wiki/Favre_averaged_Navier-Stokes_equations (visited on 05/08/2021).

[144] "Handbook on lead-bismuth eutectic alloy and lead properties, materials compatability, thermal-hydraulics and technologies", Organisation for Economic Co-operation and Development, Nuclear Energy Agency, Tech. Rep., 2007.

[145] *Differential_evolution — scipy v1.15.3 manual*. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential$_$evolution.html.

[146] J. Qiang and C. Mitchell, "A unified differential evolution algorithm for global optimization", Accelerator and Fusion Research Division, Lawrence Berkeley National Laboratory, Tech. Rep., 2014.