Cronfa
Swansea University's Research Repository

Simulation driven machine learning methods to optimise design of physical experiments and enhance data analysis for testing of fusion energy heat exchanger components

Rhydian Lewis

Submitted to Swansea University in fulfilment of the requirements for the Degree of Doctor of Philosophy

November 20, 2023

Copyright: The Author, Rhydian Lewis, 2023

Abstract

Plasma facing components (PFCs) must be designed to routinely withstand the harsh environment of a fusion device, where temperatures at the core of the plasma exceed 150,000,000 °C. The heat by induction to verify extremes (HIVE) experimental facility was established to replicate the thermal loads a PFC is subjected to during normal operation of a fusion device.

To maximise its impact on the design of PFCs, HIVE must deliver smarter testing and improved component insight. Currently, the experimental parameters required to deliver a certain response to the component are decided at the point of testing through a combination of previous experience, intuition, and trial & error, which is both time-consuming and unreliable. To assess a PFC's suitability, knowledge of its mechanical performance while operating at high temperatures is desirable, however HIVE only records pointwise temperature measurements on the component's surface using thermocouples. Currently, HIVE has no method of inferring a component's mechanical response using the temperature measurements.

Both the challenges of smarter testing and improved component insight can be achieved through the identification of inverse solutions. A popular approach to solving engineering inverse problems is surrogate assisted optimisation, where a machine learning model is trained using finite element (FE) simulation data. Much of the work in literature use single value surrogate models on quite simplistic problems, however HIVE is a real-world, multi-physics problem which requires full field (FF) surrogate models to solve its multitude of inverse problems.

The development of a method which can easily construct FE data driven FF surrogates would be invaluable for a variety of tasks in engineering, as well as solving inverse problems. In this work, it demonstrates that it can provide a much more robust and comprehensive method of characterising a PFC's strengths and limitations, enabling more informed decisions to be made during its design cycle.

Declarations

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

[redacted]

November 20, 2023

This thesis is the result of your own investigations, except where otherwise stated and that other sources are acknowledged by footnotes giving explicit references and that a bibliography is appended.

[redacted]

November 20, 2023

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations

[redacted]

November 20, 2023

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

[redacted]

November 20, 2023

Acknowledgements

Firstly, I would like to thank my supervisor Dr Llion Evans for his help and guidance with this project over the past four years. His passion and knowledge of science and technology has allowed me to learn and grow my skill set, while his professionalism and approachability has made this a thoroughly enjoyable and rewarding experience. I would also like to thank Prof. Perumal Nithiarasu for his advice and guidance during the project. Further to this, I would like to thank those involved with HIVE for answering my many questions, along with other colleagues at the UK Atomic Energy Authority, with a special mention to Ruben Otin, David Hancock and Andy Davis for sharing invaluable tools and ideas for the modelling of HIVE. Additionally, I would like to acknowledge the support of Jeyan Thiyagalingam and Kuangdai Leng at the Science and Technology Facilities Council, who have provided their expertise in the use of ML for this application.

A PhD is an individual project which at times can be all consuming, however its completion would be impossible without the support of friends and family. Thank you to all my friends, especially my fellow PhD students, who I have been extremely fortunate to be on this journey with. Thanks to my partner, Meg, for your support and for keeping my mind on other things, along with Bonnie for providing a welcome distraction and fresh air during a walk. Lastly, I would like to thank my family, especially my parents, for their continued love and support during all of my endeavours, both academic and personal. I could not have done any of it without you. Diolch.

Contents

List of Figures

List of Tables

Nomenclature

ρ_c	Charge density
T_{CHF}	Wall temperature for critical heat flux
k	Thermal conductivity
$P_{coolant}$	Coolant pressure
T_{sat}	Coolant saturation temperature
$T_{coolant}$	Coolant temperature
$V_{coolant}$	Coolant velocity
I	Induction coil courrent
$\mathbf{J}$	Current density
x_d	Coil displacement in x direction
y_d	Coil displacement in y direction
z_d	Coil displacement in z direction
$\mathbf{E}$	Electric field
T_{film}	Film temperature
f	Induction coil frequency
H_{fg}	Heat evaporation of fluid
h	Heat transfer coefficient
JH	Joule heating

$K_{te,te}$	Covariance matrix calculated using test data points
$K_{tr,te}, K_{te,tr}$	Covariance matrix calculated using training and test data points
$K_{tr,tr}$	Covariance matrix calculated using training data points
$\mathbf{B}$	Magnetic field
G	Mass velocity of coolant
σ_n^2	Gausian noise
Nu	Nusselt number
T_{ONB}	Wall temperature for onset of nucleate boiling
σ_f^2	Gaussian process output scale
θ	Parameters of supervised learning mapping
μ_r	Relative magnetic permeability
μ_0	Permeability of free space
ε_0	Permittivity of free space
q	Pipe flux
q_{CHF}	Pipe flux for critical heat flux
q_{FC}	Pipe flux during the forced convection regime
q_{NB}	Pipe flux during nucleate boiling regime
q_{ONB}	Pipe flux for onset of nucleate boiling
l_{pipe}	Pipe length
d_{pipe}	Pipe width
μ_*	Mean of posterior distribution
Σ_*	Variance of posterior distribution
Pr	Prandtl number

ρ	Electrical resistivity
$Re_{d_{pipe}}$	Renyolds number with reference length l_{pipe}
θ	Coil rotation along x axis
ϕ	Coil rotation along y axis
ψ	Coil rotation along z axis
d	Depth of eddy current in conductor
C_p	Specific heat capacity
σ_{VM}	Von Mises stress
$\mathbf{F}_\theta$,$\mathbf{F}$	Supervised learning mapping
X_{test}	Test data inputs
Y_{test}	Test data outputs
X_{train}	Train data inputs
Y_{train}	Train data outputs
x_{ex}	Vapor quality at exit of pipe
μ	Viscosity of fluid at bulk temperature
μ_{sat}	Viscosity of fluid at saturdation temperature
μ_{wall}	Viscosity of fluid at the wall temperature
T_{wall}	Wall (pipe) temperature
AC	Alternating current
AI	Artificial intelligence
ANN	Artificial neural network
BC	Boundary condition
CAD	Computer aided design

CFD	Computational fluid dynamics
CHIMERA	Combined heating and magnetic research apparatus
DoE	Design of experiments
DoPE	Design of physical experiments
EDF	Electricite de France
EM	Electromagnetic
FE	Finite element
FF	Full field
GDP	Gross domestic product
GPR	Gaussian process regression
GUI	Graphical user interface
HDF	Hierarchical data format
HHF	High heat flux
HIVE	Heat by induction to verify extremes
HPC	High performance computing
HTC	Heat transfer coefficient
IHTP	Inverse heat transfer problem
ML	Machine learning
MLP	Multi layer perceptron
MPI	Message passing interface
PC	Principal component
PCA	Principal component analysis
PDE	Partial differential equation

PFC	Plasma facing component
RAM	Random access memory
RNN	Recurrent neural network
SAO	Surrogate assisted optimisation
SV	Single value
UKAEA	UK atomic energy authority

Chapter 1

Introduction

1.1 Fusion Energy

The world's energy consumption has increased dramatically over the past 80 years, increasing from 22,869 TWh in 1940 to 176,431 TWh in 2021 [1]. This increase has been predominantly supplied through the burning of fossil fuels such as coal, oil and natural gas, highlighted by the breakdown of energy sources over time shown in fig. 1.1. The burning of fossil fuels releases carbon dioxide (CO_2) in to the atmosphere, resulting in an increase of more than 30% in its concentration in the atmosphere in the past 70 years, rising from 315 parts per million in 1950 to 414 in 2021 [2]. The link between rising global atmospheric temperatures, often termed as global warming, and increasing concentration of CO_2 and other greenhouse gases in the atmosphere is well established [3]. This has resulted in the majority of countries committing to net-zero greenhouse gas emissions by 2050, with China - the world's largest emitters - committing to 2060 [4], [5].

The energy sector accounts for 72% of global greenhouse gas emissions, therefore to achieve the ambitious 2050 net-zero targets alternative, emission-free energy sources are required [5]. Renewable energy sources, such as solar, wind and tidal, have developed rapidly over the past two decades. By 2026 wind turbines which produce in excess of 16 MW of power will be available, producing more than 80,000 MWh of energy per year [6]. While renewables are integral to the makeup of the net-zero energy portfolio, their supply is unreliable due to their dependence on weather, especially on a local level. It is essential that other energy sources are available at times when renewables are unable to satisfy the energy demand.

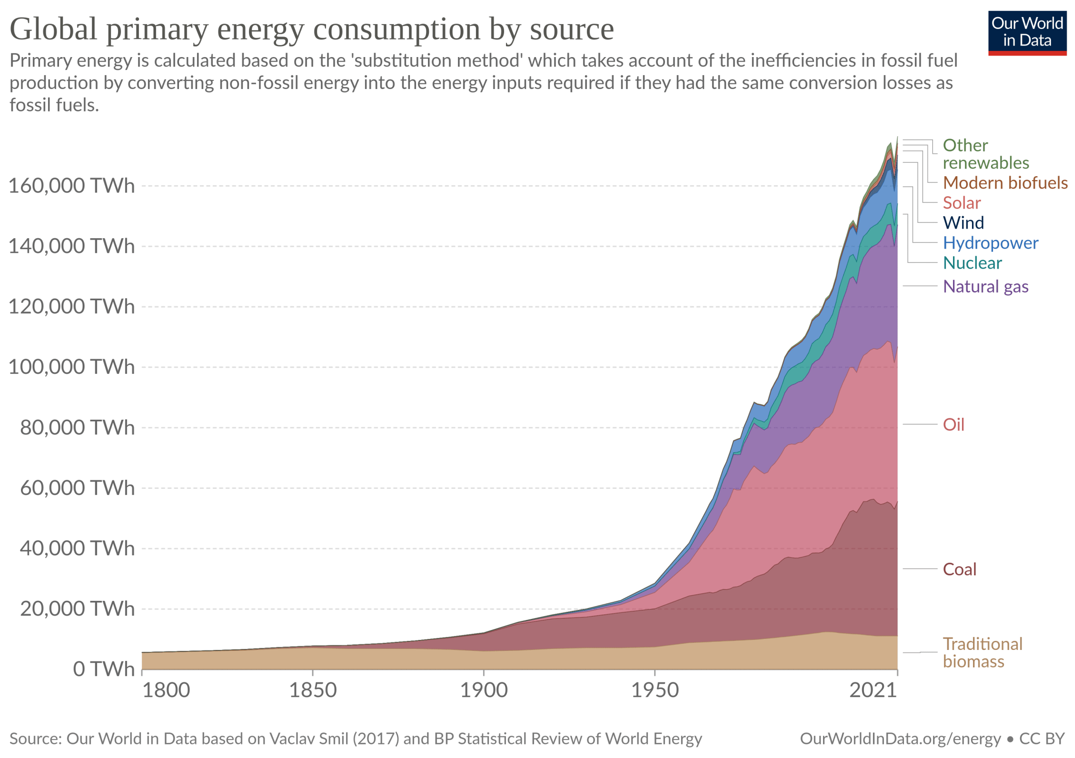


Figure 1.1: Global energy consumption by source, 1800-2021 [1].

Nuclear power plants have been delivering power to the grid in the UK since 1956. These plants produce energy by means of nuclear fission, where the nucleus of a heavy element is split into two lighter nuclei, releasing energy in the process. While nuclear fission is an extremely efficient energy source, its use is opposed by many due to the by-production of radioactive waste, which is harmful to the environment if released. This waste must be properly disposed of and monitored for hundreds, if not thousands, of years [7].

Often viewed as the ‘holy grail’ of energy sources, fusion energy can play an important role in the future energy portfolio. Conversely to fission, nuclear fusion involves fusing lighter nuclei to form a heavier nucleus, which releases between three and four times more energy than nuclear fission for the same quantity of fuel [7]. Fusion also produces no CO_2 or other harmful atmospheric emissions, meaning that it does not contribute to greenhouse gas emissions or global warming.

Achieving fusion of nuclei on Earth requires heating a plasma to in excess of 150,000,000 °C, around ten times hotter than the core of the sun. This has routinely been achieved over the past 40 years at the Joint European Torus (JET) fusion facility located at the UK Atomic Energy Authority (UKAEA) site in Culham, Oxfordshire. JET is currently the world's largest and most advanced operational tokamak, the inside of which is shown in fig. 1.2. Due to its size and design, JET will never be able to produce net energy.

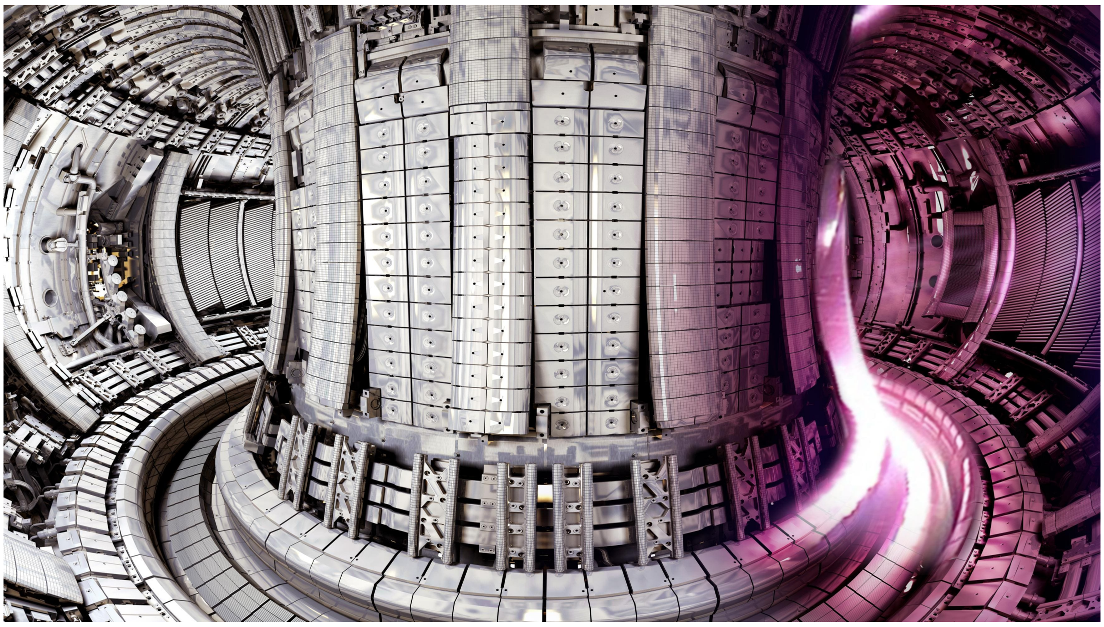


Figure 1.2: JET tokamak [8].

There are a number of key challenges which must be overcome for fusion energy to become a widely available and a financially viable energy source [9]:

1. Creation and sustainment of a controlled burning plasma.
2. Controlling the exhaust of heat and helium 'ash'.
3. Developing materials for fusion reactors, including those which are neutron-tolerant.
4. Developing components to work inside a fusion device.
5. Breeding and managing tritium, an important fuel source for fusion.
6. Achieving a high availability for the fusion plant through robotic maintenance.

These challenges span a wide variety of disciplines, from plasma physics to materials science, from chemistry to robotics, highlighting the truly multi-physics nature of fusion.

1.2 HIVE Experimental Facility

The heat by induction to verify extremes (HIVE) facility, also located at UKAEA's Culham site, was established to help tackle challenge number 4 in the above list; the development of components to work inside a fusion device [10]. Plasma facing components (PFCs) are those in closest proximity to the extremely hot plasma, and thus must be designed to withstand this hostile environment long-term. For reference, the majority of components shown in fig. 1.2 are classified as PFCs.

HIVE is a multi-physics, high heat flux (HHF) experimental facility which aims to replicate the thermal loads a PFC is subjected to during normal operation of a fusion device. Under vacuum, induction heating is used to thermally load components, while pressurised coolant is used to remove excess heat. HIVE was designed to replicate even the most extreme conditions experienced in a fusion device, which is a heat flux of up to $20\,\mathrm{MW/m^2}$.

The primary experimental data collected by HIVE is via thermocouples, which are probes that are joined on to the surface of a component that record pointwise temperature measurements. Unfortunately, there are only a limited number of thermocouples available for use in HIVE, and their placement is restricted, i.e. they cannot be placed in a location which interferes with the induction coil.

To maximise the impact that HIVE has during the research & design (R&D) of PFCs, two different opportunities have been identified; smarter testing and enhanced component insight.

1.2.1 Smarter Testing

Prior to testing, the client – those who have designed the component under testing – will specify the different conditions they desire the component to be subjected to during the experimental campaign. Examples include ensuring a prescribed temperature is delivered to a certain part of the component or subjecting the component to a large thermal gradient, all the while ensuring that the component's maximum temperature stays below its service limit. The suit of tests a component will be subjected to during an experimental campaign is known as the design of physical experiments (DoPE). Problems such as this, where the outcome is known, but the conditions needed to achieve it are not, are known as inverse problems and arise frequently in science and engineering [11]–[13].

Due to the limited diagnostics available in HIVE, it's extremely difficult for its operators to accurately assess whether these desirable outcomes have been achieved. Moreover, the experimental parameters required to deliver each outcome are identified at the point of testing through a combination of previous experience, intuition, and trial & error. An initial experiment is performed using a set of experimental parameters chosen by the HIVE operator, which are then iteratively changed until the perceived desired result is achieved. Since HIVE often tests components designed with novel features, materials and manufacturing methods, these initial guesses can be highly inaccurate. Coupling this with the large number of experimental parameters inevitably leads to performing a high number of redundant experiments.

This approach is not only expensive due to the high running costs of the facility, but also extremely time-consuming due to the need to depressurise the vacuum vessel prior to each experiment. Furthermore, the component will undergo lifecycle degradation during this iterative process, which will impact the results of the desired tests.

Due to its involvement in numerous projects within UKAEA and collaborating fusion research labs, the HIVE facility is in high demand, therefore it is essential components are tested in a high throughput manner. Along with this, relying on the intuition of experienced operators is an unsustainable approach, since this breadth of knowledge, which has been built up over many years, can be quickly lost through staff turnover, for example. It is imperative, therefore, that a faster, more robust, and cost-effective method of solving this inverse problem is found.

1.2.2 Enhanced Component Insight

To assess a component's suitability as a PFC, knowledge of its mechanical performance while subjected to large thermal gradients, caused by the induced heating and active cooling, is desirable. Unfortunately, due to operational and logistical constraints, the inclusion of diagnostic tools to measure a component's mechanical performance, such as strain gauges, is infeasible. Instead, only the temperature of the component at specific points is measured by means of thermocouples. Currently, HIVE has no method of inferring the mechanical properties of interest from the sparse thermal experimental data, meaning that only limited understanding of a component's suitability is gained from each experiment. The development of a method which uses the sparse temperature data to infer a component's mechanical behaviour would enable more informed changes to be made during the design cycle, thus reducing the number of iterations required and the time taken to develop PFCs.

Problems such as this, where the property of interest can't be measured directly but is inferred using other 'proxy measurements', arises often [14]. In economics, gross domestic product (GDP) is often used as a proxy for living standards [15], in conservation, habitat areas are used as a proxy for the degree of welfare of endangered species [16], and in marine biology coral reef health is used as a proxy to measure the bio-diversity of the marine life [17]. The work of Chen *et al.* even shows that the movement of a computer mouse can be used as a proxy for a person's level of attention [18].

The use of proxies is also commonplace in other experimental facilities. Laser flash analysis is an experiment used to measure the thermal properties of a material [19], [20]. In this experiment, a short laser pulse heats the 'top' surface of a standardised disc-shaped specimen, with the resulting temperature rise on the opposing 'bottom' face measured as a function of time using a detector. The time dependent temperature data recorded by the detector is used as a proxy to estimate the thermal conductivity of the material, which are related by means of an empirical relationship. Similarly, in a tensile test, a specimen with a standardized geometry is subjected to uniaxial loading along its length until failure [21]. The force required to uniaxially load the component along with the reduction in its cross-sectional area are used as a proxy to infer key mechanical material properties, such as its elastic modulus, yield strength, and tensile strength.

The ability to use these proxy measurements to infer more insightful information is attributed to the single mechanism nature of these experiments, along with the highly controlled environment in which they are tested, e.g. specific specimen geometry, specific loading conditions etc. The multi-physics nature of HIVE along with its variability in terms of the component design and experimental parameters means that a novel method is required to identify the properties of interest from the sparse experimental data.

The mechanical response of a component is derived from its thermal state, therefore knowledge of the temperature field throughout the component is required. Inferring the entire temperature field from a few pointwise measurements can be thought of as an extension of the inverse heat transfer problem (IHTP). The IHTP is a well known inverse problem in science and engineering, where unknown boundary conditions (BCs) or material properties are estimated using discrete temperature measurements taken within the domain of the heat conducting medium [22]. In general, the IHTP terminates with the knowledge of the unknown quantities.

An extension of the IHTP would be to use the estimated unknown quantities along with other known quantities as inputs in the forward model to generate the temperature field throughout the domain. This temperature field will have matching temperatures with those discrete measurements which were used to estimate the unknown parameters. As a result, the ability to solve inverse problems is crucial to the task of enhancing the component insight, as well ensuring smarter testing.

1.3 Inverse Problems and Machine Learning

Inverse problems are characterised by knowledge of the outcome, but not the conditions needed to achieve it. Given the highly nonlinear nature of the HIVE experiment along with the plethora of experimental parameters means that solving this inverse problem is simply too complex for the human mind. Inverse modelling, which is the term used to describe the process of gaining an inverse solution, is an extremely active area of research and can provide more accurate and robust solutions than those currently employed.

Many of the recent developments in the field of inverse modelling use supervised learning algorithms, which is a branch of machine learning (ML) [23]–[25]. ML is the science of getting computers to learn and make predictions without being explicitly programmed to do so. ML algorithms enable computers to self-learn by finding patterns and extracting features from data, which are used to make predictions on new, previously unseen data.

The theory behind supervised learning techniques is well documented in literature [26], [27]. Many of the supervised learning algorithms used nowadays were postulated a long time ago, however due to the recent development in computer hardware and the availability of machine learning libraries, such as PyTorch and TensorFlow, means their full potential is now being realised. For example, the logic of artificial neural networks (ANN) were conceptualised by McCulloch and Pitts in 1944, while regression – which is now commonly grouped in with supervised learning – has its roots as far back as the late 18th century [28], [29].

A supervised ML model uses available data to update the values of the model parameters to improve its accuracy in a process known as training. The number of parameters a model consists of will depend on the supervised learning algorithm used. In most engineering applications, the data used to train these models is synthetic. Synthetic data is generated using a computational model, commonly referred to as a simulation, of the system in question. It is extremely useful in circumstances where the real data is insufficient, and is used in most, if not all, branches of science and engineering to guide decision-making. Given the limited quantity of experimental data generated by HIVE, this work will depend entirely on synthetic data for training ML models.

While computational models have been an extremely important tool over the past seven decades, they are often computationally expensive to evaluate [30]. A popular application of ML in the field of computational science and engineering is for the development of surrogate models. Surrogates, often referred to as metamodels or emulators, use supervised learning algorithms to imitate the behaviour of computational models, providing orders of magnitude speed up in their evaluation time [31]. This faster evaluation time makes them useful for a number of applications, including sensitivity analysis, 'what-if' scenarios, and optimisation problems, commonly referred to as surrogate assisted optimisation (SAO) [32].

Surrogates can also be used to solve inverse problems in an approach known as indirect inverse modelling, which is a branch of SAO and is discussed in more detail in section 2.2 [33]. Much of the SAO work in literature is insufficient for the problems faced by HIVE. Often, these problems are highly idealised, e.g. specimens with simple geometries made from single materials or 2D problems based on single mechanisms, e.g. thermal or mechanical [34], [35]. HIVE, on the other hand, will test multi-material components with complex geometries in a highly-nonlinear, multi-physics, real world setting. There are also a wide variety of different unknowns which HIVE will need to estimate, whereas the work in literature tends to revolve around estimating at most two different parameters [24], [36].

The majority of SAO work in literature uses single value (SV) surrogates [37]. The outputs of SV surrogates are key metrics extracted from the simulation results, e.g. maximum temperature or stress at a specific location [37]–[39]. The advantage of SV surrogates is that the small number of outputs means that training and evaluating the ML model is fast, however a new model will need to be generated if the desired outcome changes. Full field (FF) surrogates, on the other hand, predict entire results fields generated by a simulation, e.g. predicting the temperature at the nodes of a mesh [40], [41]. These offer more flexibility since a variety of different key metrics can be extracted from the results field, however they come at an increased computational cost since the number of outputs can be very large.

The use of FF surrogates for inverse modelling, and indeed SAO in general, in literature is extremely limited. This work will investigate their use for solving inverse problems along with SV surrogates to compare their performance. This work will also investigate the use of dimensionality reduction techniques to reduce the computational burden of generating FF surrogates. While many supervised learning algorithms have been used for the generation of surrogate models, with an excellent review given by Kianifar *et al.*, this work will use Gaussian process regression (GPR) and multi layer perceptron (MLP), a type of ANN [42].

1.4 Aims and Objectives

The primary aim of this thesis is to investigate the use of surrogate models in solving inverse problems for a highly complex, nonlinear, real world testcase to maximise its impact on the design of PFCs. As HIVE continually tests components of different geometries and materials, a different surrogate model will be required for each. This means that new synthetic data will need to be generated on a case-by-case basis. Also, given that HIVE is in high demand, it is essential that the required analysis is performed in a timely manner. Achieving this will require the development of a fully automated workflow which takes advantage of parallelisation and high performance computing (HPC) capability to provide rapid solutions.

The secondary aim is to develop a method for optimising the location of thermocouples for each component tested. Currently, all available thermocouples are used in each experiment and are arbitrarily placed. Joining the thermocouples to the component is a very time-consuming process, with each individual thermocouple taking around half an hour to join. Identifying an optimal configuration of thermocouples could result in more insightful data, and possibly reduce the number required, enabling experiments to take place more rapidly.

To achieve these aims, the following objectives are identified;

1. Provide a review of the recent work and developments surrounding inverse modelling with ML and sensor optimisation.

2. Develop a framework which supports the automation of entire workflows, and utilises HPC systems for improved parallelisation.

3. Develop a robust and accurate computational model for the HIVE facility.
4. Investigate the accuracy of different ML algorithms for the generation of surrogate models and their ability to identify inverse solutions.
5. Demonstrate the improvements in the DoPE and component insight using this methodology.
6. Demonstrate a proof of concept for inferring the stress state of components from sparse, discrete temperature measurements.
7. Develop a method of optimising the placement of thermocouples.

The research carried out in order to achieve these objectives is split into several stages, which are described in the outline of the thesis. This research has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 and 2019-2020 under grant agreement No 633053 and from the RCUK Energy Programme [grant number EP/I501045] and EPSRC [grant number EP/R012091/1]. The research was conducted under the supervision of Dr Llion Evans and Prof. Perumal Nithiarau, both of Swansea University, and Dr Andrew Davis of UKAEA.

1.5 Outline of the Thesis

This thesis is formed of 9 chapters and is complemented by an appendix. The organisation of these is as follows:

- **Chapter 1: Introduction** presents a brief introduction to fusion energy and its role in the future energy portfolio, along with its challenges and motivation for the HIVE facility. The current limitations of HIVE are used as motivation for this thesis, specifically the construction of a comprehensive DoPE and estimation of key properties without relying on the need for previous experience and intuition. The aim and objectives of this thesis are then stated.
- **Chapter 2: Review of Current Methodologies** provides a review of inverse modelling methods used in literature, especially those which use surrogate models. Both the use of SV and FF surrogate are discussed, along with the dimensionality reduction techniques which are commonly

employed in the latter. Following this, the literature surrounding the optimisation of sensor placements is also reviewed to harness the state-of-the-art methods for optimal data collection using thermocouples. This chapter also presents an overview of popular optimisation algorithms, with the novel *MultiSLSQP* package presented. Along with this, the difficulty currently faced in establishing automated workflows for computational engineering purposes is discussed, highlighting the need for the development of the *VirtualLab* package.

- **Chapter 3: Surrogate Model Generation** presents the theory behind the supervised learning algorithms used to generate surrogate models for this work. This chapter also presents the theory of dimensionality reduction techniques used in this work, providing examples of their use for both linear and nonlinear data. This chapter also discusses the different data sampling strategies available in literature for collecting simulation data to train surrogate models, with both pre-determined and adaptive methodologies presented.

- **Chapter 4: The HIVE Facility** gives a more detailed discussion on fusion energy, its challenge with regard to the exhaust system and need for a facility such as HIVE. This section discusses the setup of the HIVE facility, including its multi-physics loading and available diagnostic tools. Here, common desired experimental outcomes are discussed, along with the experimental parameters which affect the heating of a component and their sensitivity.

- **Chapter 5: VirtualLab** presents the *VirtualLab* platform which has been developed for this thesis to provide an automated workflow for synthetic data generation and analysis. A detailed description of the simulation created within *VirtualLab* for the HIVE experimental facility is presented, which includes the use of clustering algorithms to reduce the computation times substantially. The developed simulation is validated using experimental data, showing a high degree of accuracy.

- **Chapter 6: Optimisation of Hardware Configuration** uses the supervised learning algorithms discussed in chapter 3 to construct SV and FF surrogates to model the impact of the coil design and positioning in terms of the thermal loading delivered to a component. The SV surrogate models reduce the heating profile generated by the FE model to two values; the power delivered to the component and the variation of the heating profile. The FF surrogate models aim to replicate the heating profile on the surface adjacent to the induction coil. These models are then used to inversely identify a range of experimental parameters which deliver desirable outcomes.

- **Chapter 7: Smarter Component Testing** builds on the knowledge gained in chapter 6 to construct three-dimensional FF surrogate models for both the temperature and Von Mises stress field. These are used to identify the experimental parameters which deliver a variety of desirable behaviour in the component, enabling a more comprehensive DoPE to be generated. These surrogate models are also used to assess the sensitivity of the experimental parameters on the results. To showcase the developed workflow, the same analysis is performed on an alternative component with a different design and materials to highlight that the same level of insight can be gained without any prior knowledge of the component other than its design.

- **Chapter 8: Enhancing Sparse Experimental Data** presents a methodology for using the thermocouple data recorded by HIVE to enhance the understanding of a component's strengths and limitations. First, this data is used to infer both the temperature and stress field throughout the component. Following this, the placement of the thermocouples is optimised, reducing the number required. A method of adding prior knowledge of the experiment in to the methodology is presented, which helps reduce the number of thermocouples required further. Finally, the experimental data is used to estimate the values of uncertain parameters relating to the component design. This is demonstrated here for the thermal efficiency of the bond between two surfaces, however its use can be applied to any number of other uncertain parameters relating to the manufactured state of the designed component.

- **Chapter 9: Conclusions and Future Work** presents an overview of the work and the main achievements of the thesis, along with a discussion on how this work is of benefit to the wider scientific community. A summary of the most useful tools developed for this thesis are also presented, and their use in other research projects. Finally, a discussion on future work which could improve the methodology and frameworks for use in other applications is discussed.

- **Appendix A:** presents specific scripts extracted from the *VirtualLab* package which are discussed during the thesis.

1.6 Research Outcomes

This section presents a list of the research outcomes from the work performed as part of this thesis.

1.6.1 Contributions

This thesis has demonstrated that surrogate models can be used to solve a variety of inverse problems for a challenging engineering problem. In chapter 7 it is shown that surrogates can advise a substantially more comprehensive DoPE for a component to assess its suitability, even taking into account the stress state of the component. This is demonstrated for two very different components to highlight that this insight is possible without the need for any prior knowledge of the component.

This thesis has also demonstrated that surrogate models can be used alongside proxy measurements to infer highly elusive properties in a complex use case. This is demonstrated in section 8.2, where it is shown that as few as four pointwise surface temperature measurements are sufficient to infer the temperature and stress field throughout the component, thus providing a much better understanding of the component.

The methodology described in this thesis has resulted in the development of two new open source packages; *VirtualLab* and *MultiSLSQP*. *VirtualLab* is a fully-automated, parallelised and portable framework for rapid generation of synthetic data and subsequent analysis, which is discussed in more detail in chapter 5 [43]. Along with the software itself, there is considerable support offered around the package in the form of a comprehensive set of documents [44]. These documents cover the installation of the package, tutorials and guide on how to contribute. In addition to the software, four different datasets generated for this thesis have been made publicly available [45]. These consist of data extracted from simulations which were used to train surrogate models.

MultiSLSQP is an extension of SciPy's sequential least squares programming (SLSQP) optimisation algorithm to support the input of multiple initial points, and is discussed in section 2.4.3. Harnessing the use of tensor multiplication reduces the computational burden of performing tasks individually, enabling a substantial improvement compared with a sequential strategy. This is especially useful in scenarios where evaluation of the objective function is computationally expensive to evaluate. *MultiSLSQP* has been developed so that it's compatible with any version of SciPy from 1.4.0 to the most recent version (1.11.0), and requires no additional python packages to those required by SciPy [46].

1.6.2 Conference Papers and Presentations

- **Rh. Lewis**, Ll.M. Evans, R. Otin, K. Leng , A.D.L. Hancock, A. Davis, J. Thyiagalingam, P. Nithiarasu. Lab Experiment Optimisation Using Coupled Finite Element Analysis and Machine Learning. *UK Association for Computational Mechanics Conference*, Loughborough, United Kingdom. April 2021.

- **Rh. Lewis**, Ll.M. Evans, R. Otin, A.D.L. Hancock, A. Davis, P. Nithiarasu. Using Gaussian Process Regression with Coupled Multi-physics FEA Simulations to Enhance Sparse Experimental Data. *European Community on Computational Methods in Applied Sciences*, Oslo, Norway. June 2022. Keynote in inverse modelling stream.

- **Rh. Lewis**, Ll.M. Evans, R. Otin, A.D.L. Hancock, A. Davis, P. Nithiarasu. Optimisation of Sensor Placement for HIVE Experimental Facility using Data Driven Surrogate Models. *FuseNet PhD Event*, Padova, Italy. July 2022.

- **Rh. Lewis**, Ll.M. Evans, R. Otin, A.D.L. Hancock, A. Davis, P. Nithiarasu. Machine Learning Guided Optimisation of Sensor Placement for HIVE Experimental Facility. *Symposium on Fusion Technology*, Dubrovnik, Croatia. September 2022.

- **Rh. Lewis**, B. Thorpe , Ll.M. Evans. VirtualLab: A fully automated, open-source platform for virtual experiments. *Image Based Simulation 4i*, London, United Kingdom. October 2023.

1.6.3 Workshops

- A one-day workshop demonstrating the *VirtualLab* package for use in image based simulation workflows. *Image Based Simulation 4i*, London, United Kingdom. October 2021.

1.6.4 Research Posters

- **Rh. Lewis**, Ll.M. Evans, R. Otin, A.D.L. Hancock, A. Davis, P. Nithiarasu. A Virtual Twin of UKAEA HIVE Experiment for Improved Analysis of Divertor High Heat Flux Testing. *Symposium on Fusion Technology*, Dubrovnik, Croatia. September 2020.

- **Rh. Lewis**, Ll.M. Evans, R. Otin, A.D.L. Hancock, A. Davis, P. Nithiarasu. Coupling Simulation with Machine Learning to Enhance the HIVE Experimental Facility. *FuseNet PhD Event*, Padova, Italy. July 2022.

- **Rh. Lewis**, Ll.M. Evans, R. Otin, A.D.L. Hancock, A. Davis, P. Nithiarasu.Optimisation of Design of Physical Experiments for the HIVE Experimental Facility using 3D GPR surrogate models. *Symposium on Fusion Engineering*, Oxford, United Kingdom. July 2023.

Chapter 2

Review of Current Methodologies

2.1 Introductory Remarks

The aim of this chapter is to provide a review of the state-of-the-art of two research topics integral to this thesis; inverse modelling and sensor placement optimisation. As discussed in chapter 1, both the goal of smarter testing and enhanced component insight in HIVE can be achieved by solving an inverse problem, therefore this in an integral component of this thesis. Inverse modelling is still a very active area of research, therefore this chapter will provide a review of the most recent work presented in literature in section 2.2, with the primary focus on those that use surrogate models.

The second aim of this thesis is to develop a method of optimally locating the sensor placements in HIVE, ensuring only the most valuable data is recorded. This has close parallels to the field of IHTP, where sensor optimisation has been identified as an avenue of future research [23]. Section 2.3 will provide a review of the use of sensor placement optimisation in other fields and identifying those applicable to HIVE and the wider IHTP field.

As optimisation plays an important role in both the inverse modelling and sensor placements, an overview of popular optimisation algorithms is given in section 2.4, with the novel *MultiSLSQP* package presented [46]. *MultiSLSQP* enables a fare more efficient search of the parameter space for the identification of a more optimal solution.

Finally, section 2.5 provides an overview of computational modelling techniques and the issues faced in generating vast quantities of synthetic data.

2.2 Inverse Modelling

Inverse problems are defined as the identification of a set of conditions $\mathbf{x}^* = \{x_1^*, ..., x_i^*\}$ which achieve a certain outcome $\mathbf{y}^* = \{y_1^*, ..., y_j^*\}$ for a process or system which maps $\mathbf{x}$ to $\mathbf{y}$. Depending on the application, an equation for the forward model, $\mathbf{f}$, for this mapping may or may not be known.

Inverse problems, inverse modelling and inverse methods are overlapping, amorphous terms which are used interchangeably depending on the field in question. For clarity, this work uses the term inverse problems to describe the problem statement, e.g. what the desired unknown parameters are for the given outcome, while inverse modelling refers to the way in which these unknown parameters are estimated.

One of the reasons inverse modelling has received much attention in literature is due to the ill-posedness of these types of problems. Any problem which is not well-posed is referred to as ill-posed. The idea of well-posedness was defined by Jacques Hadamard in 1902 and states that any problem must have the following properties to be considered well posed [47]:

1. a solution exists,
2. the solution is unique, and
3. the solution depends continuously on the initial values.

For inverse problems, condition 2 is violated for all but the most trivial examples, as multiple combination of inputs can produce the same outcome (non-uniqueness). Condition 3 is also often violated, as slight changes in the outcome will often lead to large changes in the inverse solution (ill-conditioning).

2.2.1 Direct and Indirect Methods

Many modelling techniques have been researched for solving inverse problems, which can be grouped in to two distinct categories; direct and indirect methods [48].

Prior to computers becoming available, direct methods were the only practical method to solve inverse problems [33]. The approach of direct methods is to transform from the data space into the model space view [48]. That is, an inverse model is sought who's inputs are the outputs of the forward model, while its output is the forward model's input, see fig. 2.1. Using this inverse model, the desired conditions, $\mathbf{y}^*$, are input to the model, resulting in the inverse solution, $\mathbf{x}^*$. Due to the aforementioned ill-posedness issue of inverse models, regularisation must be added, the most common of which is the Tikhonov regularisation [49].

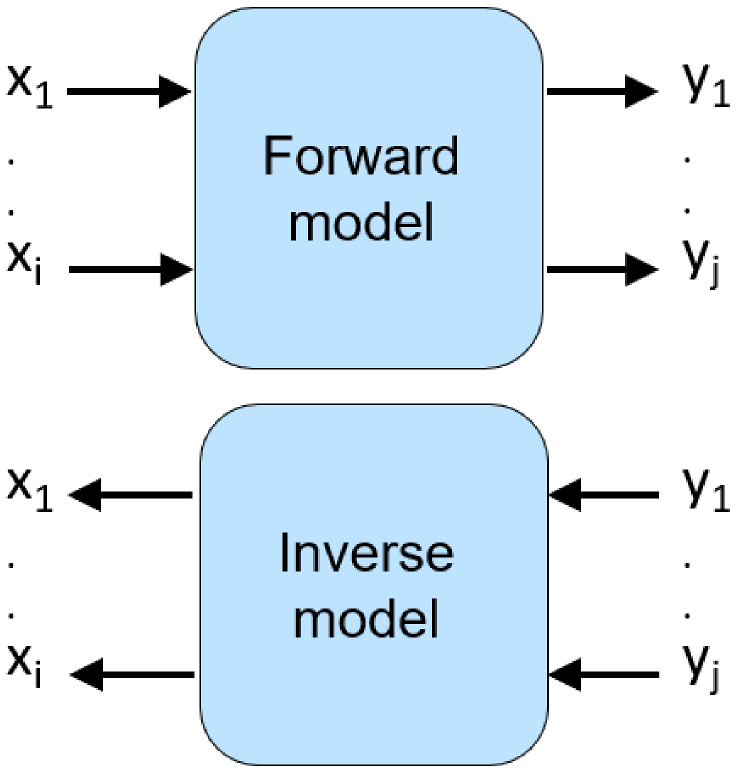


Figure 2.1: Forward and inverse models

Direct methods have the advantage of delivering rapid solution to inverse problems, which is especially useful in scenarios where speed is important, such as robotics [12]. Originally the inverse mapping was achieved through a combination of exponential, power law and polynomial shape functions that best fit the data. An example of this can be found in the work of Jaluria, where a number of different thermal systems are investigated [50].

More recently, ML algorithms have been used to create the mapping for these inverse models. In the work of Wanigasekara *et al.* a shallow MLP was used to identify the ideal conditions to manufacture a thermoplastic laminate with desirable characteristics [51]. The work of Tamaddon-Jahromi *et al.* used a deeper MLP, often referred to as deep neural networks, to solve the IHTP [23]. Here, the focus was on the 2D diffusion and convection-diffusion problems in a square domain. Toquica *et al.* investigated the accuracy of three variations of ANNs to solve the inverse kinematics problem of a robotic arm [25].

Nowadays, indirect inverse methods are more commonly used than direct methods [33]. Instead of creating an inverse mapping, this method identifies $\mathbf{x}^*$ using an optimisation algorithm to minimise the difference between the output of the forward model, $\mathbf{f}$ and $\mathbf{y}^*$. This is formulated in eq. 2.1 where L_f is the objective function of the optimisation problem and $\mathcal{P}$ is the i-dimensional search space where solutions are sought.

$$\begin{aligned} \mathbf{x}^* = \arg\min_{\mathbf{x}} \quad & L_f(\mathbf{f}(\mathbf{x}), \mathbf{y}^*) \\ s.t.\ \mathbf{x} \in \mathcal{P} & \end{aligned} \tag{2.1}$$

The form of L_f will depend on the problem, however when $\mathbf{y}^*$ are real valued outputs this is commonly the squared error between the two, which is given by eq. 2.2. Minimisation of this objective function is often referred to as the least squares error (LSE) approach.

$$L_f^{SE}(\mathbf{f}(\mathbf{x}), \mathbf{y}^*) = ||\mathbf{f}(\mathbf{x}) - \mathbf{y}^*||^2 \tag{2.2}$$

The squared error objective function is commonly used since its derivative with respect to $\mathbf{x}$ is easily derived, which is given in eq. 2.3. This result means that the objective function's differentiability depends on the differentiability of the forward model with respect to $\mathbf{x}$. This is an important result, since having a well-defined gradient for the objective function means that gradient-based optimisation algorithms can be used. An overview of gradient-based and gradient-free algorithms are provided in section 2.4.

$$\frac{d}{d\mathbf{x}} L_f^{SE}(\mathbf{f}(\mathbf{x}), \mathbf{y}^*) = 2(\mathbf{f}(\mathbf{x}) - \mathbf{y}^*)\frac{d\mathbf{f}}{d\mathbf{x}} \tag{2.3}$$

Indirect methods provide a great deal more flexibility compared with direct methods, enabling the exploration of various solutions to the inverse problem (where multiple valid solutions exist). This is extremely advantageous for the construction of a DoPE in HIVE, where knowledge of several combinations of experimental parameters which deliver the same desired outcome enables the operators to decide which to use based on other external factors. For example, consider that there are two different combinations of experimental parameters which will deliver the same outcome, both of which use a different induction coil, then to save time, the experimental parameters with the induction coil which is already in place would be chosen. This is an example of human-machine collaboration guiding, but not making, decisions [52].

2.2.2 Surrogate Assisted Optimisation

In the majority of applications in science and engineering, the forward model $\mathbf{f}$ is an expensive to evaluate computational model of the system in question. This means that their use in optimisation algorithms can be extremely time-consuming. An example of this is given in the work of Dabrowski and Dabrowski, where thermocouple data was used alongside a computational model to estimate the heat flux a rocket surface is subjected to [53]. This analysis took over 3 days to complete, with only a single unknown parameter estimated. This explains why most of the work carried out in literature use surrogate models in a field known as SAO.

In the majority of cases, the surrogate models generated for the purpose of SAO are SV surrogates. Jin *et al.* compared the use of polynomial regression, GPR and radial basis function (RBF) to ascertain the diameter and height of a two bar structure which minimises its volume [36]. Values for these two variables are also sought which minimise the uncertainty of the system. Varhan *et al.* use a random forest based surrogate model to identify geometric parameters which maximises the efficiency of a propeller design [24]. Zhu *et al.* use GPR to reduce the weight of an automotive front-body structures by modelling its structural crashworthiness performance indicators [54].

There are occasions where the number of outputs a SV surrogate model must predict is large. An example of this are transient problems, where data is extracted from the computational model and different times [55]. Many supervised learning algorithms are unsuited to modelling a large number of outputs, therefore a common approach is to reduce the dimensionality of the output. The most popular approach for this is to project the high dimensional data on to a lower dimensional, linear subspace, with is achieved using a principal component analysis (PCA), also termed as proper orthogonal decomposition (POD), of the data. This approach, which is outlined in more detail in section 3.3.1, identifies the principal components (PC) of the data on which to project it, thus reducing its dimensionality.

In the work of Khaledi *et al.* the vertical displacement is recorded at 9 locations at 45 different steps for a tunnel excavation, resulting in 405 outputs which need to be modelled [56]. PCA is used, with the subsequent reduced outputs modelled using RBF. In the work of Havinga *et al.* PCA and GPR is used to reduce the dimensionality of the output during the process of metal forming [57]. In the work of Kato *et al.* computational fluid dynamics (CFD) analyses are replaced by a POD and RBF-based surrogate model to optimize turbine blades [58].

Currently, FF surrogates are not commonplace, especially for SAO application. One possible reason for this is the difficulty in generating models with a very large number of outputs. For example, a simple cube discretised in to 10 elements in each direction would consist of 1331 nodes, with the value (e.g. temperature) at each accounted for by the surrogate model. A mesh for a HIVE component will consist of tens or hundreds of thousands of nodes.

There are examples of the use of dimensionality reduction techniques for predicting FF surrogates in literature. Dupuis *et al.* used a combination of GPR and PCA to model the pressure coefficient on the 2D external surface of an AS28G aircraft [59]. Li *et al.* similarly uses GPR, but with a modification of the PCA algorithm, for predicting the thermal behaviour during a three-layer printing process for laser powder bed fusion [60].

While the PCA algorithm is extremely powerful and easy to implement, it projects the data on to linear subspaces, making it poorly suited for certain types of data [61]. Auto-encoders are a special type of ANN algorithm which are used to efficiently learn a lower dimensional encoding of a dataset. Since ANNs underpin auto-encoders, they are able to identify nonlinear relationships between high dimensional data and its lower dimensional representation. Auto-encoders have been used for a variety of tasks, including image compression [62], time series data [63] and denoising data [64], however its use for dimensionality reduction for FF surrogate model construction has thus far not been established.

While most supervised learning algorithms will require the use of dimensionality reduction for constructing FF surrogates, for ANNs this is not necessarily the case. ANNs are able to model numerous outputs, meaning they can be used for FF surrogates without dimensionality reduction, however these models would consist of a huge number of trainable parameters, making it slow to train and prone to overfitting if the training data is not sufficiently large.

A few different types of ANNs have been used in literature to generate surrogates. Recurrent neural networks (RNNs) are used for the prediction of sequences, meaning they were originally developed for tasks such as natural language processing [65], however more recently they have been applied to the field of FF surrogate modelling [66]–[68]. Modelling stationary or steady-state problems can be achieved using MLPs, as it has been in the work of Liang *et al.* [69]. Since the focus of this work is on the steady-state behaviour of PFCs tested in HIVE, MLPs would be the most practical in terms of the ANNs available.

The supervised learning algorithms discussed thus far are solely data driven, which require no underlying understanding of the physics involved, and instead just learn an optimal mapping between the inputs and target values. An active area of research these days is to incorporate the governing equation of the system as part of the training of the surrogate model. These physics informed methods have the advantage of needing fewer, if any, data points to train the model. The most well known of these are physics informed neural networks (PINNs), an overview of which is given by Sharma *et al.* [70].

Billah *et al.* use a PINN to solve the IHTP, however this is for a 1D problem only [34]. While the author state that PINNs can be used to solve extremely complex problems, this paper only demonstrates their use for a simplistic 1D problem. Moreover, the PINN used in this work took more than 10 hours to train, meaning that this would be substantially larger for more complex problems in 2D or 3D. A PINN is used by Manavi *et al.* to construct a 2D surrogate model on rectangular square domain for the IHTP, however no information is provided in terms of it's training time [71]. While PINNs offer a huge amount of potential in surrogate modelling, their development is still in their infancy, meaning that they are not yet applicable to tackling a problem as complex as HIVE.

None of the work presented thus far in literature is comparable in complexity to that of HIVE. HIVE tests multi-material components with complex geometries in a highly-nonlinear, multi-physics, 3D problem, while the work presented thus far in literature have been for simple geometries and problems e.g. 2D geometries, single mechanism problems etc. Along with this, the greatest number of parameters estimated in the above work is two, whereas the HIVE experiment consists of estimating substantially more unknowns. The use of FF surrogates in literature is also limited, especially in the context of SAO. This work will investigate the development and use of such models, including the success of the PCA and auto-encoders techniques in reducing their computational expense.

GPR has shown to perform excellently for engineering based surrogate models, usually requiring fewer data points compared with many of its competitors [36], [54], [72], [73]. MLP are an extremely popular choice for supervised learning applications and have shown excellent performance in extracting high level features from data, therefore these will also be investigated [69].

2.3 Sensor Placement Optimisation

As discussed in section 1.2, HIVE is only able to record a small quantity of experimental data via thermocouples. Fitting each thermocouple is a very time-consuming process, with each taking around 30 minutes to join successfully. Currently, all the available probes are joined to a component prior to an experiment at arbitrary locations decided by the HIVE operators. These locations are chosen based on the previous experiences of HIVE operators, which is an unsustainable approach. Moreover, given the complexity of the problem, it's unlikely that these thermocouples are placed in an optimal configuration to extract the most insightful data. Identifying an optimal configuration of thermocouples would not only provide better data, but could also reduce the number required, thus speeding up the frequency in which components are tested.

The work of Tamaddon-Jahromi *et al.* highlighted the sensitivity of the estimated BCs on the placement and quantity of data used for the 2D IHTP [23]. While they identify optimisation of the used data as future work, this has yet to be covered in the literature. This work aims to create a generic methodology which can be used for HIVE along with other types of IHTPs.

The thermocouples used in HIVE are somewhat similar to a sensor network (SN). SNs consist of a network of sensor devices, where each device can autonomously sense the target environment and communicate with other sensors to achieve the goal of delivering valuable information to the end user. In recent years, SNs have been used in a wide variety of settings, including structural health monitoring, environmental monitoring and factory automation [74], [75]. While the thermocouples in HIVE aren't able to communicate with each other, their combined information can provide a huge amount of insight about a system, much like a SN does.

It has been shown that the optimal placement of sensors is an important design consideration, providing improved performance of the SN [76]. As a result, this topic has received much attention in literature. In the majority of cases, gradient-free optimisation algorithms are employed, since the objective function does not have a well-defined gradient. In the work of Akbarzadeh *et al.* an evolutionary strategy is used to identify the optimal locations of sensors for a wireless network

[77], while Jourdan and de Weck used a genetic algorithm for a similar problem [78]. In the work of Yi *et al.* particle swarm optimisation (PSO) is used to identify the optimal locations of sensors in a high rise building for structural health monitoring purposes [74]. Simulated annealing was used by Chiu and Lin to solve some benchmark problem relating to sensor placements in a 2D domain [79].

Interestingly, Xu *et al.* use a combination of PCA and Gaussian processes to optimise the placement of temperature sensors in the Berkeley Intel laboratory [80]. Temperature data was recorded at 50 equally spaced times over a period of two months for 54 different locations in the lab, which are effectively points in a 2D domain. PCA is used to identify the PCs of the spatial data, which are subsequently modelled as Gaussian processes and used with the branch and bound optimisation algorithm.

For the Berkeley lab problem, the branch and bound method is applicable, since there are a finite number of combinations of sensor placements. For HIVE, an alternative optimisation algorithm is required, such as evolutionary strategies or genetic algorithm, since there are infinitely many combinations due to the continuous nature of the problem. These types of algorithms perform a large number of evaluations, therefore the performance of numerous configurations of thermocouples will need to be assessed. To achieve this in a timely manner, FF surrogate models will be beneficial to support the rapid evaluation of various thermocouple configurations.

2.4 Optimisation Algorithms

The use of optimisation can be found in most, if not all, industries. In the aerospace industry it is used to improve aerodynamic performance of an aircraft [81], in manufacturing it is used in the design of production lines and minimising waste [82], [83], while in logistics it is used for scheduling and route planning [84], [85], to name only a few. This work also requires the identification of optimal solutions, both for the task of identifying inverse solutions and sensor placement.

The formal definition of an optimisation problem is given in eq. 2.4, where L_f and $\mathcal{P}$ are like those defined in eq. 2.1 and h_e and g_r are optional equality and inequality constraints, respectively. The inclusion of constrains in an optimisation problem means that an optimal solution is identified while ensuring that the constraints are satisfied.

Constrained optimisation problems arise frequently, such as in aerospace, where the goal is to reduce the drag of an aircraft while ensuring the internal volume of the aircraft stays above a certain value [81]. Maximising the volume of the aircraft enables increasing capacity, thus increasing revenue, while minimising the drag will reduce the quantity of fuel needed, thus reducing costs. However, there is a trade-off between the two, because drag will increase with the cross-sectional area of the aircraft [86].

$$\begin{aligned} \mathbf{x}^* = \arg\min_{\mathbf{x}} \quad & L_f(\mathbf{x}) \\ s.t.\ \mathbf{x} \in \mathcal{P} & \\ h_e(\mathbf{x}) = 0, \quad & e = 1, ..., E \\ g_r(\mathbf{x}) \geq 0, \quad & r = 1, ..., R \end{aligned} \tag{2.4}$$

Since optimisation is such a widely used discipline, there are a variety of optimisation algorithms available in literature. These algorithms can be divided into two categories; gradient-free and gradient-based algorithms.

2.4.1 Gradient-free

In a number of scenarios, the derivative of the objective function does not exist. Fortunately, there are a family of optimisation algorithms available which do not rely on any gradient information and are known as gradient-free algorithms. These are sometimes preferred to the gradient-based algorithms even if the gradient is available, because these can avoid getting trapped in local minima [87].

A number of these types of algorithms follow a similar structure; start off with an initial number of guesses, evaluate the value of the objective at these points, and then use these to guide the next set of points to sample. Evolutionary strategies, particle swarm optimisation and simulated annealing all follow a similar pattern and differ in how they choose the next set of points from the previous set.

For operational research and computer science problems, the genetic algorithm is especially popular [88]. This algorithm work on the idea of survival of the fittest and is a subclass of evolutionary strategies. An initial population (generation 0) of N points are chosen, with the 'fitness value' for each member calculated using the objective function. The $L < N$ best performing members of the population, that is those with the highest or lowest fitness values, are chosen as parents for mating.

From the pool of parents, the next generation is produced. The first step is known as crossover, which involves combining two parents to create an offspring. One way in which this is achieved is by encoding each parent as a genetic sequence, and then choosing genes from each parent randomly. The next step is mutation, where genes in the offspring's genetic sequence are randomly altered based on a defined criterion. The offspring's genetic sequence is then decoded back in to its original form. New offspring are produced until there are N members in the new generation (generation 1), with the option to include the parents of the previous generation as members of the next. This procedure is followed until some stopping criterion is met, such as a set number of generations or convergence of the solution. A visualisation of the steps required for the genetic algorithm is shown in fig. 2.2.

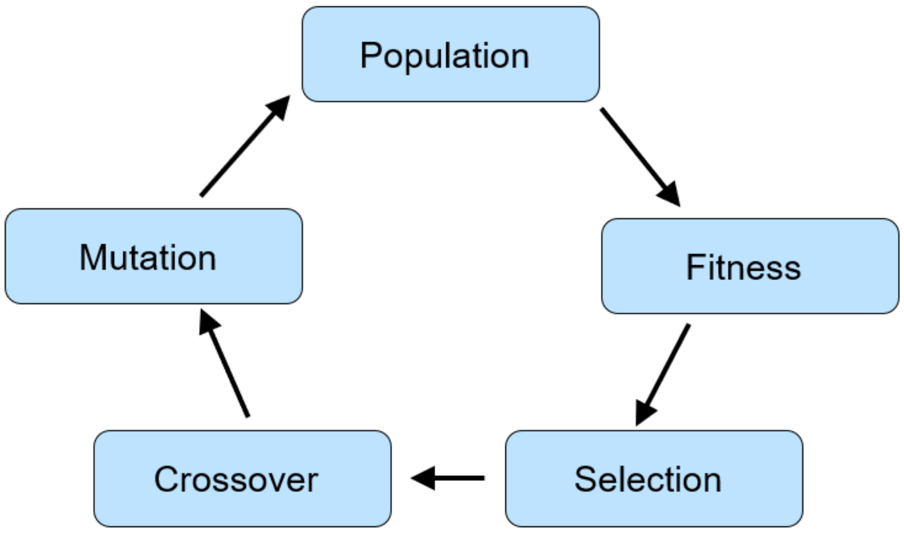


Figure 2.2: Iterative loop of the genetic algorithm

PyGAD is a python package which provides an easy-to-use class for performing optimisation via genetic algorithm [89]. This package has a multitude of options available for crossover, mutation, and parent selection, as well as supporting custom-made functions for each. Along with this, it supports optimisation of parameters which take discrete or continuous values.

An important aspect of any gradient-free algorithm is a combination of exploration and exploitation of the search space. Exploration is the search of new areas in the search space, while exploitation is where the neighbourhood of a promising solution is searched more thoroughly. A balance of the two is key, as too much exploitation will likely result in the solution being trapped in a local optima, while too much exploitation means that a large amount of time is wasted on poor solutions by ignoring the information previously gathered. In the case of the genetic algorithm, mutation ensures that the algorithm has exploration properties, while crossover provides exploitation by combining good solutions.

2.4.2 Gradient-based

In general, gradient-based algorithms are faster for solving optimisation problems compared with gradient-free algorithms [90]. This isn't all that surprising, given that a large amount of information is stored in the gradient. They also have fewer, if any, hyperparameters which must be decided compared with gradient-free algorithms.

A popular method used to calculate this derivative is the adjoint-state method [91]. This is an efficient way to calculate the gradient of the objective, which is independent of the number of parameters for which the gradient is required (the dimension of $\mathbf{x}$). This approach can also be applied to constrained optimisation problems.

The information gained from the gradients is used to define a search direction to move the solution forwards towards an optima. The oldest and most intuitive of these algorithms is gradient descent (GD) [92]. As the gradient points in the direction of the greatest ascent, this algorithm updates the current point by taking a step in the direction of the negative gradient, thus moving towards the minima via the steepest descent. This algorithm is shown in eq. 2.5, where the parameter α is a positive number known as the learning rate, which dictates the step size taken at each iteration. The learning rate is the cause of many of the issues faced using this algorithm. If the value is too large the solution oscillates around the minima (and can even diverge), while if it is too small the algorithm takes many iterations to converge, which is extremely inefficient.

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \alpha \nabla L_f(\mathbf{x_k}) \tag{2.5}$$

The Momentum algorithm [93] was developed to reduce the issue of oscillations of the solution near the optima by adding a fraction to the first term in eq. 2.5. This factor ensures that the gradients are increased for dimensions where the gradient points in the same direction (thus gaining momentum), while reducing it for dimensions whose gradient alternates direction.

Given there is no way of knowing the 'best' learning rate in advance, a number of adaptations to GD have been proposed in the literature which aim to overcome this. The Adagrad algorithm applies adaptive learning rates to different dimensions to speed up training [94]. This removes the need to tune the learning parameter α and it is usually set to the value 0.01. The Adadelta algorithm is an adaptation of Adagrad which aims to improve its performance [95]. The adaptive moment estimation algorithm, commonly referred to as Adam, is another popular algorithm which is similar to Adadelta but also includes aspects of the Momentum algorithm to improve its performance [96]. A number of other adaptations to the vanilla GD algorithm include RMSprop, AdaMax and Nadam [92], [96], [97].

The GD algorithm and its variants discussed thus far use the first order gradients, however there are a variety of algorithms which take advantage of higher order derivatives to optimise the objective. The Newton-Raphson method uses second order derivatives which removes the need for a learning rate, as shown in eq. 2.6, where H is the Hessian of the objective function [98]. While this method usually requires much fewer iterations than first order methods, calculating the Hessian and its inverse scales with $O(n^3)$, meaning it becomes extremely inefficient in high dimensional problems [99].

$$\mathbf{x_{k+1}} = \mathbf{x_k} - H(\mathbf{x_k}))^{-1} \nabla L_f(\mathbf{x_k}) \tag{2.6}$$

To overcome the issues relating to the Hessian, a class of optimisation tools known as quasi-Newton methods were developed. These aimed to approximate the Hessian using a matrix B which is updated at each iteration using information gathered from the previous iteration. The way this update is performed is what distinguishes one algorithm from the next. One of the most popular of these algorithms is the BFGS method, which is an acronym of its authors; Broyden, Fletcher, Goldfarb and Shanno [100]. Instead of approximating the Hessian, it is the inverse of the Hessian which is approximated, as it is this which is ultimately needed to perform the update. The SLSQP algorithm is an extension of the BFGS algorithm to a Lagrangian function, meaning that it can be used in optimisation problems with constraints, which BFGS cannot [101].

2.4.3 MultiSLSQP

Given that gradient-based algorithms are faster and simpler to implement for solving inverse problems, these are preferred for this work. As outlined in section 2.2.1, the differentiability of squared error objective function depends entirely on the differentiability of the surrogate model. The differentiability of the surrogate model depends on the supervised learning algorithm used, therefore these must be chosen appropriately.

For this work, second order methods will be used as they remove the problem specific knowledge of the learning rate and are able to converge to a solution in substantially fewer iterations [102]. Given that constraints will occasionally be placed on the optimiser, the SLSQP algorithm is the best choice for this work.

An implementation of the SLSQP algorithm is provided in SciPy's Optimize package, along with a host of other algorithms [103]. In the majority of cases, the objective function will be non-convex, meaning there exists multiple local optima (non-uniqueness of inverse problems). To identify several inverse solutions, and to avoid being trapped in a bad local optima, multiple runs of the optimisation algorithm are required using differently initiated values $\mathbf{x_0}$. As is the case with many optimisation libraries, SciPy Optimize supports using only a single initial point. This means that trying l different initial points would have to be performed sequentially, which would be inefficient for large search spaces (since l would need to be large).

MultiSLSQP is a package developed by myself for this project which enables multiple initial points to be trialled simultaneously [46]. This is achieved through the use of tensor computation, something which the NumPy library (on which SciPy is built) is able to perform much more efficiently than following a sequential operation. The benefit of *MultiSLSQP* is that it enables the functional value and gradients for l different points to be evaluated together, before being updated by the SLSQP Fortran compiler used in SciPy. This package requires no additional libraries to those required by SciPy, and is compatible with version 1.4.0 to the most recent version (1.11.0).

The performance of *MultiSLSQP* is compared with the original SciPy implementation for a 10D problem with a forward model which requires a matrix inversion. The forward mapping is given by eq. 2.7, where A is a matrix of size

10×200, B is a matrix of size 200×200 and c is a vector of size 200×1. If $k = 1000$ different starting points were to be investigated then this previously would have required inverting a 200×200 matrix for each of the 1000 attempts, however with the multi-start implementation this only requires inverting this once. This leads to a reduction in the evaluation time from 17.08 s to 0.52 s, a reduction of more than 96%. While this is a simplistic example, it demonstrates the improvement that *MultiSLSQP* offers for rapidly identifying multiple optima in the parameter space.

$$f(\mathbf{x}) = \mathbf{x}AB^{-1}C \tag{2.7}$$

2.5 Synthetic Data Generation

In the majority of cases in science and engineering, the available, real data is insufficient for the given task. This may be because collecting sufficient quantities of the real data is too time-consuming, too costly, or is infeasible. Synthetic data, generated by means of computational models, is widely used in science and engineering to guide decision-making. In some industries, it is preferred to real data, i.e. medical sciences, where the use of synthetic patient data avoids potential ethical issues.

This section provides an overview of synthetic data generation in the field of computational engineering, highlighting the need for the development of a workflow package for streamlining this process.

2.5.1 Computational Modelling

Most physical systems can be described by a governing set of partial differential equations (PDEs), however for the vast majority of geometries and problems, PDEs cannot be solved with analytical approaches [104]–[106]. Computational modelling is a field which uses numerical methods to approximate the behaviour of these systems, and has been an extremely active field of research over the past seven decades.

The earliest and simplest numerical method was the finite difference method, which used finite differences to approximate derivatives in differential equations [30]. Although the idea of finite differences can be traced back as far as the early 18th century, it was the work of Courant, Friedrichs and Lewy in 1928 where a more general theory was established [107].

During the 1960s, the finite element (FE) method was developed by Zienkiewicz, Turner, Clough, Martin and Topp for structural mechanics applications [108]. In this method, the PDE is converted in to a weak formulation using a weighting function before it is returned to integral form over the domain. The domain is subdivided in to a collection of elements, with the equation of the system represented on a simplified, local, geometry. The equations representing each element are then recombined to create a global system of equations which are solved.

Dividing the domain into elements has a number of advantages; it provides an accurate representation of complex geometries, enables refinement in the size of element near regions with large gradients, and allows the use of multiple materials. Because of its generality, the FE method is most commonly used to perform structural, thermal and electromagnetic analysis.

The finite volume method was developed by McDonald, Mac-Cormack and Paullay shortly after the FE method and similarly subdivides the domain into smaller elements, or volumes as they are referred to here [109]. The main difference between this and the FE method is the adherence to conservation laws for each volume, meaning that what goes out of one cell must go into its neighbouring cell. Historically, this method has been very successful in solving CFD problems.

While computational modelling tools are instrumental in a variety of fields, one drawback is that they can be expensive to evaluate. This is because these methods require solving a system of equations, which is often time-consuming, especially for highly discretised, 3D problems.

Two important steps in the creation of any type of computational model is its verification and validations. Verification is the process of confirming that the computational model is correctly implemented with respect to the conceptual model. This is often characterised by the finding and fixing of modelling errors to ensure that the model works correctly and matches any agreed-upon specifications and assumptions [110]. Validation is the process of confirming that the computational model represents the real system to a sufficient level of accuracy. The validity of a model is usually determined by following a series of tests and comparing the results with the real system [111].

2.5.2 Obstacles

Generating synthetic data using computational models requires executing steps of a workflow. These steps will vary depending on the problem, the individual, and the organisation. In computational engineering, a typical workflow will first require creating an idealised representation of the component used for the analysis using an appropriate computer aided design (CAD) software. Following this, the CAD geometry is discretised in to a mesh using a meshing software, which is then used as an input to the computational model along with information about the analysis, e.g. material properties, BCs and initial conditions.

Unfortunately, in the majority of cases, executing this workflow is a time-consuming process, which is largely attributed to a lack of workflow automation. Workflow orchestration tools are extremely popular in a number of industries for creating more automated workflows, however as discussed in the work of Maric *et al.* they are not commonly employed in the field of computational engineering [112]. One reason for this is the manual steps performed at the different stages of the workflow, which is largely attributed to a dependence on the graphical user interfaces (GUI) of software packages. While GUIs are a central component to modern computers and provide ease of use and accessibility, dependency on them will likely limit the adoption of more automated processes that could require scripting or programming. During a typical computational engineering workflow, GUIs are used to visually construct the CAD geometry during its construction, observe and refine the fidelity of the mesh, and then used to manually assign BCs to parts of the mesh in the simulation, for example.

Many simulation packages will have design of experiment (DoE) capabilities, which enables a variety of different types of analyses to be specified in advance, thus providing a degree of automation in the collection of synthetic data. However, as high-performance computing (HPC) systems typically do not feature GUIs, or indeed any form of interactive usage, the simulations which make up the DoE are usually performed on a local computer. Given the more modest computing resources available on a local computer and the lack of parallelisation capabilities, these simulations would likely be performed sequentially, thus taking an infeasibly long time. For example, performing 400, one hour long simulations sequentially would take over two weeks to complete.

Another issue with the workflows developed in computational engineering is their lack of portability. Usually, a computational engineering workflow will require a variety of different software packages (e.g. CAD, meshing, modelling), which can make it extremely challenging to move the workflow from one system to another. This is because certain software packages are only compatible with certain versions of operating systems, and some will require a paid licence. This means that in many cases, the generation of synthetic data can only be achieved on a specific computer by an individual who knows the necessary steps. This inevitably leads to repetition of work between colleagues, resulting in a waste of both time and resources [112].

Certain commercial modelling packages aim to deliver a one-platform solution to overcome this issue [113]. The one-platform approach is where a single software offers a variety of tools, reducing the reliance a consumer has on an external or competing package. Ansys Workbench is an example of this, where meshing, various modelling tools, visualisation, and data analysis are all supported [114]. However, these platforms do not cater for modelling all kinds of phenomena, and their one-platform approach means it's inherently difficult to incorporate external tools – which are unavailable in the main package – in to the workflow. This can make it extremely challenging to automate a workflow using these types of packages. Along with this, these types of commercial packages usually have very large licence fees, meaning that it's inherently expensive to use them on multi-node HPC systems.

Generating surrogate models requires substantial quantities of synthetic data from across the parameter space of interest, e.g. different material properties, different magnitude of loads. Since HIVE continually tests different types of components, new synthetic data will need to be generated and used to train models in a short space of time. Therefore, there is need for a platform which supports the rapid generation of synthetic data in a fully automated fashion. Along with this, since this project is a collaboration between Swansea University and UKAEA, it is essential that the workflow developed at Swansea is easily portable to colleagues at UKAEA so that they can perform the same analysis.

2.5.3 Workflow Platform

This work will aim to develop a workflow platform for computational engineering which streamlines the production of synthetic data by harnessing the power of HPC systems. Such a platform will make data more abundant, supporting the use of ML algorithms and helping overcome many of the obstacles discussed by Pan *et al.* in moving engineering towards a more 'data centric' era [115]. A workflow platform for computational engineering will need to deliver the following;

1. The capability to automate the entire workflow.
2. Parallelisation to support rapid data generation.
3. Compatibility with HPC systems.
4. Support installation and configuration of multiple external software packages.
5. Enable data generated from various software packages to be seamlessly linked.
6. Ease of portability of the platform to different workflows and organisations.
7. Easily enable the inclusion of new analysis and new software packages.
8. Flexibility to adapt the workflow for different objectives.

Full automation enables the workflow to run 'in the background', while parallelisation, HPC compatibility and ease of portability means that it can be easily deployed on HPC clusters to perform the analysis in a fraction of the time. Portability also ensures that the platform is usable by a wide variety of individuals, who can easily include new types of analysis and software packages.

Such a platform would also greatly improve the ability to reproduce data, something which Maric *et al.* say is rarely adhered to, especially in academia [112]. This is because reproduction of the computing environment used to generate data is challenging, and not all details used throughout the workflow are available in publication. The creation of a workflow tool will improve the ability to reproduce data generated by others, since its fully-automated nature means that all the required details to perform the analysis are available. This not only supports the verification of others work, but also allows for modifications to be made to tackle different problems.

This has led to the development of the *VirtualLab* platform, predominantly by myself, for this thesis [43]. Details on how this package delivers the above objectives are discussed in chapter 5.

2.5.4 FAIR Principle

The FAIR principles, outlined by Wilkinson *et al.* in 2016, are a cornerstone of good scientific data management which states that data must be findable, accessible, interoperable, and reusable with minimal human intervention [116].

- **Findable**: It should be easy for both humans and computers to find the raw data and the metadata.
- **Accessible**: The data should be retrievable using an identifier (e.g. DOI) using a standardised and open communications protocol, with restrictions in place if necessary.
- **Interoperable**: It should be possible to integrate the data with other data, applications and workflows. The format of the data should therefore be open and interpretable for various tools.
- **Reusable**: Ultimately, the FAIR principle is in place to optimise the reuse of data. In order to do this, data should be well-documented, have a clear licence to govern the terms of its reuse.

All the synthetic data generated to perform the analysis for this thesis is hosted on Zenodo, which uses CERN's data centre to house scientific data [45].

2.6 Summary

This chapter has presented the state of the art in both inverse modelling and sensor placement optimisation, providing justification for the approach this work follows in addressing these issues. Most of the work in this chapter presents material from literature, and is designed to set the scene for the novel contributions provided in chapters 6 - 8. A novel contribution was provided by means of the *MultiSLSQP* package discussed in section 2.4.3, which showed a 96% reduction in the time taken to identify optima compared with the original SciPy implementation.

Section 2.5 highlighted that a lack of workflow automation means that engineering is slow in its transition towards a more data centric approach. This has resulted in the development of a fully automated workflow package, which will be presented in chapter 5.

Chapter 3

Surrogate Model Generation

3.1 Introductory Remarks

The focus of this chapter is on the methodologies required to develop both SV and FF surrogate models, and can be thought of as an extension of the current methodologies presented in chapter 2. Firstly, the methodology for the MLP and GPR algorithms used in this work are presented in section 3.2. Following this, section 3.3 presents the theory behind the PCA and auto-encoder algorithms for the purpose of dimensionality reduction, which are key components of successful FF surrogate models. Finally, since synthetic data is generated specifically for training these surrogate models, section 3.4 provides an overview of different data sampling techniques available in literature, along with their strengths and weaknesses for this application.

3.2 Supervised Learning Algorithms

Supervised learning is one of the three main branches of ML algorithms, along with unsupervised learning and reinforcement learning. The goal of supervised learning is to use labelled data to learn an optimal mapping between the inputs and outputs of the model. Labelled data is the term given to a data point which has an associated, or paired, observation, denoted as $\{\mathbf{x_t}, \mathbf{y_t}\}$. These observations are the 'ground truth' values, the difference from which the model is aiming to minimise during training. This difference is quantified by the loss function used. Scenarios where the observations are continuous values are categorised as regression models, while those that have discrete labels are termed classification models. Unsupervised learning uses unlabelled data, meaning that the goal instead is to identify the underlying structure of the data. Reinforcement learning is significantly different to the others, where the goal is to train an 'agent' in an interactive environment, where correct decisions are rewarded, and incorrect ones are punished. Surrogate models require regression supervised learning algorithms, and will be the focus of this section.

A supervised model, $\mathbf{F}_\theta$, maps the i input features to the j output labels. The subscript $\theta = \{\theta_1, .., \theta_k\}$ are parameters which parameterise the model. For conciseness, this mapping will be denoted as $\mathbf{F}$ from here on. During a process known as training, the values of these parameters are optimised using the available data. This data is referred to as the training dataset, and is made up of m observations. Another p observations are held back during training and are used to test the generality of the model. As a result, this dataset is commonly referred to as the test dataset. A model which performs poorly on both datasets is said to be under fitting, while one that performs well on only the training dataset is over fitting. A model which performs equally well on both is optimal, see fig. 3.1. In the majority of cases $p < m$ and the ratio of p to m is referred to as the test-train split.

The inputs of the training dataset are combined to create a matrix $X_{train} = [\mathbf{x_1}, ..., \mathbf{x_m}]^T$ of size $m \times i$, while a matrix of size $p \times i$ of test input data is given by $X_{test} = [\mathbf{x_{m+1}}, ..., \mathbf{x_{m+p}}]^T$. To ensure unbiased assessment of the model using the test dataset, it is essential that $X_{train} \cap X_{test} = \emptyset$. Similarly, training and testing matrices are constructed for the outputs, which are denoted by Y_{train} and Y_{test} and have size $m \times j$ and $p \times j$ respectively.

There are a variety of different metrics available by which to assess the performance of a supervised regression ML model. These are usually calculated individually for each of the j outputs, allowing the performance for each to be interpreted. The average value of these is then used to assess the model's general performance. These are calculated for the train and test data to assess the scores on both.

The mean absolute error (MAE) calculates the averaged absolute difference between the model prediction and ground truth over the observations of the dataset. This is often normalised by the range of the ground truth observations to give the normalised mean absolute error (nMAE), which is given in eq. 3.1 for the l-th output. Here $n = m$ for the training dataset and $n = p$ for the test dataset, $\mathbf{y_l}$ is column l of the target matrix (Y_{train} or Y_{test}), y_l^k is the k-th entry

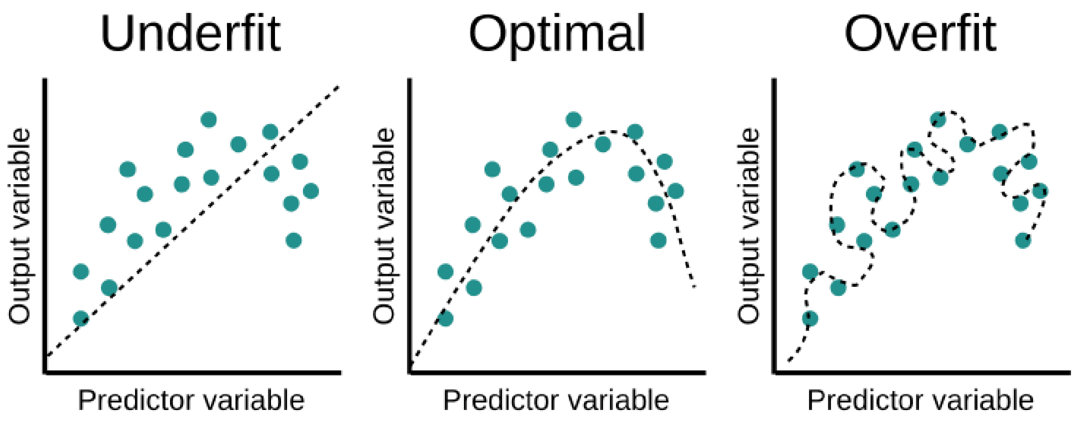


Figure 3.1: Comparison of three models, where one is under fitting, one is optimal, and one is over fitting [117].

of that vector and $F_l^k = F_l(\mathbf{x_k})$ is the l-th output using the ML model evaluated for input $\mathbf{x_k}$, which is the k-th column of the input matrix (X_{train} or X_{test}).

Similarly, the normalised root mean squared error (nRMSE) normalises the square root of the mean squared error (MSE) by the range of the data, see eq. 3.2. Eq. 3.3 is known as the R^2 score, where $\overline{F_l}$ denotes the averaged value for all mode outputs $\{F_l^1, ..., F_l^n\}$. This formula provides information about the fit accuracy, so that values of 1 is a perfect fit, while lower (and possibly negative) values suggests a poorer fit.

$$nMAE_l = \frac{\frac{1}{n}\sum_{k=1}^{n}\left|y_l^k - F_l^k\right|}{max(\mathbf{y_l}) - min(\mathbf{y_l})} \tag{3.1}$$

$$nRMSE_l = \frac{\sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(y_l^k - F_l^k\right)^2}}{max(\mathbf{y_l}) - min(\mathbf{y_l})} \tag{3.2}$$

$$R_l^2 = 1 - \frac{\sum_{k=1}^{n}\left(y_l^k - F_l^k\right)^2}{\sum_{k=1}^{n}\left(y_l^k - \overline{F_l}\right)^2} \tag{3.3}$$

All ML algorithms used in this work has been carried out using the PyTorch library, unless otherwise stated [118]. Since its conception in 2016 PyTorch has become an extremely popular ML library, especially in academic circles where it accounts for nearly 80% of the work carried out in published papers in 2021 [119].

3.2.1 Multi Layer Perceptron

ANNs are an extremely powerful and popular tool for tackling a wide variety of supervised ML problems. Inspired by the behaviour of biological neural networks found in human and animal brains, each artificial neuron (referred to as node) is connected to each other with a connection type and strength dependent on the type of ANN used.

The simplest and most common type of ANN is the MLP, an example of which is shown in fig. 3.2. An MLP consists of 1 or more hidden layers between the input and output layer and is an extension of a simple perceptron model which linked the input layer to the output layer directly. The nodes in one layer are connected to the nodes in the following layer via weights which are unknown initially but are learnt during training.

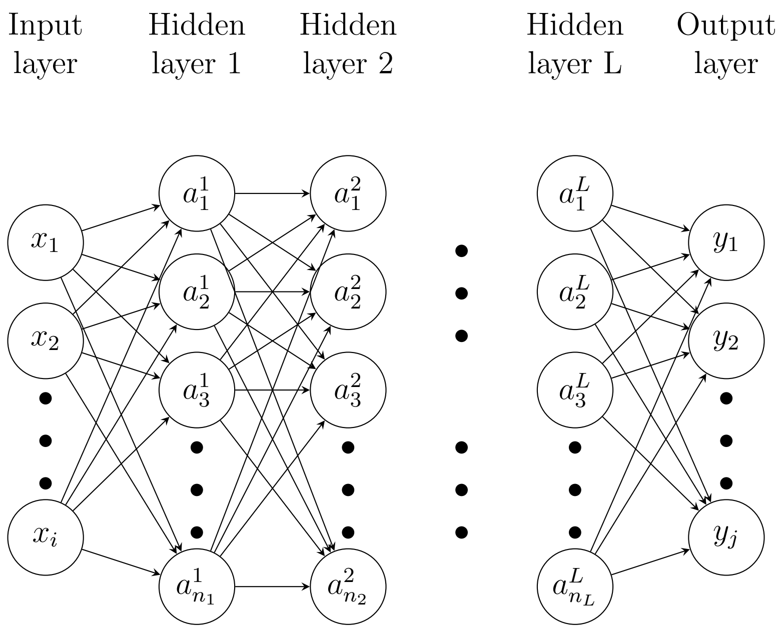


Figure 3.2: Graphical representation of a multi layer perceptron network.

MLPs are commonly referred to as deep neural networks because of the depth of the architecture. Each layer in the MLP can extract different features from the data, with deeper networks able to understand more complex and high-level features. The number of nodes in the input and output layer are the number of inputs and outputs to the model, which in this case is i and j respectively.

The general structure of the MLP is shown in eq. 3.4 - 3.6. W^p is a matrix of weights with size $n_p \times n_{p-1}$ with the entry at row k, column l defining the weight that connects node l in layer $p-1$ to node k in layer p, denoted by W^p_{kl} in eq. 3.5. An activation function g is applied to the sum of the information received by each node, with a^p_l denoting the value assigned to node l in layer p. The values which make up the entries to the matrices $W^1, ..., W^L$ are the parameters, θ often referred to as weights, which are optimised during training.

$$a^1_k = g\left(\sum_{l=1}^{i} x_l W^1_{kl}\right), k = 1, ..., n_1 \tag{3.4}$$

$$a^p_k = g\left(\sum_{l=1}^{n_{p-1}} a^{p-1}_l W^p_{kl}\right), k = 1, ..., n_p, p = 2, ..., L \tag{3.5}$$

$$y_k = g\left(\sum_{l=1}^{n_L} a^L_l W^{L+1}_{kl}\right), k = 1, ..., j \tag{3.6}$$

Equation 3.7 - 3.10 define 4 well known activation function; sigmoid, tanh, leaky ReLU, and swish respectively, with their profiles shown in fig. 3.3. The activation function is key to the MLP, as it is this which adds non-linearity to the model. While the formulation given by eq. 3.4 - 3.6 have the same activation function for each layer, this is not a requirement. It is common that regression problems will have an activation function of unity, $g(z) = z$, for the final layer (eq. 3.6).

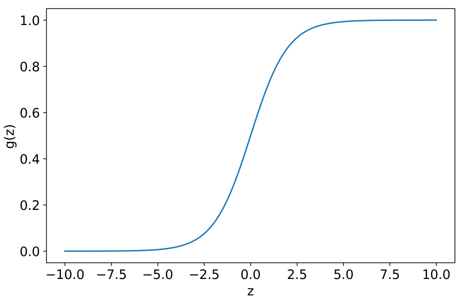
$$g_{sigmoid}(z) = \frac{1}{(1+e^{-z})} \tag{3.7}$$

$$g_{tanh}(z) = \frac{(e^z - e^{-z})}{(e^z + e^{-z})} \tag{3.8}$$

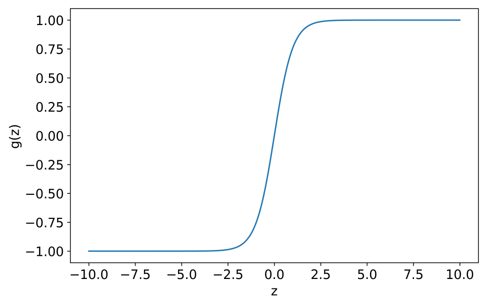
$$g_{ReLU}(z) = max(z, \alpha) \tag{3.9}$$

$$g_{swish}(z) = \frac{z}{(1+e^{-z})} \tag{3.10}$$

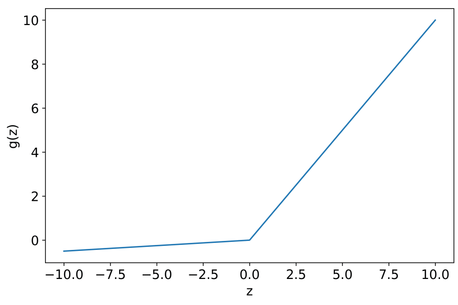
The first step in creating a MLP model is deciding the architecture of the model, which is dictated by the number of hidden layers the model has and the number of nodes in each layer. The architecture of a model interacts with other hyperparameters, so changing one of these can affect its performance, meaning there is no generic method for choosing a 'best' architecture. For example, a certain architecture may perform well using the ReLu activation function, however the sigmoid activation function would likely perform better on a different



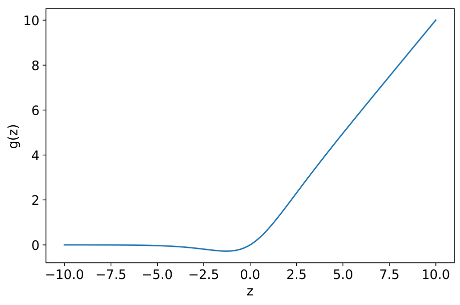
(a) Sigmoid



(b) Tanh



(c) Leaky ReLU



(d) Swish

Figure 3.3: Activation functions for ANNs.

architecture.

A common approach is to rely on experience and start with an architecture successfully used previously on similar problems. Variations to this initial architecture can be explored by adding or removing nodes and layers and changing activation functions until an appropriate model is found. There are a huge number of different permutations of models possible when changing these three parameters, which depending on the size of the model can be extremely slow to train. Along with this, the motivation for using surrogate models is to remove the trial & error aspect of choosing the DoPE, however with MLPs this uncertainty is being shifted to the model instead.

The loss function most commonly used for MLPs is MSE summed over the training dataset. The most popular method of updating the model parameters is using the back propagation algorithm [120]. In this algorithm the gradients for the weights in the output layer (layer $L + 1$) are calculated first, which are used to inform the gradients of the weights in layer L, and so on. Once the gradients for all the weights are known, they are updated. Due to the large number of trainable parameters, ML algorithms tend to use first order methods to make the update.

Calculating these gradients using the entire training data is referred to as a batch gradient descent. For large datasets, this can be time-consuming and give poor results due to ‘sharp’ minima [121]. Mini batch methods are a popular alternative, where the gradients are calculated on sub-sets of the training data before updating the weight and moving on to the next mini batch of data. Calculating the gradient on a fraction of the data ensures a degree of noise, which improves the model performance and generalisation properties by avoiding these sharp minima. Common choices for the mini batch size ranges from 16-512 data points, with a size of 1 referred to as stochastic gradient descent.

As MLPs usually have a large number of trainable parameters over-fitting is often an issue, however there have been a number of techniques developed which aims to avoid this. A popular technique employed is dropout, where the values of some nodes are randomly set to zero during training. The probability that a node being changed to zero is decided prior to training, and it can be applied to any number of layers. The idea is that this stops the model depending too heavily on the values of certain nodes within the network, improving its ability to generalise to new data. Another technique commonly employed is to calculate the loss value on the test data set alongside the training data set. The values of both losses are monitored, with training terminated once the loss on the test dataset begins to increase. Another consequence of models with many parameters is the existence of numerous local minima for the loss function, with the optima identified dependent on the randomly initiated weights.

It can be shown that the gradients of an MLP model’s outputs with respect to its inputs depends on the differentiability of the activation function [122]. The ReLU function, for example, is not differentiable at the origin, therefore a model which uses this activation function does not have a well-defined gradient everywhere.

3.2.2 Gaussian Process Regression

All regression models make assumptions regarding the behaviour of the unknown function. In linear regression problems, this is dictated by the degree of polynomial afforded to each input feature, while in MLPs this is decided by the architecture of the network. GPR models follows the assumption that the function can be modelled as a Gaussian process (GP).

By definition, a GP is a collection of random variables whereby any linear combination of those random variables has a uni-variate Gaussian distribution. The GP of f is given by eq. 3.11, where $m(\mathbf{x})$ is the mean function and $k(\mathbf{x}, \mathbf{x}^{'})$ is the covariance function. The choice of mean and covariance function are key as it encodes any assumptions made about the data.

$$f(\mathbf{x}) = \mathcal{N}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{3.11}$$

In most cases, values observed from a system are susceptible to variability due to a number of factors. This variability is factored in to the model by means of noise, ε, added to the function value $f(\mathbf{x})$, see eq. 3.12. The noise is assumed to be an independent and identically distributed Gaussian with zero mean and variance σ_n^2. Note that GPR models are a single output procedure only, therefore the theory presented here is for a function f which maps the i inputs to a single output. Extension to multiple outputs are discussed at the end of this section.

$$y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{3.12}$$

By following a Bayesian approach, the prior assumption made about the data using the covariance function is updated with the new evidence, which in this case is the training data. A joint distribution of the training outputs, $\mathbf{y}_{train}$, and the unknown test outputs, $\mathbf{f}_{test} = f(X_{test})$ is given by eq. 3.13. The mean vector $\mathbf{m}_{tr}$ is calculated using the mean function, m, applied to rows of X_{train} , while $K_{tr,te}$ is a $m \times p$ matrix whose entry at row t, column g is calculated using the covariance function, k, applied to row t and row g of X_{train} and X_{test} respectively. The other terms are calculated following a similar approach. Generally the mean function is assumed to be zero for most applications since the target outputs can be normalised to have a zero mean, therefore $m(\mathbf{x}) = 0$ from this point forward.

$$\begin{bmatrix} \mathbf{y}_{train} \\ \mathbf{f}_{test} \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} \mathbf{m}_{tr} \\ \mathbf{m}_{te} \end{bmatrix}, \begin{bmatrix} K_{tr,tr} + \sigma_n^2 I & K_{tr,te} \\ K_{te,tr} & K_{te,te} \end{bmatrix} \right) \tag{3.13}$$

The posterior distribution is calculated by conditioning the joint distribution on the observations. Conditioning is the process of fixing the values of certain variables in a multi-variate distribution and observing the distribution for the remaining, free variables. In this case, the m variables associated with the observations are fixed, resulting in the posterior distribution, which is summarised in eq. 3.14. The simplification to the posterior mean in eq. 3.15 is due to the use of the zero mean function, while the posterior variance is given in eq. 3.16. One of the biggest drawbacks of GPR is highlighted here, with the need to invert a $m \times m$ matrix. This operation scales with $O(n^3)$, so this is usually calculated using the Cholesky decomposition. For the noise-free case, the term σ_n^2 can be omitted in these equations.

$$p(\mathbf{f}_{test}|X_{test}, X_{train}, \mathbf{y}_{train}) \sim \mathcal{N}(\mu_*, \Sigma_*) \tag{3.14}$$

$$\begin{aligned}\mu_* &= \mathbf{m}_{te} + K_{te,tr} \left(K_{tr,tr} + \sigma_n^2 I\right)^{-1} (\mathbf{y}_{train} - \mathbf{m}_{te}) \\ &= K_{te,tr} \left(K_{tr,tr} + \sigma_n^2 I\right)^{-1} \mathbf{y}_{train}\end{aligned} \tag{3.15}$$

$$\Sigma_* = K_{te,te} - K_{te,tr} \left(K_{tr,tr} + \sigma_n^2 I\right)^{-1} K_{tr,te} \tag{3.16}$$

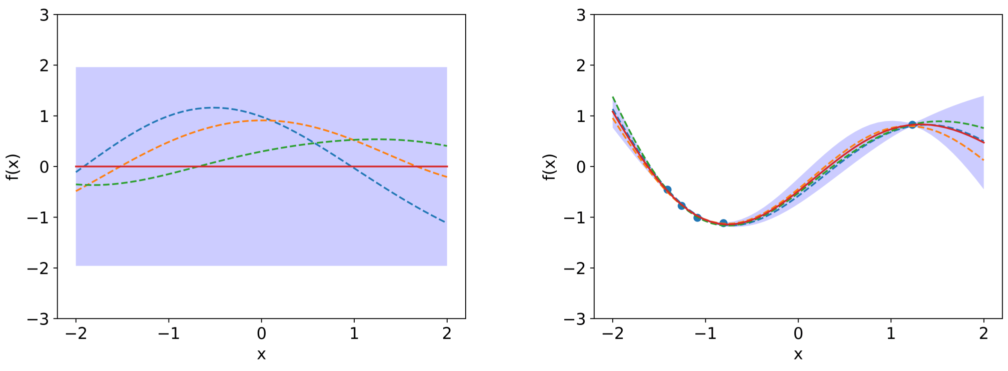
From the posterior distribution, predictions of the values at the points in X_{test} are made. As this is a Gaussian distribution its mean is the value with the highest probability, so it is this which is used to infer the values at these points. An additional bonus to GPR is that the variance, which indicates the spread of the data in the distribution, is available and can be used to give a measure of the confidence in the prediction. Using the variance, a confidence interval (CI) can be constructed, which gives a range of values that the prediction is estimated to fall between for a specified level of confidence.

The mean, three randomly drawn samples and the 95% CI is shown in fig. 3.4 for the prior and posterior distribution of a simple 1D example with 5 observed points. These are calculated at 50 equally spaced query points across input range. The improvement using the posterior is clear as it has been conditioned to pass through the observed points. The CI is small in the vicinity of the observed points, but are larger further away from them.

Crucial to any GPR model is the covariance function chosen. There have been a number of different covariance functions defined in literature that poses varying number of free parameters, with some of the most popular presented here.

The simplest of these is a linear kernel, eq. 3.17, where σ_o^2 is an offset term and $\Lambda = diag\,(l_1^2, ..., l_i^2)$ is a diagonal matrix of lengthscales for each dimension. This kernel has $i + 1$ trainable parameters and makes the assumption that the regression problem is linear.

$$k(\mathbf{x}, \mathbf{x}') = \sigma_o^2 + \mathbf{x}\Lambda\mathbf{x}'^T \tag{3.17}$$



(a) Prior (b) Posterior

Figure 3.4: Mean and 95% confidence interval of prior and posterior distribution, along with 3 randomly drawn samples.

Covariance functions that prescribe different lengthscales for each dimension are known as automatic relevance determination (ARD) kernels [123]. These naturally give better performance compared with those that have a single lengthscale $l = l_1 = ... = l_i$ and will be employed by all covariance functions used in this work.

The most popular covariance function used for non-linear regression problems is the radial basis function (RBF) [124]. The formula for this is given in eq. 3.18, where σ_f^2 is a scaling factor and Λ is the same as that of the linear kernel. This assigns a higher value to points closer together and tends towards zero as the distance between them increases. The lengthscale in the denominator accentuates this, with a smaller value producing a more rapidly changing function. A general rule of thumb is that it is not possible to extrapolate more than l_k units way from your data for dimension k [124].

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \Lambda^{-1} (\mathbf{x} - \mathbf{x}')\right) \tag{3.18}$$

This impact that the lengthscale has on the outcome is highlighted in fig. 3.5, where the posterior mean and 95% CI are shown for 3 different lengthscales for a simple 1D problem. As anticipated, the CI increases rapidly away from the observed points for a small lengthscale while for larger values the function is unable to pass through all observations.

Even with the ability to change the lengthscales the RBF function is still very smooth, which is attributed to the fact that the GP has mean squared of all orders due to the infinitely differentiable nature of the function [124]. It

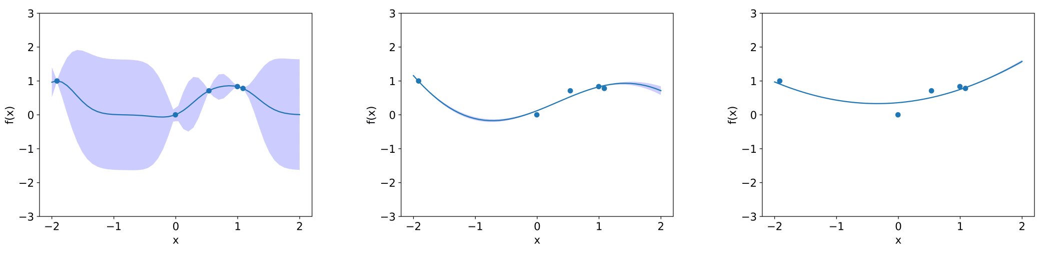

(a) $l = 0.1$ (b) $l = 1$ (c) $l = 3$

Figure 3.5: Mean and 95% confidence interval for the RBF covariance function with different lengthscale l.

is argued that a function with strong smoothness properties such as this are unrealistic for modelling many real world problems, with the Matérn covariance function recommended instead [125]. This covariance function is shown in eq. 3.19, where K_ν is a modified Bessel function, Γ is the gamma function, ν is a positive parameter which dictates the sharpness of the function and r is the distance between $\mathbf{x}$ and $\mathbf{x}'$ scaled by each lengthscale as shown in eq. 3.20.

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} (\sqrt{2\nu} r)^\nu K_\nu(\sqrt{2\nu} r) \tag{3.19}$$

$$r = \left((\mathbf{x} - \mathbf{x}')^T \Lambda^{-1} (\mathbf{x} - \mathbf{x}')\right)^{1/2} \tag{3.20}$$

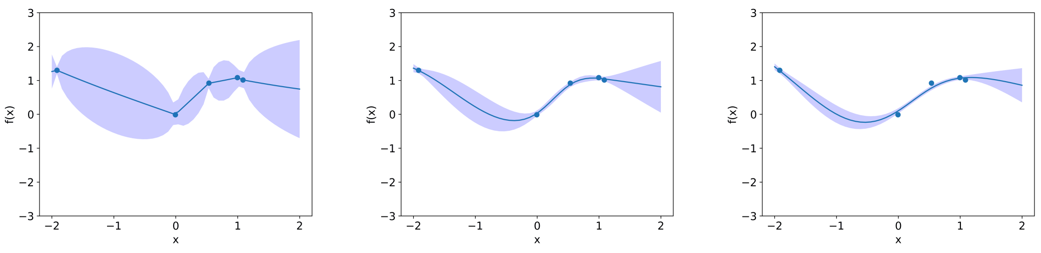
The behaviour of the Matérn function varies widely with the choice of ν. As $\nu \to \infty$ Matérn becomes equivalent to the RBF function, however, for $\nu = 1/2$ the Matérn function is identical to the absolute exponential covariance function which results in a jagged, piecewise linear function.

Calculating the Bessel and gamma function can be computationally demanding, however the formulation of the Matérn function is simplified in the case where $\nu = p + 1/2$ and p is a non-negative integer. Values of $\nu \geq 7/2$ are ignored as it is very difficult to distinguish between them and the RBF kernel, while $\nu = 1/2$ has a limited range of applications. This leaves $\nu = 3/2$ and $\nu = 5/2$ as the most commonly used variants of the Matérn function, which are given by eq. 3.21 and 3.22 respectively. These are denotes as Matérn$_{5/2}$ and Matérn$_{3/2}$.

$$k_{\nu=3/2}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \sqrt{3}r\right) \exp\left(-\sqrt{3}r\right) \tag{3.21}$$

$$k_{\nu=3/2}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \sqrt{5}r + \frac{5}{3}r^2\right) \exp\left(-\sqrt{5}r\right) \tag{3.22}$$

The predicted mean and CI using the Matérn covariance function are shown in fig. 3.6 for the three values of ν discussed above. The lengthscale is fixed at $l = 1$ which highlights how the Matérn function converges towards the RBF function for larger ν by comparison of fig. 3.5b and 3.6c. The piecewise nature of the function with $\nu = 1/2$ is also clearly shown.



(a) $\nu = 1/2$ (b) $\nu = 3/2$ (c) $\nu = 5/2$

Figure 3.6: Mean and 95% confidence interval using the Matérn covariance function for different values of ν .

Other popular covariance functions which aren't presented here include the periodic function, which assumes a degree of periodicity to the data, the γ-exponential functions and the rational quadratic function [124].

In a Bayesian framework, the inference of the posterior distribution is calculated by eq. 3.23, where the terms in the numerator are the likelihood and prior (respectively) and the denominator is the marginal likelihood. Note that this is the same posterior distribution as eq. 3.14 with the addition of the term θ which were previously omitted.

$$p(\mathbf{f}|X, \mathbf{y}, \theta) = \frac{p(\mathbf{y}|X, \mathbf{f}, \theta)p(\mathbf{f}|\theta)}{p(\mathbf{y}|X, \theta)} \tag{3.23}$$

Continuing with the Bayes methodology a probability distribution over all θ can be created, in what is called the second level inference, see eq. 3.24. It is similarly composed of a likelihood, prior and marginal likelihood for the parameter θ. To maximise the second level posterior, one must maximise its likelihood, $p(\mathbf{y}|X, \theta)$, which is equal to the marginal likelihood from the first level inference (eq. 3.23). This is commonly referred to as a type-II maximum likelihood estimator.

$$p(\theta|X, \mathbf{y}) = \frac{p(\mathbf{y}|X, \theta)p(\theta)}{p(\mathbf{y}|X)} \tag{3.24}$$

For mathematical convenience, the log of the marginal likelihood is used and is referred to as the marginal log likelihood (MLL). The MLL is shown in eq. 3.25 where $K_y = K + \sigma_n^2 I$ and K is the covariance matrix. As most ML packages focus on minimising a loss function, it is the negative of the MLL which is used for training a GPR model.

$$MLL = \log(p(\mathbf{y}|X, \theta)) = -\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi \tag{3.25}$$

The differentiability of a GPR model depends entirely on the differentiability of the covariance function used [124]. If the covariance function is first order differentiable, then the model will also be first order differentiable, and so on. The RBF covariance function is infinitely differentialy, while Matérn$_{3/2}$ is first order differentiable and Matérn$_{5/2}$ is second order differentiable.

To model cases where multiple outputs are required, there are two options available. The first of these is to train a separate, independent GPR model for each output and will be referred to as a multi-output GPR (MO-GPR) model [126]. This approach is flexible as it enables each output to be modelled using a different covariance function, however this does require training j separate models, although it is important to note that this process can be performed in parallel due to their independent nature.

The second option is a multitask GPR (MT-GPR) model, where similarities between the outputs are learnt during training [126]. The similarities between any two outputs are stored in a inter-task covariance lookup table, which is used in conjunction with the standard covariance matrix to make predictions. While this implementation requires training only a single model, it does assume that the outputs are correlated to each other, which limits its range of applications.

Note that PyTorch does not include GPR models, therefore the additional package GPyTorch is required and is what's been used to perform the analysis in this section [127].

3.3 Dimensionality Reduction

In the case of FF surrogate models, the number of outputs are extremely large, often having to predict the values at tens or hundreds of thousands of nodal positions. Modelling each of these individually would be infeasible for most ML algorithms, therefore these are usually compressed using dimensionality reduction techniques. These are an extremely popular and widely used tool in ML, moving data from a high-dimensional space to a lower-dimensional space while preserving certain characteristics of the data. These are a type of unsupervised learning algorithms, as no labels are assigned to the data points.

In the vast majority of applications, dimensionality reduction is used to reduce the number of input features a model has. The most basic reason to do this is to reduce the data to 2 or 3 dimensions so that is can be visualised and interpreted more easily. Another reason it is employed is to overcome the curse of dimensionality [128]. As the number of dimensions grow, the volume of the hyperspace becomes larger, meaning that the current data becomes more and more sparse. This curse states that as the number of dimensions grow, the quantity of data required to maintain a consistent result will need to increase at an exponential rate. Another benefit of reducing the dimensionality is the removal of any potential multi-collinearity between two variables.

When reducing the number of input features, it is highly unlikely that the original, high dimensional data would ever need to be recovered. As a result, certain techniques, such as manifold learning, are unsuitable for the task of data compression as they are unable to project the data back to its original, higher dimensional space. This section present two dimensionality reduction techniques which enable the compressed data to be projected back to its higher dimensional space; PCA and auto-encoders.

3.3.1 Principal Component Analysis

The PCA algorithm reduces the dimensionality of data by projecting in on to lower dimensional subspaces. The PCs which data is projected on to are optimal in terms of retaining the variance in the data. The first PC is that which has the highest variance when it is projected on to the 1D subspace. Similarly, the second PC, which is orthogonal to the first, is chosen as that which maximise the variance on 2D projected data. The PCs which maximise the variance of the projected data are equal to those which minimise the distance between the data and its projection on to the subspace.

Prior to performing PCA, the data must first be normalised. Normalising is a process which involves shifting data so that each dimension has a mean of zero and scaling it to so that each has a standard deviation of 1. Shifting the data to centre around the origin is intuitive as the PCs, much like the original axes, travel through the origin.

To understand the PCA algorithm, first the singular value decomposition (SVD) of a matrix is presented. SVD is a technique which factorises any rectangular $m \times n$ matrix, A, into three matrices, U, S and V^T, see eq. 3.26. S is a diagonal, $p \times p$ matrix whose entries are the singular values of matrix A where p is the rank of A, and is usually $\min(m, n)$. These singular values are denoted by s_i and sorted in descending order such that $s_1 > s_2 > ... > s_p$. U is a $m \times p$ matrix whose columns are the left singular vectors of A, while V^T is a $p \times n$ matrix whose rows are the right single vectors of A. It turns out that the left and right singular vectors are the eigenvectors of AA^T and A^TA respectively, while the singular values are the square root of the eigenvalues, $\lambda_i = \sqrt{s_i}$.

$$A = USV^T \tag{3.26}$$

Take A to be the matrix of data where each of the m rows is a n-dimensional sample point, with $n >> m$ in most cases. In the case where the data is centred, meaning that the mean of each column of A is zero, A^TA is the covariance matrix for the dataset. The diagonal elements of a covariance matrix are the variances for each variable, the sum of which gives the total variance in the data. The trace theorem states that the trace of a square matrix is equal to the sum of the eigenvalues of the matrix, which highlights the connection between variance and eigenvalues [129].

The entries of S are sorted in descending order, therefore the largest value, and thus that which contributes most to the overall variance, is the first entry. This is associated with the first row of V^T, which in turn is the first PC. Likewise, the second PC is the second row of V^T and contributes the second most to the variance, and so on.

Deciding how many PCs to project the data on to is dependent on the application. If it is so that the data can be interpreted, then the data will be projected to 1, 2 or 3 dimensions so that it can be visualised. More often than not, the data will be projected to a number of dimensions, which ensures that the overall variance retained is above a certain threshold. The fraction of data retained by a projection on the first k PCs is given by eq. 3.27. The fraction of variance to retain advised in literature vary, however all advise retaining at least 0.99 of the variance of the original data. This is one of the biggest advantages of using PCA; as all PCs are identified from the beginning, it allows the number to be increased if accuracy is insufficient.

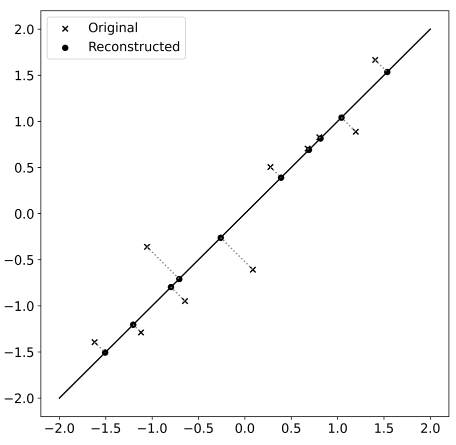
$$\theta_{PC} = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{i=1}^{p} \lambda_i} \tag{3.27}$$

Projection of the data in to the lower dimensional space is achieved by multiplying the high dimensional data by the transpose of V^T, which is of course just V. If k PCs are used, then the reduced form of V, given by V_k, is used to project A to the lower dimensional space $A_k = AV_k$. Reconstructing this data, which involves projecting it back in to its original higher dimensional space, is achieved by $\tilde{A} = A_k V_k^T$. If $k < p$ then $\tilde{A} \neq A$ due to this being a lossy compression algorithm.

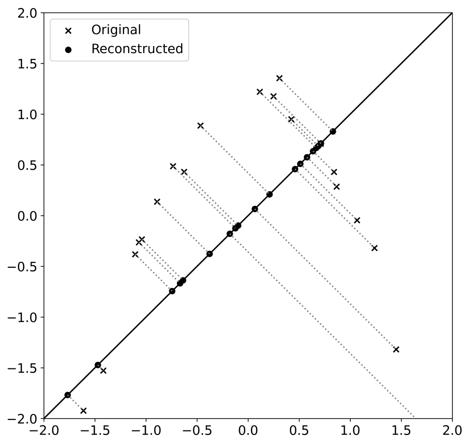
An example of the PCA algorithm being used to project 2D data to a 1D subspace is shown in fig. 3.7a. The dotted line shows the distance between the original point, A, and reconstructed ones, $\tilde{A}$. $\theta_{PC} = 0.966$ of the overall variance of the data is retained using the 1D encoding of the data. Comparing this results with the PCA analysis shown in fig. 3.7b highlights the weakness of the algorithm. As data is projected on to linear subspaces, it performs poorly when the form of data is non-linear, as it is in this case. As anticipated from the high errors between the original and reconstructed data, the amount of variance retained using this projection is $\theta_{PC} = 0.538$.

3.3.2 Auto-encoders

Auto-encoders are a special type of ANN algorithm which are used to efficiently learn a lower dimensional encoding of a dataset. They consist of three parts; an encoder, a latent space and a decoder, see fig. 3.8. The role of the encoder is to compress the original, high-dimensional data to an encoded representation, which is often orders of magnitude smaller than the original. The latent space, often referred to as the bottleneck, is the encoded representation of the data, which is the last layer of the encoder and the first layer of the decoder. The decoder then reconstructs the data to its high-dimensional form from the encoded



(a) Linear data



(b) Non-linear data

Figure 3.7: Projection of 2D data on to 1D subspace using PCA.

representation, which is compared with the ground truth (the input) to assess its accuracy.

Auto-encoders are used for a variety of tasks, meaning that there are a variety of different types found in the literature. The simplest of these is a MLP auto-encoder, where the encoder and decoder are fully-connected, feed-forward MLP models [61]. Convolutional auto-encoders, which use CNNs to represent the encoder and decoder, are popular for tasks which involve images [62] and time series data [63]. As this work requires the compression of simulation field data, an MLP auto-encoder will be sufficient for this task.

As with any MLP model, there are key hyperparameters relating to the architecture which must tuned. The size of the latent space is the most important of these, as the ratio of this against the input size dictates the level of compression achieved by the model. As with any MLP, the number of layers and nodes per layer will vary, however the bottleneck will almost always be the layer with the fewest nodes. In general, the number of nodes in each subsequent layer decrease in the encoder and increase in the decoder. This ensures there aren't rapid changes in the sizes of the layers, avoiding issues which this can present.

Since MLPs are able to learn non-linearity between the inputs and outputs, it means that they are much better suited for data compression tasks than PCA when the data is non-linear. Using a 2-10-1-10-2 architecture, a much better reconstruction of the data is achieved on the non-linear dataset, as shown in

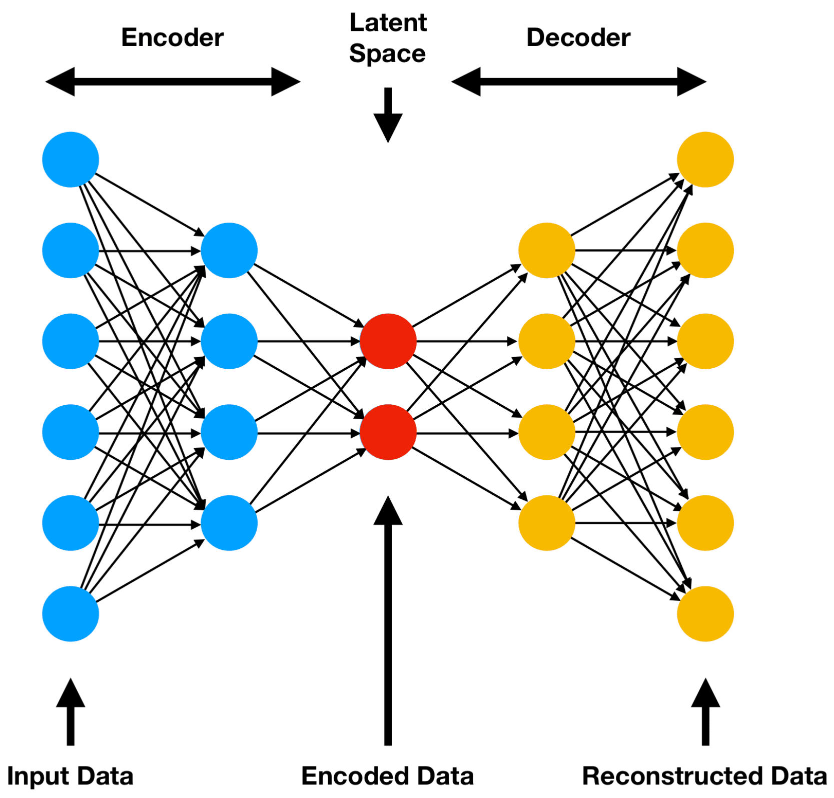
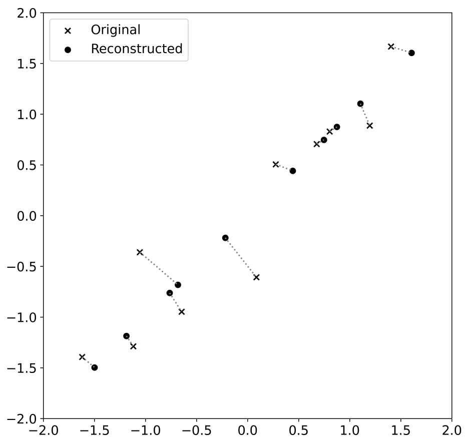


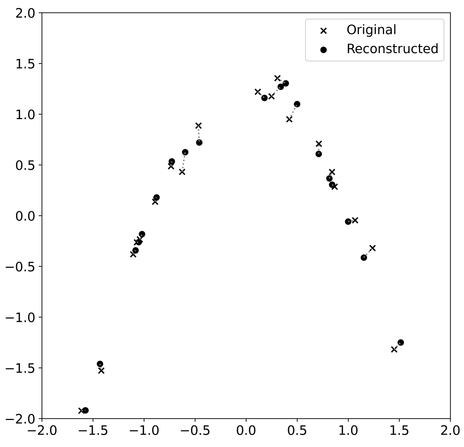
Figure 3.8: Example auto-encoder architecture, where 6D data is compressed to 2D.

fig. 3.9b. For completeness the results of the same auto-encoder on the linear data set is also included, fig. 3.9a, where a similar linear reconstruction as the PCA is identified.

Although they are able to outperform PCA in certain circumstances, they do poses certain drawbacks. Their main disadvantage is that due to the fact that the input and output layers are usually large, the models consist of a high number of trainable parameters, which can be time-consuming to train and can be prone to over fitting. Along with this, changing the level of compression (the size of the bottleneck) to improve the performance will require training an entirely new auto-encoder, which would add considerable time to the workflow.



(a) Linear data



(b) Non-linear data

Figure 3.9: Comparison of original and reconstructed 2D data after compression to 1D using a 2-10-1-10-2 auto-encoder.

3.4 Data Collection Strategies

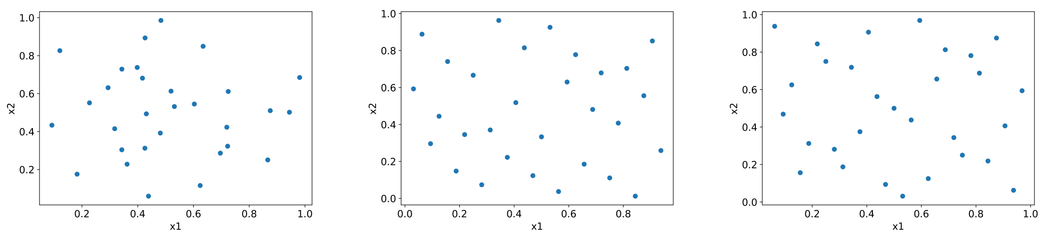
In order to train a surrogate model, synthetic data from across the parameter space must be gathered. There are a number of different sampling methods presented in literature to effectively choose the most informative points. These sampling methods can be divided into two categories; pre-determined and adaptive sampling.

3.4.1 Pre-determined

Pre-determined sampling, often referred to as one-shot sampling, is where all points in the parameter space which require evaluation are known from the outset. This makes them extremely easy to use and straightforward to apply in terms of their coding.

While there are a variety of different schemes quoted in literature, one requirement for this work is that adding new data to the existing data is easy and does not imbalance its distribution. For example, Latin hypercube sampling (LHS) is an extremely popular method used to spread data across the parameter space as evenly as possible. Using the LHS method, however, if an initial set of l points, denoted by S_l, is deemed to give insufficient accuracy and a larger set of m points is required, it is the case that $S_l \not\subset S_m$. This means that adding new points to the initial dataset becomes more complicated, which means that the LHS method is not applicable for this work.

Random sampling, which randomly chooses points from across the parameter space using a uniform distribution, is the simplest of the available schemes. By using a uniform distribution, each point has the same probability of being chosen, meaning that each point in the parameter space is treated equally. Due to the independent nature of drawing from a uniform distribution, it can mean that certain regions are densely sampled while others are not explored at all. This behaviour is highlighted in fig. 3.10a, where 30 points in a 2D parameter space are generated using the random sampling approach.



(a) Random (b) Halton (c) Sobol

Figure 3.10: 30 points in 2D parameter space generated using pre-determined sampling schemes.

The Halton sequence is a quasi-random, low discrepancy sequence which assigns a different co-prime as the base of each dimension [130]. The first 30 points of the Halton sequence in two dimensions is shown in fig. 3.10b. Here, the first dimension (x1) uses base 2, while the second dimension (x2) uses base 3. It has been noted that there is a high degree of correlation between certain higher dimensional bases, the first of which are 17 and 19. There are ways to avoid this issue, such as discounting the first n number of points, with n depending on the bases used, or by 'scrambling' the Halton sequence. More details on this problem and how it can be avoided can be found in [130].

Another popular low-discrepancy method is the Sobol sequence. The mathematics behind this sequence is slightly more complex than that of the Halton sequence, however fundamentally it uses generative matrices to represent numbers for each dimension in binary form (base 2). Similarly to Halton, for higher dimensions this will also face correlation problems, however it has been shown that both Sobol and Halton work well up to 10 dimensions [131]. It has been shown that Sobol and Halton both produce similar levels of accuracy in terms of data collection for surrogate models [132].

One drawback of pre-determined sampling is its evenly distributed approach, meaning that interesting regions of the parameter space are sampled as much as those which are of less interest, see fig. 3.11. These regions of interest could be a highly non-linear response, discontinuities or those which are key for a given application.

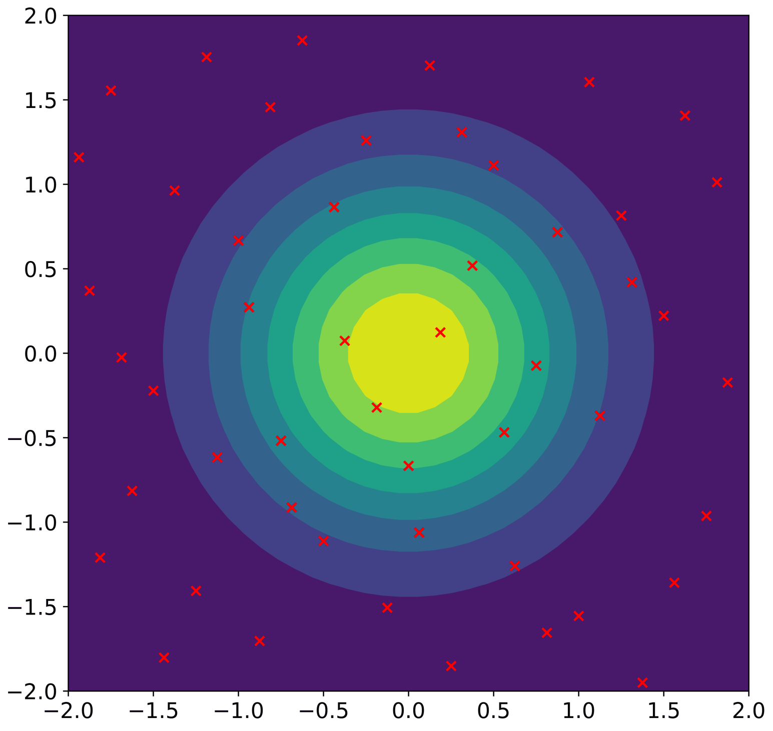


Figure 3.11: Inefficient sampling using pre-determined Halton sampling method.

The advantage of pre-determined sampling methods is it allows simulations to be performed concurrently, as all points are known a priori. If one has the capability of running multiple simulations simultaneously, such as in *VirtualLab*, the time required to collect the data can be reduced considerably.

3.4.2 Adaptive

Adaptive sampling differs from pre-determined sampling in that the next point is decided using information from the previously gathered points. The true forward model is evaluated using this point, which is then added to the pool of gathered points to inform the next. Most of the adaptive schemes presented in literature are single selection policies, meaning that only one new point is identified at each step. This process is summarised in eq. 3.28, where the next point in the parameter space $\mathbb{X}$ is that which maximises the refinement criteria, denoted by RC.

$$\mathbf{x}^{m+1} = \arg\max_{\mathbf{x}\in\mathbb{X}} RC(\mathbf{x}) \tag{3.28}$$

Each adaptive scheme has a different refinement criteria and is what guides the next point. Refinement criteria are usually composed of two parts; exploration and exploitation. The aim of exploration is to evenly scan the parameter space and build up a general understanding of the mapping. Exploitation, on the other hand, aims to use the knowledge gained from the available observations and place points in regions which are deemed to be interesting, e.g. those with large errors, high gradients or optima, for example. Schemes which are exploration only will spread the points out across the parameter space more evenly, but may miss some important features, while those which are exploitation only will heavily sample areas at the expense of poor performance elsewhere.

The most common type of exploration technique is distance-based exploration. Here, the distance between each of the candidate points and the previously sampled points are calculated, with that which is furthest away from its nearest neighbour contributing most to the RC. This is summarised in eq. 3.29, which is commonly referred to as the Monte Carlo intersite-proj-th (MIPT), where $\mathbf{x}^*$ is the nearest, previously sampled point to $\mathbf{x}$.

$$RC_{MIPT}(\mathbf{x}) = |\mathbf{x} - \mathbf{x}^*| \tag{3.29}$$

When a GPR is used for the surrogate model, another explorative technique is available, which is variance based exploration. As the variance gives a measure of the uncertainty of the model's prediction, the candidate point which maximises this has the biggest impact on the refinement criteria. The simplest of these schemes is the maximum mean squared error (MMSE), given by eq. 3.30 [133]. As the variance is based on distance information combined with the covariance kernel, there is a natural link between distance based and variance based approaches [134].

$$RC_{MMSE}(\mathbf{x}) = \sigma^2 \tag{3.30}$$

The exploitative component is the more nuanced aspect of adaptive schemes and the one which has the greater number of options. Geometry based exploitation schemes aim to target regions near certain geometric features which lead to high errors between the surrogate and original model. These geometric features could be sudden changes, such as spikes, in the profile which would easily be missed by exploration only strategies. There are generally two ways to try to target these areas; by observing the gradients (usually approximated) or looking at the error, or distance, between the surrogate and original model.

One of the most popular gradient based schemes is the local linear approximation (LOLA) on Voronoi cells proposed by Crombeq *et al.* [135]. The previously sampled points are used as vertices of Voronoi cells, which divide up the parameter space. For each Voronoi cell the magnitude of the gradient, E, and the volume, V, are calculated and summed together, see eq. 3.31. Here the volume of each cell works as the exploration aspect, ensuring no large areas of the parameter space are without a point, while the gradient gives the exploitation, which are larger around certain geometric features. Generally, the mid-point of the Voronoi cell with the largest score is used as the next sampling point.

$$RC_{LOLA}(\mathbf{x}) = E(\mathbf{x}) + V(\mathbf{x}) \tag{3.31}$$

While the logic of the LOLA scheme is straight forward, its application can be very difficult. Discretising high dimensional spaces into Voronoi cells is computationally expensive, meaning that this scheme becomes infeasible when the number of parameters is large.

Schemes which look at the error between the surrogate and ground truth model overcome this issue, as they do not require creating Voronoi cells. The expected improvement for global fit (EIGF) by Lam instead uses the squared error between the predicted value at the candidate point and the ground truth value at its nearest neighbour for the exploitation component [136]. The refinement criteria for the EIGF scheme is shown in eq. 3.32, where S is the surrogate model and $y(\mathbf{x}^*)$ is the value at the nearest, previously sampled point. The exploration is accounted for using the model variance σ^2 at point $\mathbf{x}$ however this can easily be changed to a distance based exploration if the surrogate is not a GPR model.

$$RC_{EIGF}(\mathbf{x}) = (S(\mathbf{x}) - y(\mathbf{x}^*))^2 + \sigma^2 \tag{3.32}$$

Another interesting approach to exploitation is query by committee (QBC). Instead of using a single model to guide the next point, a committee of models is used, where the location where the biggest difference between the predictions of the committee members is used. This difference is usually calculated as the variance between the predictions, shown in eq. 3.33, where $\{S_1^c, ..., S_{n_c}^c\}$ are the n_c surrogate models which make up the committee and $\bar{S}^c$ is their mean value.

$$\sigma_{QBC}(\mathbf{x}) = \frac{1}{n_c}\sum_{i=1}^{n_c}\left(S_i^c(\mathbf{x}) - \bar{S^c}(\mathbf{x})\right)^2 \tag{3.33}$$

The mixed adaptive sampling algorithm (MASA) was developed by Eason and Cremaschi and uses this QBC variance along with a distance-based exploration [137]. The refinement criteria for the scheme is shown in eq.3.34, where $D_{min} = |\mathbf{c} - \mathbf{x}^*|$ with $\mathbf{x}^*$ defined above. The exploration and exploitation components are each scaled by their respective maximum value measured over the set of candidate points, denoted by $\mathcal{C}$ to ensure both contribute equally to the score.

$$RC_{MASA}(\mathbf{x}) = \frac{D_{min}(\mathbf{x})}{\max_{\mathbf{x}\in\mathcal{C}} D_{min}} + \frac{\sigma_{QBC}(\mathbf{x})}{\max_{\mathbf{x}\in\mathcal{C}} \sigma_{QBC}} \tag{3.34}$$

Another popular choice for exploitation is cross validation, where multiple models are trained using different sub-sets of the training data. One of the most well-known of these is the leave one out cross validation (LOOCV), which measures the error using m surrogate models trained by omitting a single point from the training data, i.e. $X_{train}\backslash\mathbf{x}_i$ for $i = \{1, .., m\}$. The error between the full surrogate model and augmented one measures the sensitivity of the loss of data at a point, and thus aims to sample more in regions with larger errors. These types of exploitation based schemes are not presented here as they are infeasible for models with larger numbers of training data, as is used in this work.

Adaptive sampling methods have the advantage of being able to focus on key areas within the parameter space. This usually results in a reduction in the number of data points required and/or an improvement in the model accuracy compared with pre-determined sampling.

One drawback of adaptive sampling is that the collected data is optimal with regard to the refinement criteria chosen. Using the collected data for another purpose, such as creating a surrogate of an alternative results field calculated during the simulation, may lead to poor result. For example, if the adaptive refinement criteria was based on the temperatures generated by a thermo-mechanical model, then these data points collected are likely to be suboptimal for the creation of a surrogate model of the mechanical response of the component, e.g. displacement, stress, or strain. Along with this, the single selection nature of adaptive schemes means that high-throughput running of simulations is impossible due to the dependence of the next point on the previous. This means that while they tend to be more efficient in terms of the number of data points required, they would be more inefficient with respect to time compared with a pre-determined sampling strategy used with concurrently running simulations.

It is possible to extend the single selection policies discussed above to multi-selection policies. Multi-selection policies are where a batch of $l > 1$ future sampling points are advised at each iteration. The most common approach for this is to approximate the ground truth value at the first sampling point using the surrogate model, temporarily adding this point to the list of previously sampled points, and then repeating the procedure to identify the next best sampling point. Once the l best sampling points are identified, they are used to generate the ground truth values using the simulation and replace the approximated ones. While this enables the l simulations to be performed concurrently, thus speeding up the procedure, the need to approximate values detracts from the underlying purpose of adaptive schemes.

3.5 Chapter Summary

This chapter has presented the methodologies required for generating SV and FF surrogate models. These methods are the basis for the surrogate models used for the analysis presented in chapters 6, 7, and 8. This chapter also provided an example highlighting the different capabilities of the PCA and auto-encoder techniques for compressing 2D data to 1D.

Chapter 4

The HIVE Facility

4.1 Introductory Remarks

The first goal of this chapter is to provide more details of the experimental setup of the HIVE facility. This is essential to establish a computational model of HIVE which accounts for the different aspects of the experiment. The second goal is to highlight the current shortcomings of HIVE, specifically the diagnostics which provide limited insight in to the component's suitability, and the inaccuracy and inefficiency in creating a DoPE for a component.

This chapter is structured as follows: firstly, the motivation for establishing HIVE is provided in section 4.2. Section 4.3 provides a more detailed description of the HIVE facility and how experiments are performed, while section 4.4 discuses some common outcomes the HIVE operators aim to deliver during an experiment.

4.2 Motivation

4.2.1 Nuclear Fusion

Fusion is the process of combining two lighter nuclei to create one heavier nucleus, and powers the Sun and all the stars of the universe. For fusion to occur, two nuclei must overcome their electrostatic repulsion to one another and move within close proximity, where the attractive nuclear force enables them to fuse. The extreme pressure produced by the Sun's immense gravity means that fusion is achieved at around 15,000,000 °C. Because Earth's gravitational field is 28 times weaker than the Sun's, the reduced pressure means that temperature in excess of 150,000,000 °C is needed to achieve fusion. Generating this level of heat is usually achieved through a combination of Ohmic heating and neutral beam injection [138].

Fusion can be achieved using a variety of different fuels, however a mixture of deuterium and tritium, usually referred to as a D-T reaction, is the most widely used as it delivers an optimal fusion triple product score [139]. This score is – unsurprisingly – the product of three factors; the density of ions within the plasma, the temperature of those ions and the energy confinement time. A D-T reaction maximises the triple product score for a lower temperature compared with other fuel mixtures.

Deuterium and tritium are both isotopes of hydrogen, with the former having two particles in its nucleus (one proton and one neutron) while the latter having three (one proton and two neutrons). These two fuse to create helium and an additional neutron, releasing 17.6 MeV in the process. This is summarised in fig. 4.1, where ^{2}H and ^{3}H are deuterium and tritium respectively.

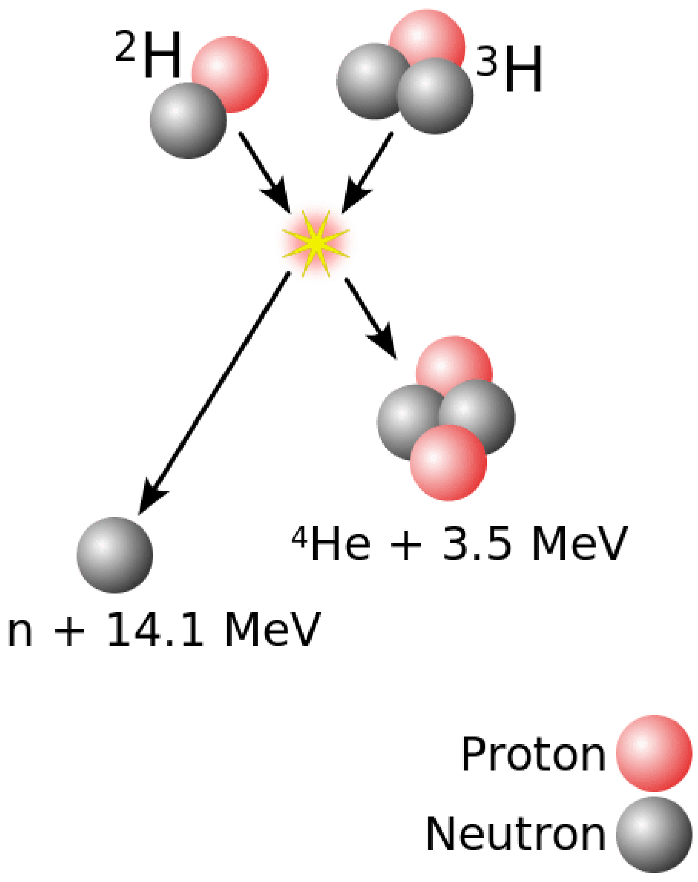


Figure 4.1: Deuterium-tritium fusion reaction

4.2.2 Fusion Devices

The most commonly followed approach for achieving fusion on Earth is through the use of a tokamak. Tokamaks are characterised by their use of magnetic coils to generate the necessary magnetic fields to confine a plasma into a torus (doughnut) shape. When a plasma is achieved, the negatively charged electrons and the positively charged nucleus become detached from one another, enabling the plasma to be controlled using a magnetic field. The use of these magnetic fields ensure that the extremely hot plasma will never come in to contact with the walls of the tokamak. To generate the necessary magnetic fields, superconducting magnets are used, whose temperatures can reach as low as -269 °C (4 °K) in the cryopump.

The world's largest tokamak is JET, and has been pioneering fusion for the past 40 years. In 1997, JET achieved a then world-record peak power output of 16.1 MW from a fusion device, along with a total energy output of 22 MJ achieved over a 5-second period. Commercial fusion devices will require longer pulse lengths and greater energy outputs, however JET is unable to perform longer pulse lengths than 5-seconds due to its superconducting magnet configuration. Following an upgrade to its inner wall, JET set a world record in 2021 for the largest total energy output over a sustained period, generating 59 MJ of energy over a 5-second period [140].

To highlight the efficiency of fusion, this 59 MJ of energy was achieved using a plasma fuelled with 0.1 mg of tritium and 0.07 mg of deuterium, whereby to produce the same quantity of energy one would require more than 2 kg of coal or more than 1 kg of natural gas [140].

Due to its size and design, JET will never be able to produce net energy. This means that its fusion gain factor, Q, which is the ratio of fusion power produced in a nuclear fusion reactor to the power required to maintain the plasma in steady state, is less than 1.

Scheduled for completion in 2025, ITER will become the world's most expensive and complex science experimental facility [141]. Located in Cadarache, France, ITER is a tokamak which has been designed using concepts tested out in JET. ITER is much larger than JET, boasting a major radius of 6.2 m, more than double that of its predecessor (2.96 m). This increase in size means that the total plasma volume of ITER is 840 m^3, a substantial increase compared with the 100 m^3 available in JET. Increasing the volume of the plasma greatly improves the plasma efficiency, meaning that ITER will be able to demonstrate a fusion energy gain factor of $Q = 10$. A conceptual design of the ITER facility is shown in fig. 4.2.

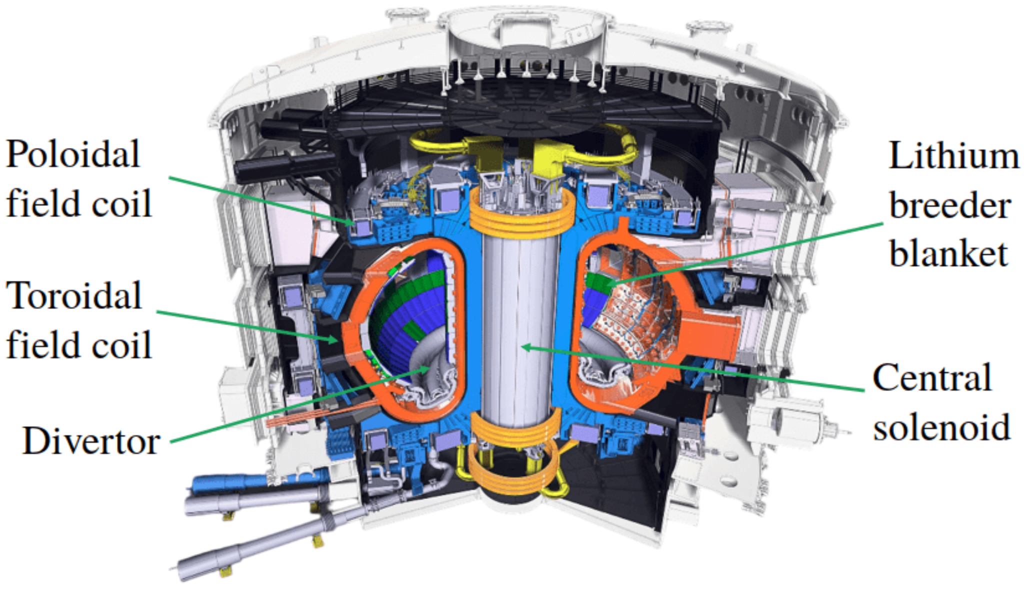
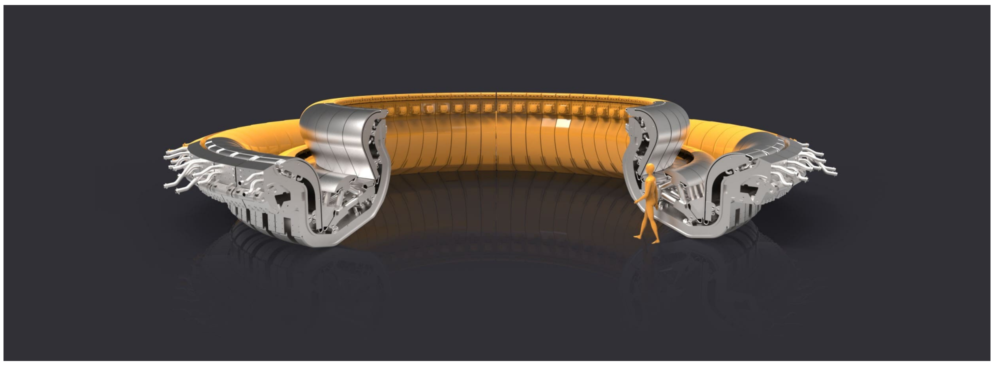


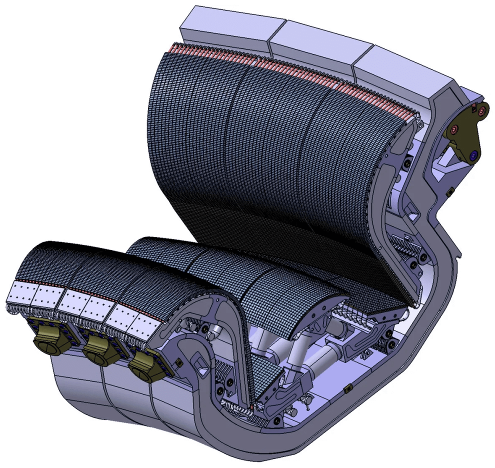
Figure 4.2: ITER conceptual design [142].

Although ITER will demonstrate net power, it will never supply energy to the grid. Its main purpose is instead to test and develop technologies, designs, and diagnostics for use in the next iteration of prototype fusion power plants, such as the design of the divertor [143]. The plasma is exhausted via a part of the tokamak known as the divertor, which sits at the bottom of the tokamak, as highlighted in fig. 4.2. A mock-up of the ITER divertor is shown in fig. 4.3a, which is made up of 54 cassettes, each weighing 10 tonnes each. These cassettes consist of 3 main components; an inner and outer vertical target and the dome, see fig. 4.3b. During steady state operation, the vertical targets must sustain a heat flux of $10\,\mathrm{MW/m^2}$ while during the slow transient stage this will increase to $20\,\mathrm{MW/m^2}$ [144]. These targets are made up of monoblocks, which consist of an armour and a pipe which actively cools the component using coolant, see fig. 4.3c.

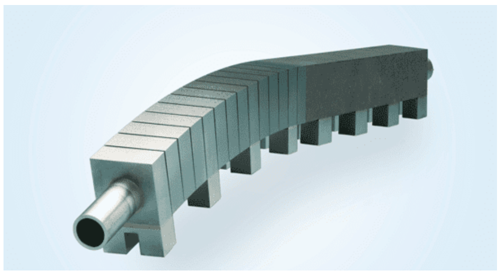
The design of the ITER monoblocks has been the subject of an international R&D operation. Initially, the monoblock armour was designed using carbon, however carbon has extremely high tritium retention rates, making them unsuitable [145]. Instead, tungsten was chosen as the armour material due to its extremely



(a) Full assembly showed to scale [144]



(b) Divertor cassette (x3)



(c) Monoblock assembly

Figure 4.3: ITER divertor

high melting point and low tritium retention, while the cooling pipes are made of a copper chromium zirconium alloy. In preparation for the construction of ITER, the inner wall and divertor of JET was upgraded to an ‘ITER-like-wall’, which was in place when the world-record 59 MJ pulse was achieved [146].

A key measure for critical components such as the monoblock is the ratio of mean time to failure (MTTF) and mean time to repair (MTTR). Reduction of the MTTR will be achieved through the use of robotic maintenance, however without a large enough MTTF a fusion power plant will never be commercially viable as large periods of time will be spent with the plant in shut down while old parts are removed and replaced with new ones.

4.2.3 HHF Facilities

The design of PFCs, especially those in the divertor, are still a heavily researched topic as they contribute to a variety of other fusion projects. The demonstration power plant EU-DEMO will learn from ITER and improve on its design, aiming to provide between 300 MW and 500 MW of net electricity to the grid during the 2050s [147]. There are also projects investigating the use of spherical tokamaks for fusion devices. Spherical tokamaks have lower aspect ratios compared with conventional tokamaks such as JET and ITER. This lower aspect ratio provides greater plasma stability and higher magnetic fields [148], enabling plasma pulses to last longer. The spherical tokamak for energy production (STEP) is a prototype fusion device which aims to provide net electricity to UK national grid by 2040 [149]. This will build on the knowledge gained from the mega amp spherical tokamak upgrade (MAST-U) facility currently in operation at UKAEA's Culham site [150].

The primary goal of this continued PFC research is to increase their lifespan, resulting in an improved MTTF. Increasing the lifespan of components can be achieved in multiple ways. It can use novel designs features, such as the inclusion of castellations on the plasma facing surface, using newly developed materials and alloys, or use alternative manufacturing methods, such as additive manufacturing (AM) [151]. Any newly designed component must be tested to ascertain its suitability when subjected to the fusion relevant loads.

Experimental facilities play an integral part during the design cycle of engineering components. Fig. 4.4 shows the pyramid of tests a component is subjected to during its design. At the bottom are coupons which are small parts which are easy to test, and thus are tested frequently. At the top of the pyramid is the full assembly, which due to its cost to may only get tested once. An example of this is a bird strike on an aircraft engine. The blades of the engine will be frequently tested to assess their response to a high impact, however an entire engine will only be tested a handful of times due to the cost of producing them [152].

Due to the multi-physics nature of fusion, testing PFCs for the whole gamut of different conditions outside a fusion device is impossible. Instead, components such as the monoblock are tested in HHF facilities to assess their thermal performance.

There are a number of HHF testing facilities which can replicate certain fusion-related loads. The Magnum-PSI facility, located at the Dutch Institute for Fun-

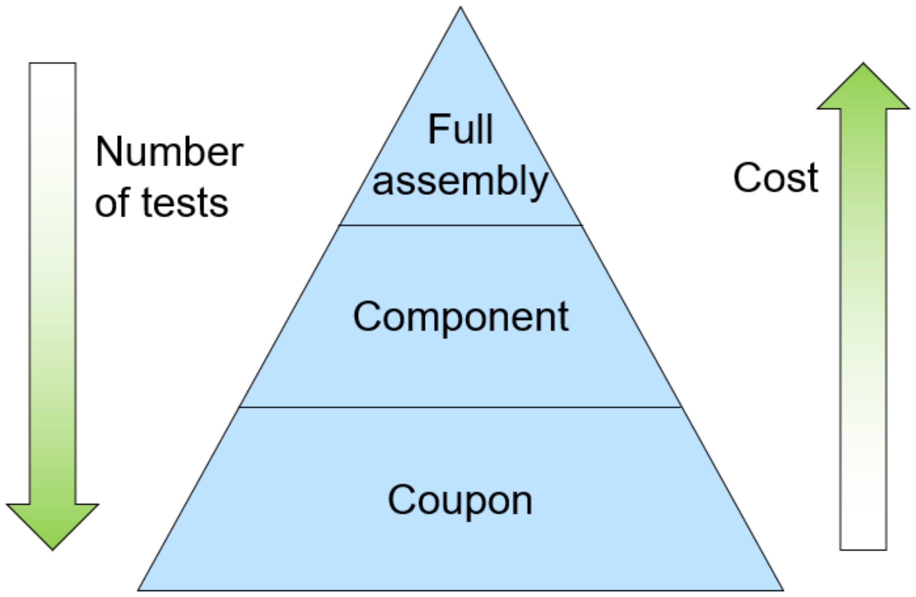


Figure 4.4: Pyramid of physical testing.

damental Energy Research, uses a hot, dense plasma guided by superconducting magnets to heat the target [153]. This replicates the particle fluxes a component is exposed to in a fusion device and can deliver over 10 MW/m^2 of surface power to a component.

Located at the Max Planck institute for plasma physics, GLADIS is equipped with two hydrogen ion sources to deliver up to 45 MW/m^2 of surface power to a component [154]. Its large scale means it can test components which are up to 2 m in size, which is useful for testing the entire inner vertical target, which is 1.5 m in length [155]. Other facilities where high flux testing takes place are the ITER divertor test facility and JUDITH II [156], [157].

Scheduled for completion in 2024, the Combined Heating and Magnetic Research Apparatus (CHIMERA) will be the only machine in the world able to test components under the unique combination of conditions encountered in large fusion devices, such as ITER [158]. Up to 20 MW/m^2 of surface power can be delivered to a component while it is subjected to mechanical loading via a 4 Tesla magnetic field.

Due to the size and complexity of HHF facilities, their use comes at a substantial cost. This sizeable outlay is acceptable during the latter stages of the design cycle to confirm a component's suitability, however during the early stages this cost often becomes an obstacle to innovation. The HIVE facility was established in 2015 to enable early verification of the thermo-fluid performance of PFCs. Instead of focusing on full component qualification, HIVE focuses on the comparison between different concepts and manufacturing methods. The primary goals of HIVE are [10]:

- Analyse the thermal performance of a component.
- Investigate concepts which employ novel features or materials.
- Provide a comparison between different manufacturing methods.
- Validate computational models.

4.3 Experimental Setup

The HIVE vacuum vessel, inside which components are tested, is shown in fig. 4.5. This vessel has height and diameter of 500 mm, and consists of 6 ports located around its circumference. Two ports are assigned to vacuum pumping and monitoring, while the other four provide a range of viewing angles through windows ranging from 100 mm to 200 mm in diameter. The pressure in the vessel is reduced to 1E-06 mbar during an experiment to protect the experimental apparatus from the high temperatures subjected to a component.

To perform an experiment, firstly the component is fitted to the mounting bracket with the pipe attached to the inflow and outflow of the coolant, as shown in fig. 4.6. Next, the coil design is chosen and is positioned relative to the component, however it is difficult to accurately measure the exact positioning of the coil due to the scale of the problem. Once positioned, the lid is then fitted and bolted to the vacuum vessel, in order for it to be depressurised. Finally, the parameters relating to the coolant and induction heating system are set.

During an experimental campaign, changing the loading conditions a component is subjected to is usually achieved by varying the coolant and induction heating parameters. This is because these are easily accessible, while changing the induction coil and/or its positioning requires substantially more effort.

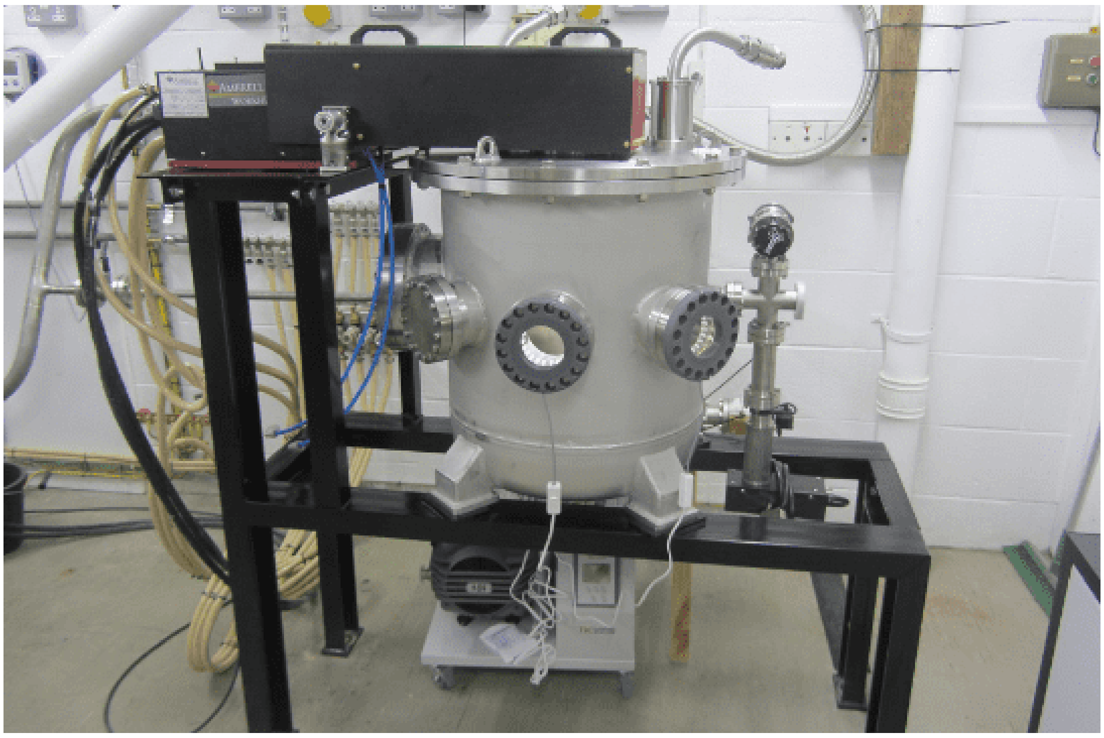


Figure 4.5: HIVE vacuum vessel

4.3.1 Induction Heating

Induction heating is the process of heating electrically conductive materials, such as metals, via electromagnetic (EM) induction. An alternating current (AC) is run through a coil, producing a rapidly changing magnetic field. This varying field generates electric currents (often referred to as eddy currents) within the nearby conductor (component in HIVE). These eddy currents are converted in to a heat source based on the electrical resistivity of a material in a process known as Joule heating, or resistive heating. The electric resistivity of a material is its measure of how strongly it opposes the flow of an electric current, with larger values yielding greater heat generation. An image of an induction coil used in HIVE is shown in fig. 4.7.

EM induction is a form of volumetric heating, however it's a surface heat flux which a component in a fusion device would be subjected to. The current-carrying layers in the conductor can be found between the surface adjacent to the coil and a level called the skin depth, d. The formula for the skin depth is given by eq. 4.1, where ρ is the electrical resistivity of the conductor, f is the frequency of the current in the coil, μ_r is the relative magnetic permeability of the conductor and μ_0 is the magnetic permeability of free space in a vacuum, which

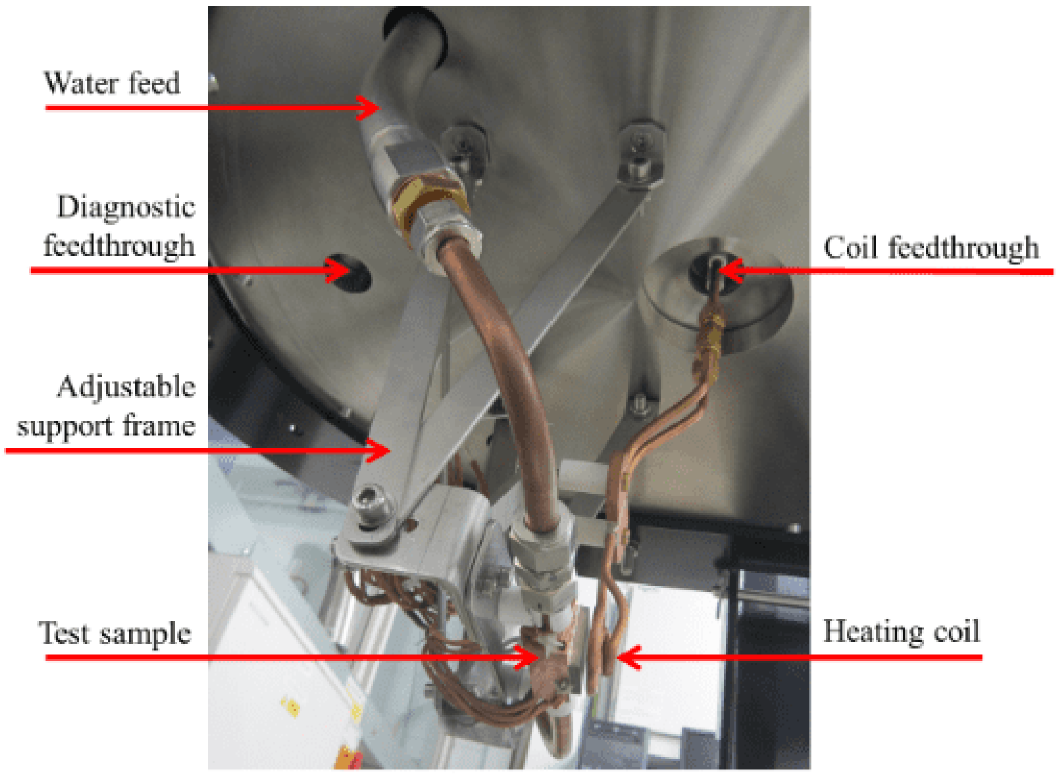


Figure 4.6: View of vacuum vessel lid from below, showing coil and sample mounting arrangement [159]

is $4\pi \times 10^{-7}$ [160]. The magnitude of the eddy currents decreases exponentially from the coil adjacent surface [161].

$$d = \sqrt{\frac{\rho}{\pi f \mu_0 \mu_r}} \tag{4.1}$$

Since the skin depth is inversely proportional to the current frequency, it can be made smaller by operating at higher frequencies. Fig. 4.8 shows the skin depth as a function of the frequency for tungsten, from which the PFCs in ITER are made. The induction heating system used by HIVE has a frequency range of 50 kHz to 150 kHz, and usually operates around 100 kHz, meaning that the skin depth would be below 0.5 mm. It's imperative that the skin depth be suitably small so that thermal loads generated by the induction heating system accurately replicates the surface heat flux imparted by the plasma.

The magnitude of the induction heating loads are decided using the power settings on the induction heating system, which has an upper limit of 45 kW.

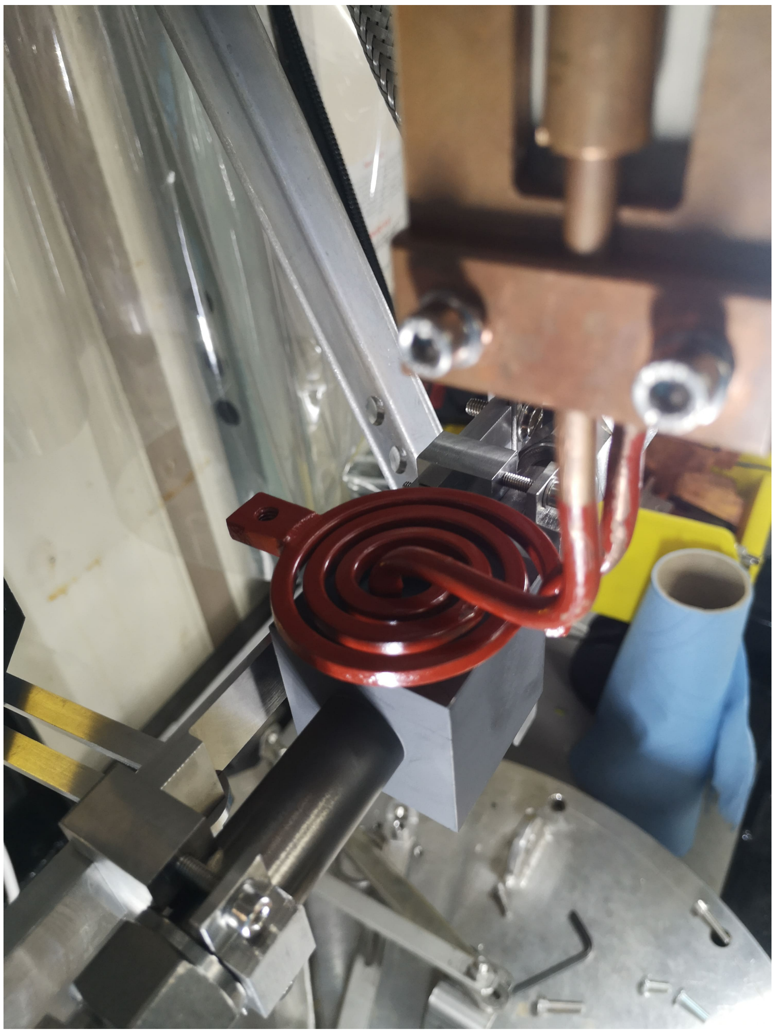


Figure 4.7: Induction coil and component prior to testing in HIVE.

This upper limit of 45 kW isn't the quantity of thermal power which will be delivered to the component, rather the quantity of power extracted from the power supply. The quantity of thermal power delivered to the component depends on the coupling efficiency of the induction coil, given by eq. 4.2. This coupling efficiency is inversely proportional to the square root of the distance between the coil and conductor [162]. This means that very small changes in the positioning of the coil relative to the component in HIVE will have a large effect on the coupling efficiency and thus the heating power delivered to the component.

$$Coil\ efficiency = \frac{Energy\ transferred\ to\ the\ conductor}{Energy\ delivered\ to\ the\ coil} \tag{4.2}$$

Although there are correlations available in literature to estimate the coupling

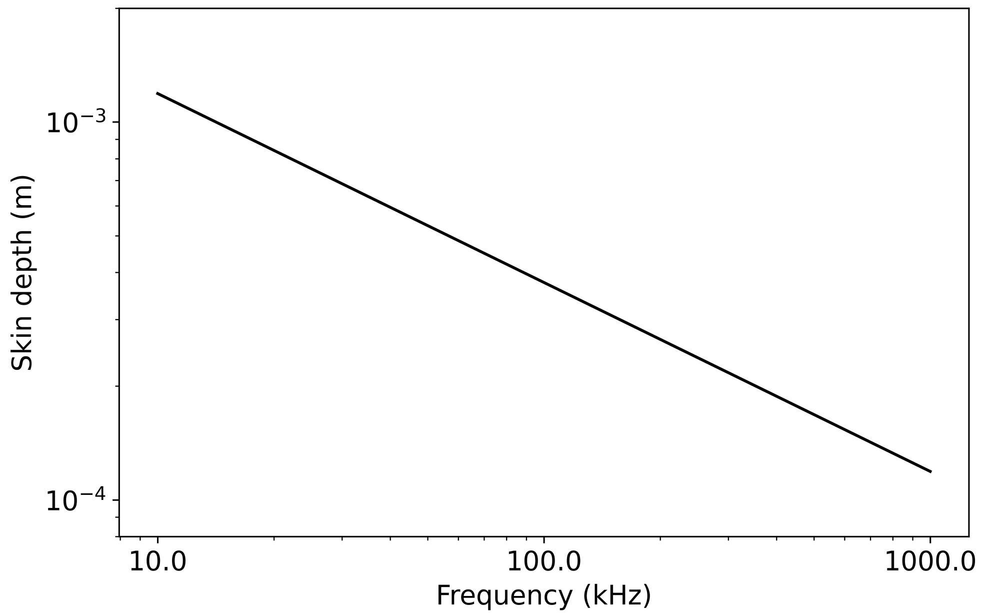


Figure 4.8: Skin depth of eddy current penetration against frequency for tungsten

efficiency of an induction coil, the sensitivity of the coil placement along with the presence of immeasurable losses in the system means that these estimates have a high degree of variability [162].

4.3.2 Active Cooling

Cooling is supplied to HIVE using a closed loop temperature control unit. Through this the temperature, pressure, and flow rate of the coolant are set. The cooling rig used in HIVE allows for a flow rate ranging from 5 l/min to 80 l/min, pressure between 0.4 MPa and 2 MPa, and temperature between 25 °C and 200 °C, although these higher temperatures are only achievable when the coolant is also at high pressure. The coolant input and output are mounted to the removable lid of the vacuum vessel, as shown in fig. 4.6.

4.3.3 Diagnostics

Unfortunately, the data recorded by HIVE's diagnostics are limited by operational constraints. An infrared (IR) camera is used to record the experiment, however the point of view of the footage is restricted by the location of the portholes in the vacuum vessel. Along with this, the proximity of the induction coil to the component means that the view of the component is obstructed, see fig. 4.9. This is especially problematic since this inhibits the ability to view the temperature field on the coil adjacent surface.

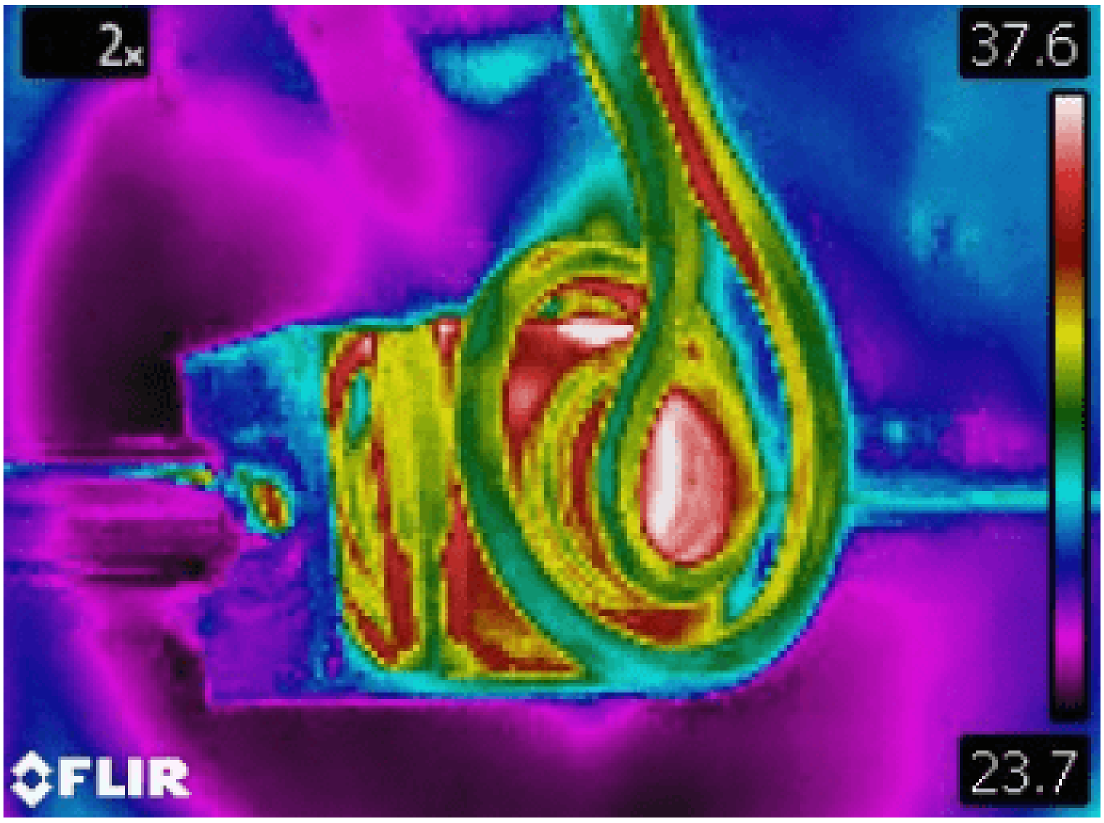


Figure 4.9: IR image recorded during HIVE experiment [159]

Thermocouples, which are probes that are joined on to the surface of a component, are used to record pointwise measurements of the temperature at a limited number of locations on the component's surface. Their accuracy, however, is highly dependent on the strength of its bond to the component, meaning that these can be prone to underestimating the true temperature. Fitting these thermocouples to the surface of each component is also very time-consuming process.

There are also 2 pyrometers available to measure surface temperature, however, similarly to the IR camera, these are located outside the vacuum vessel and are thus limited by the location of the portholes on the vacuum vessel. Moreover, each pyrometer is only able to record temperatures above 300 and 350 °C respectively, meaning they are unusable for a number of experiments.

4.4 Desirable Experimental Outcomes

To adequately test a component's suitability for a fusion device, it is desirable to subject it to a variety of different loading conditions during its DoPE. As HIVE is only capable of monitoring a component's thermal response, this is the only measure around which the DoPE can currently be constructed. Thus, some common loading conditions currently requested are;

- **Ensure the peak temperature stays below a certain critical temperature such that the component isn't damaged.** It is desirable for components to be subjected to similar heat loads as those experienced within a fusion device, however it is essential that the temperature of the component stays within their service range, and critical that they are below the melting temperature.
- **That a certain temperature is reached within a particular region of the component.** As components tested in HIVE are heat exchanger components, it is often desirable that a specified temperature be delivered to a particular region.
- **Uniformly heating the coil adjacent surface as much as possible to avoid hot spots.** The heat loads experienced by PFCs in a fusion device are uniform, with variations occurring over a much larger lengthscale. To ensure that the components tested in HIVE are suitable, it is essential that the heating be as uniform as possible to emulate in service conditions.
- **Identify power settings to gradually scale up the component temperature.** As components in HIVE are commonly made from bonded metals, it is advisable to heat up the component gradually to specified temperatures. This avoids a scenario where a component is pushed too far and fails at the first experiment, resulting in no usable experimental data.

Due to HIVE's limited diagnostics, it's very difficult to accurately assess whether these outcomes have been achieved. Currently, HIVE operators use the data collected from thermocouples to estimate the temperatures at different locations in the component using their intuition. Given that HIVE is a multi-physics, highly non-linear experiment, it is extremely unlikely that estimates are accurate. This is exacerbated by the fact that HIVE predominantly tests components with novel features, materials, and manufacturing methods, on which there is limited data available to provide the necessary intuition, thus increasing the level of uncertainty. There is also no means my which to validate these estimates, making them extremely unreliable.

Since it is the stress state of the component which is of most concern during the design of PFCs, an ideal set of loading conditions set out in the DoPE would take this into account. For example, these could include;

- **Incrementally increasing the stress in a component**. It is beneficial to know how far a component can be pushed prior to failing to ascertain its limits.
- **Minimising stresses at bonded surfaces.** It is essential that components do not fail before usable data has been gathered. Minimising the stress on bonded surface gives component's the best chance of avoiding failure.
- **For a given component temperature, identify the upper and lower limit of the maximum stress state.** There are often numerous combinations of experimental parameters which will deliver a specified maximum temperature to a component. Each of these will result in a different maximum stress in a component, the knowledge of which is extremely useful.

Given that HIVE operators have no method of inferring the stress state of the component from the sparse thermal experimental data available, delivering these types of experiments is currently impossible.

4.5 Chapter Summary

This chapter has provided an overview of the HIVE facility, including the motivation behinds its establishment. The details provided on the experimental setup is essential for the creation of a computational model of HIVE, which will be discussed in chapter 5.

The discussion surrounding the diagnostics available in HIVE and the way in which the DoPE is established provides the motivation for the use of ML. ML offers the opportunity to greatly improve the impact that the HIVE facility has during the design cycle of a component. Smarter testing would ensure that a component is more thoroughly tested for a greater variety of loading conditions, the work for which is presented in chapters 6 and 7. Improved insight ensures that the sparse experimental data is enriched to provide a much better understanding of the component's suitability for a fusion device, which is presented in chapter 8.

Chapter 5

VirtualLab

5.1 Introductory Remarks

The primary aim of this chapter is to provide details on the multi-physics computational model of HIVE which has been developed for this project. This model has been created using the *VirtualLab* package, which has been developed predominantly by myself as part of this project [43]. *VirtualLab* is a novel package which provides more automation to computational engineering workflows, enabling tools such as ML to be more easily utilised, overcoming many of the challenges outlined in section 2.5. While an overview of *VirtualLab* is provided, more detailed information including its installation, structure, use, and guidelines for contribution can be found in the documentation [44]. The documentation also includes six detailed tutorials, which guides new users through the use of *VirtualLab* showcasing its flexibility for different workflows and scenarios.

Firstly, section 5.2 provides more details on the *VirtualLab* package, including details on how parameterisation, parallelisation, and portability is achieved. Following this, section 5.3 provides a detailed explanation of the computational model created for HIVE, along with its validation against experimental results.

5.2 Package Overview

VirtualLab is a platform which streamlines the production of synthetic data of physical systems to promote the use of ML algorithms. Central to the vision of *VirtualLab* are the three P's; parameterisation, parallelisation, and portability. Parameterisation of the workflow enables key variables to be easily changed, making it easier to collect data from across a parameter space. Parallelisation allows simulations to be performed concurrently, which when deployed on supercomputers enables data to be generated in a fraction of the time. The easy portability of *VirtualLab* means that its deployment and use is the same regardless of the computer system. This ensures that it can be easily used by those in different teams or organisations.

Written mostly in Python, *VirtualLab* centres around a **class** which sets up, organises and executes the required steps of the workflow. These steps are set out in a Python script, referred to as a 'run file', an example of which is shown in listing 5.1. This run file is a rather simplistic example which follows a linear workflow, however more complex workflows can be easily implemented.

```
1
2   from Scripts.Common.VirtualLab import VLSetup
3
4   Simulation='Tensile'
5   Project='Tutorials'
6   Parameters_Master='TrainingParameters'
7   Parameters_Var=None
8
9   VirtualLab=VLSetup(
10              Simulation,
11              Project)
12
13  VirtualLab.Settings(
14              Mode='Headless',
15              Launcher='Process',
16              NbJobs=1)
17
18  VirtualLab.Parameters(
19              Parameters_Master,
20              Parameters_Var,
21              RunMesh=True,
22              RunSim=True,
23              RunDA=True)
24
25  VirtualLab.Mesh(
26              ShowMesh=False,
27              MeshCheck=None)
28
29  VirtualLab.Sim(
30              ShowRes=False)
31
32  VirtualLab.DA()
```

Listing 5.1: *VirtualLab* run file

To initiate the **class**, two inputs are required; the experiment and the project (line 9 of listing 5.1). The former simply refers to the name of the experiment for which analysis will be performed, e.g. 'Tensile' to perform a tensile test. The name provided here allows *VirtualLab* access to the relevant scripts needed to perform that particular virtual experiment. The project name is the name of the directory in which the input parameters are found (in the input directory) and where results will be saved to (in the output directory).

Once an instance of a class is created, the next step is to use the Settings attribute to define how one wishes to use *VirtualLab* (line 13). Here, decisions are made, such as where the standard output is written to (either the terminal or to a file) or how parallelisation is achieved. The use of settings is optional, without which the default values are used.

Next, the Parameters attribute is used to read in the parameterised values which will be used to run the workflow (line 18). Parameterisation is an integral part of *VirtualLab* which is more easily explained once the concept of 'methods' have been introduced.

5.2.1 Methods

Methods are routines which perform different aspects of the workflow. In listing 5.1, the methods used are Mesh, Sim and DA (data analysis), which are ubiquitous steps in most engineering workflows. VirtualLab has a variety of different methods currently implemented;

- **Mesh**: Routine to create CAD geometries and meshes of components using *SALOME*. Visualisation of the created meshes in the GUI is possible by means of the ShowMesh keyword, allowing the user to visualise the fidelity of the mesh. This method also has the CheckMesh keyword which generates the mesh in the GUI, which is useful for debugging any errors which may arise, such as identifying parts of the geometry which are incorrectly intersecting one another.

- **Sim**: Routine which perform the simulation of the experiment. Currently, this method uses the FE code *code_aster* developed by Electricite de France (EDF) to perform the necessary analysis. This method has the ShowRes keyword available to automatically open the generated results in the visualisation software *ParaViS*, which is a customisation of the *ParaView* package by EDF.

- **ML**: This routine uses data extracted from simulations to construct surrogate models using the PyTorch package. A number of useful functions have been developed to make the construction of SV and FF surrogates very simple.

- **DA**: This method is extremely flexible, allowing the user full control over what analysis to perform. This can include extracting key results from simulations for comparison, creating a database of results, capturing images using *ParaViS*, or using surrogate models to perform analysis.

- **ScanCT**: Module which creates simulated X-ray radiographs using the gVXR package [163].
- **Voxelise**: Module which converts CAD geometries in to voxelised representations using the Cad2Vox package [164].

Each method is itself a Python **class**, and are defined separately from the *VirtualLab* parent class but are attached during its initiation. This enables methods to be easily added and modified by the user, allowing a high degree of customisation and flexibility when using *VirtualLab*.

5.2.2 Modules

As seen in the above description, most methods use at least one external software packages to perform their necessary analysis. In *VirtualLab*, these external packages are referred to as ‘modules’, since they can be easily swapped for a different package or a more up-to-date version. This is made possible through the use of containers. A container image, the term commonly used for an individual container, communicates with the host system through the use of a container platform such as Docker, Singularity or Apptainer. While the inside of an image contains the specific OS and dependencies a code requires, on the outside they are identical, meaning they can be deployed on any computing environment which has a container platform installed.

To ensure *VirtualLab* can be used by those working in any organisation, the modules currently used are all open-source software (OSS) packages. OSS are distributed freely, meaning there is no cost whether they are on a personal computer or a multi-node supercomputers. Along with this, generally speaking OSS produce better scaling than commercial codes due to their frequent use in academia. A common drawback faced with OSS is that they are not always well maintained, meaning that they may become defunct with upgrades to external packages, for example. Another obstacle is that OSS do not provide formal support in the same way that commercial software does, although many popular OSS have active user communities that provide extensive assistance (e.g. OpenFoam, *code_aster*), while some offer support through a subscription (e.g. GitLab). It is essential, therefore, that OSS are carefully chosen.

The modules currently available in *VirtualLab* are;

- *SALOME*: *SALOME* is a pre- and post-processing OSS developed in a partnership between EDF and Commissariat a l'energie atomique (CEA), the French equivalent to UKAEA [165]. *SALOME* does not contain a numerical solver, but it provides the computing environment necessary for their integration. It is programmed using a combination of C++ and python. CAD geometries are created using its inbuilt GEOM and Shaper modules, with the latter used to make more complex geometries. Meshing is performed using the SMESH module, which supports the use of well known meshing algorithms such as NETGEN.

  For post-processing, *SALOME* has *ParaViS* in-built, which is an extension of the *ParaView* visualisation package to support the MED mesh format which is preferred by *SALOME*.

  As it is used in-house by both EDF and CEA, it is well maintained and well documented. Because it has a large network of users, its forums are frequently used, which is useful for identifying issues.

- *code_aster*: *code_aster* is an open-source FE code with over 40 years of development [166]. It is developed and maintained by EDF and used as their in-house solver. It consists of numerous analysis methods, including mechanical, thermal, acoustic, dynamic, metallurgic and porous media. It supports more than 400 different types of finite elements, including discontinuous media for modelling cracks and joints, along with over 200 constitutive laws to model a variety of different behaviours. As this code is used to model components used in EDF's nuclear energy facilities, its validation is rigorous, which is demonstrated for more than 2000 benchmark cases.

  Although *code_aster* is a terminal-based solver, a GUI is available for specific versions which have been integrated in to the *salome-meca* software (e.g. version 15.6 of *code_aster* is used in *salome-meca* 2021).

- *ERMES*: The Electric Regularized Maxwell Equations with Singularities (*ERMES*) is an open-source EM-FE solver developed by Ruben Otin at UKAEA [167]. Written in C++, *ERMES* is a benchmarked code and is efficiently parallelised using OpenMP [168]. It can model a variety of modelling types and BCs and can use up to third order elements.

Natively, the *ERMES* solver is integrated in to the pre- and post-processing software *GiD* to provide an easy-to-use GUI [169]. This software is a commercial package and is limited in its flexibility and automation capabilities. To overcome this, an application within *VirtualLab* has been created which removes the need for *GiD* when performing *ERMES* analysis. *ERMES* is launched using text files which contain the necessary information to perform analysis, such as the mesh data, BCs and solver parameters. Mesh information is extracted from the MED file format generated by *SALOME*, which is then written in the appropriate format alongside the necessary BCs required to simulate the EM induction performed in HIVE [170]. Once the *ERMES* solver is complete, the results are converted from the ASCII file format to the MED format so that the results can be visualised in *ParaViS*.

- Cad2Vox: Cad2Vox is a package developed by Ben Thorpe at Swansea University to efficiently perform mesh voxelisation on GPU (using CUDA) or CPU (using OpenMP) for surface and volume meshes [164].
- gVXR: This provides a programming framework for simulating X-ray images on the graphics processor unit (GPU) using OpenGL [163].

A visualisation of the methods and modules used to perform a tensile test in *VirtualLab* is shown in fig. 5.1.

5.2.3 Parameterisation

Parameterisation of the workflow is made possible using the aforementioned Parameters attribute of the *VirtualLab* class. Passed to this is Parameters_Master which is a pointer to the variable names and values one wishes to use during the analysis. In the majority of cases this pointer is the name of a file which stores the required information, as is the case in listing 5.1 which points to the file 'TrainingParameters.py' (defined in line 6).

The contents of 'TrainingParameters.py' is shown in listing 5.2. As shown, the parameter information is communicated to the specific method through the use of SimpleNamespace, which are effectively an empty Python **class** that data can be assigned to [171]. These are a similar construct to Java classes or structure arrays in Matlab. This means that any data assigned to the namespace Mesh in available to the method Mesh during its call in the run file. Based on the contents of this file, a mesh will be created, and a simulation performed, with no data analysis carried out (since there is no namespace DA defined in this file).

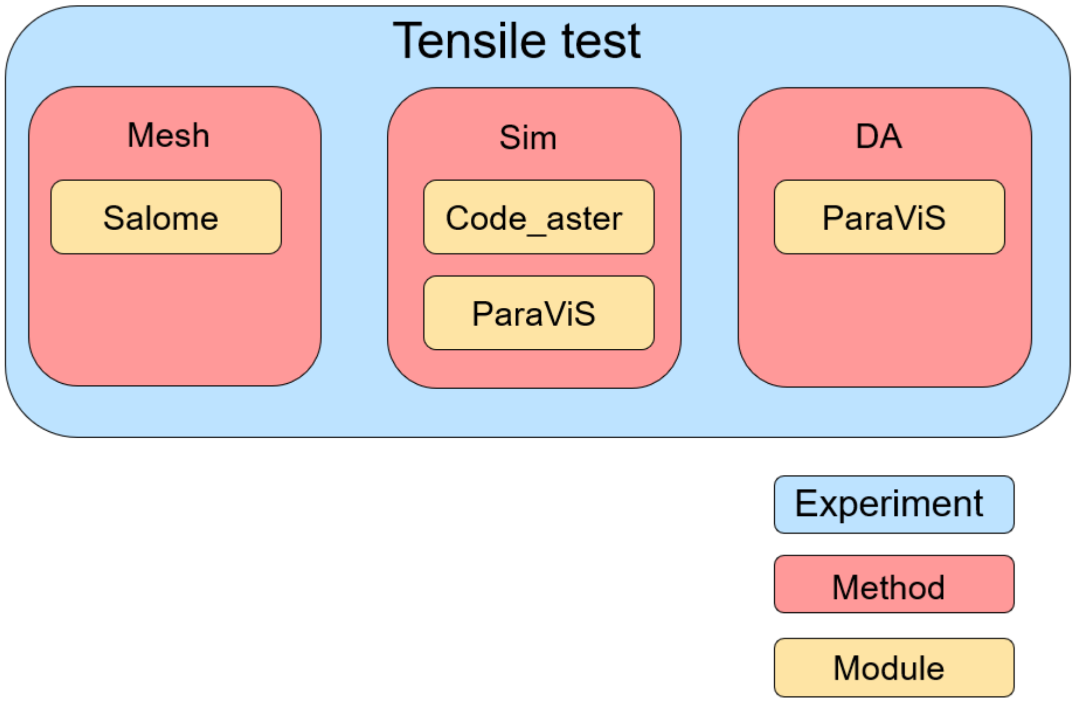


Figure 5.1: The methods and modules used for the tensile virtual experiment

```
from types import SimpleNamespace as Namespace

######## Meshing #########
Mesh = Namespace()
Mesh.Name = 'Notch1' # Name under which the mesh will be saved in Meshes directory.
Mesh.File = 'DogBone' # Salome python file used to create mesh.
# Geometric Parameters
Mesh.Thickness = 0.003
Mesh.HandleWidth = 0.024
Mesh.HandleLength = 0.024
Mesh.GaugeWidth = 0.012
Mesh.GaugeLength = 0.04
Mesh.TransRad = 0.012
Mesh.HoleCentre = (0.0,0.0)
Mesh.Rad_a = 0.0005
Mesh.Rad_b = 0.001
# Meshing Parameters
Mesh.Length1D = 0.001 # Discretisation along edges (1D)
Mesh.Length2D = 0.001 # Discretisation on faces (2D)
Mesh.Length3D = 0.001 # Discretisation on volumes (3D)
Mesh.HoleSegmentN = 30 # Number of segments for hole circumference

####### Simulation #######
Sim = Namespace()
Sim.Name = 'Single' # Name under which the simulation results will be saved.
Sim.AsterFile = 'Tensile' # The CodeAster command file can be found in Scripts/$SIMULATION.
# Simulation parameters
Sim.Mesh = 'Notch1' # The mesh used in the simulation.

Sim.Force = 1000000 # Force applied in force controlled analysis.
Sim.Materials = 'Copper' # Material component is made of.
```

Listing 5.2: Content of Parameter master file 'TrainingParameters'

Each namespace has the attributes File and Name attached to it. The attribute File specifies the name of the file used to perform the required analysis. This file is located in the directory 'Experiments/#experiment name#/#method name#' within the *VirtualLab* directory 'Scripts'. For example, the path to the mesh related scripts for the tensile experiment is 'Experiments/Tensile/Mesh', inside which 'DogBone.py' will be found.

This file then uses the other attributes associated with the namespace to perform analysis. For example, in listing 5.2 the namespace Mesh has attributes such as Thickness and HandleWidth to specify certain geometric dimensions, while Length1D specifies the mesh refinement along 1D edges of the CAD model. The attribute Name is the name under which the analysis will be saved. In this example, the mesh generated will be saved under the name 'Notch1' in the relevant mesh directory.

As stated above, the parameters do not necessarily need to be in a separate file, and can be defined directly in the run file. An example of this is shown in listing 5.3 for the purpose of generating a mesh.

```python
from Scripts.Common.VirtualLab import VLSetup
from types import SimpleNamespace as Namespace

Simulation='Tensile'
Project='Tutorials'

Mesh = Namespace()
Mesh.Name = 'Notch1' # Name under which the mesh will be saved in Meshes directory.
Mesh.File = 'DogBone' # Salome python file used to create mesh.
# Geometric Parameters
Mesh.Thickness = 0.003
Mesh.HandleWidth = 0.024
Mesh.HandleLength = 0.024
Mesh.GaugeWidth = 0.012
Mesh.GaugeLength = 0.04
Mesh.TransRad = 0.012
Mesh.HoleCentre = (0.0,0.0)
Mesh.Rad_a = 0.0005
Mesh.Rad_b = 0.001
# Meshing Parameters
Mesh.Length1D = 0.001 # Discretisation along edges (1D)
Mesh.Length2D = 0.001 # Discretisation on faces (2D)
Mesh.Length3D = 0.001 # Discretisation on volumes (3D)
Mesh.HoleSegmentN = 30 # Number of segments for hole circumference

Parameters_Master = Namespace(Mesh=Mesh)
Parameters_Var=None

VirtualLab=VLSetup(
           Simulation,
           Project)

VirtualLab.Settings(
           Mode='Headless',
           Launcher='Process',
           NbJobs=1)

VirtualLab.Parameters(
           Parameters_Master,
           Parameters_Var,
           RunMesh=True,
           RunSim=True,
           RunDA=True)

VirtualLab.Mesh(
           ShowMesh=False,
           MeshCheck=None)

VirtualLab.Sim(
           ShowRes=False)

VirtualLab.DA()
```

Listing 5.3: Self-contained run file

The advantage of this method is that the run file is self-contained, meaning that the analysis can be performed using only this single file. This method is also useful when one needs to perform analysis in an iterative loop, such as adaptive sampling, for example.

A second, optional parameter pointer, referred to as Parameters_Var, can be used in conjunction with Parameters_Master to perform parametric analysis. The file, which similarly uses namespaces, provides lists of values for parameters one wishes to vary during a DoE and supersede those similarly defined in Parameters_Master.

An example of a Parameters_Var file is shown in listing 5.4, which dictates that two meshes, named 'Notch2' and 'Notch3', will be created with varying values for two geometric parameters Rad_a and Rad_b. The other parameters associated with Mesh in Parameters_Master will be used alongside these to generate the two meshes. Two simulations will then be performed, 'ParametricSim1' and 'ParametricSim2', which perform the exact same analysis but using the two different meshes.

```
from types import SimpleNamespace as Namespace

######## Meshing #########
Mesh = Namespace()
Mesh.Name = ['Notch2','Notch3']
Mesh.Rad_a = [0.001,0.002]
Mesh.Rad_b = [0.001,0.0005]

####### Simulation #######
Sim = Namespace()
Sim.Name = ['ParametricSim1', 'ParametricSim2']
Sim.Mesh = ['Notch2', 'Notch3']
```

Listing 5.4: Structure of Parameter var file

Because the parameter files are Python based, the lists of parameter values in the Parameters_Var file can be generated using constructs such as **for** loops, thus making it much simpler to define very large numbers of analyses. *VirtualLab* also has an in-built sampling schemes, which are useful to choose points across a parameter space using a variety of different sampling techniques.

The Parameters attribute also has the capability of 'switching off' certain methods. For example, if one has already generated the necessary meshes previously, but it is still defined in the parameters file, then instead of deleting this information one can simply change the RunMesh keyword to False (line 21 in listing 5.1), resulting in the Mesh method being skipped. Similarly, this method could be avoided by removing the call to the Mesh method in the run file (lines 25-27 in listing 5.1).

5.2.4 Portability

So that *VirtualLab* can be a widely used tool, it is essential that it can be easily deployed on a variety of different operating systems. As discussed in section 5.2.2, the external software packages associated with *VirtualLab* are placed in containers, meaning that they can be used on any system with a suitable container platform. To reduce the number of requirements on the user's system, the python dependencies required for the *VirtualLab* package are also placed in a container, which is referred to as the 'manager' container. The benefit of this approach is it fixes the version of python and its dependent packages to those which are known to work, avoiding inevitable issues when using it with more recent, untested, versions. This ensures that reproducibility of results is much simpler using *VirtualLab*, which is a key component of scientific data management.

While the steps of the workflow are executed inside the manager container, analysis will also need to be carried out by external packages. This is achieved using a 'server' on the host system, which waits to receive commands from the manager container regarding which external software should be triggered. For example, once the workflow reaches the stage where *code_aster* is required, the necessary information is sent to the server, which then launches the *code_aster* container to perform the necessary analysis. Once complete, the server passes the necessary information back to the manager to continue working through the workflow. The communication between the server and the various containers is achieved using the Python socket module, which is a low-level network interface [172].

The socket module is a built-in module, meaning that only a standard installation of Python or Anaconda is required on the user's system, along with a suitable container platform and git. This means that *VirtualLab* is extremely easy to deploy, and has been demonstrated for a variety of different Linux operating systems on both personal computers and clusters at different organisations. Although not currently implemented, the setup of *VirtualLab* means that extending it to use on Windows or Mac operating systems is straightforward, and something which is considered as future work.

A slight drawback of this approach is that placing software in containers will require more storage than if they were installed on the host directly, due to certain container related requirements. Given that computers have large quantities of storage available these days, this is unlikely to be a limiting factor in its use.

5.2.5 Parallelisation

Parallel computing is a type of computation in which many calculations or processes are carried out simultaneously [173]. Its use, coupled with huge advancements in the computational power, has enabled the creation of large, extremely complex computational models that would previously have been thought impossible. Some examples are simulations which model the impact of defects such as inclusions and porosity in a component [174], modelling the movement of traffic through cities [175] and simulations of blood flow through arterial networks [176].

Parallelisation can be achieved in two ways. The first is characterised by the running of a single instance of a program, where computations are performed among the available processors, and is commonly referred to as HPC. This requires the need for a method of communicating between the different processors, such as message passing interface (MPI), and is usually used for larger problems which are subdivided between processors. The second involves running multiple, independent instances of a program over multiple processors simultaneously. Their independence means there is no need for communication between each processor and is used in scenarios where the same program is run with varying inputs. This is occasionally referred to as high throughput computing, but most of the time it's also referred to as HPC. It's also possible to use both forms in conjunction, e.g. if 10 instances of a program were required and 40 cores (or processors) were available, then it would be beneficial to perform the 10 instances and allowing each instance 4 cores to perform its parallelisation.

In *VirtualLab*, numerous analyses can be defined in advance using Parameters_Var and is a typical example of a high throughput computing problem. Fortunately, implementing this is much simpler than classical HPC parallelisation, as it doesn't require detailed knowledge of the program to identify the aspects which would benefit from parallelism. *VirtualLab* uses the Python packages pathos and pyina for this purpose [177], [178]. Pathos provides a consistent interface of the **map** and **apply** functions native to Python for several popular parallel libraries, such as threading [179], multiprocessing [180] and Parallel Python [181]. Pyina extends these functionalities for multi-node application, which is achieved through the use of MPI or job scheduling systems for clusters, such as slurm [182], [183].

To assess the performance of *VirtualLab*'s parallelisation, 32 identical simulations of the engineering tensile test were performed on the Sunbird computer cluster [184]. Shared between Swansea University and Aberystwyth University, Sunbird is a 126 node cluster, each of which having 40 cores and 384 GB of random access memory (RAM).

Using 2,4,8,16 and 32 cores, simulations are performed in parallel with the quantity equal to the number of cores (set with NbJobs in Settings). Fig. 5.2 shows that *VirtualLab*'s pathos implementation gives excellent scaling, matching closely with a perfect linear scaling, which is referenced using a dashed line. The method used for parallelisation is defined using the Launcher argument in Settings. The slight deviation from perfect scaling is attributed to the overhead required to initiate the different processes.

The same analysis was conducted using pyina's MPI routine, which is shown in fig. 5.3. For simplicity, this analysis was performed using a single node. As shown, the scaling here is poor, with a large deviation from the perfect scenario. One potential cause for this is that the manager process (often termed the master) is also a worker, meaning that on top of distributing jobs to the other processes (termed as slaves) and checking on their status's, it also must complete a job itself. This is confirmed by looking at the *code_aster* log file for the first simulation (the one assigned to master), which shows that the time taken to perform the analysis is much higher compared with the others.

Fortunately, pyina has the capability of restricting the master process to ensure it isn't also a worker. This, however, means that an additional core would be needed, e.g. if one wanted to perform all 32 simulations concurrently then 33 cores would be required as one is used by the manager. The scaling using an independent master is shown in fig. 5.4, which shows much better levels of scaling compared with fig. 5.3, although the need for the additional core for an independent manager means that perfect scaling would be impossible. Unsurprisingly, there is a larger overhead required to set up the MPI processes compared with pathos' multiprocessing.

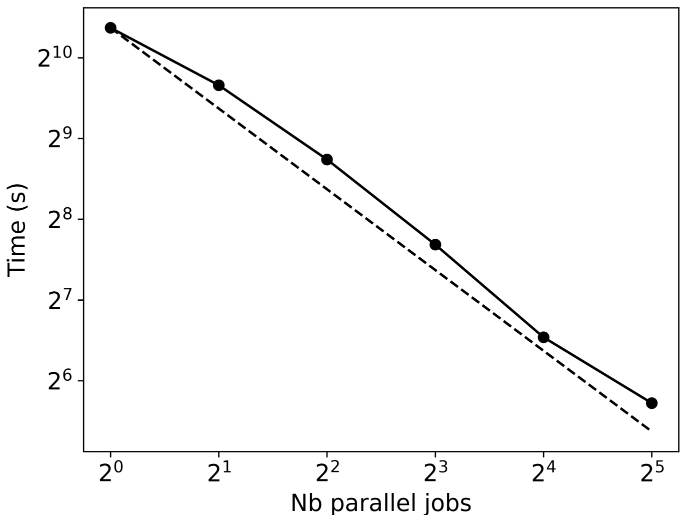


Figure 5.2: Comparison of *VirtualLab* scaling using pathos multiprocessing (-o-) with perfect scaling (–)

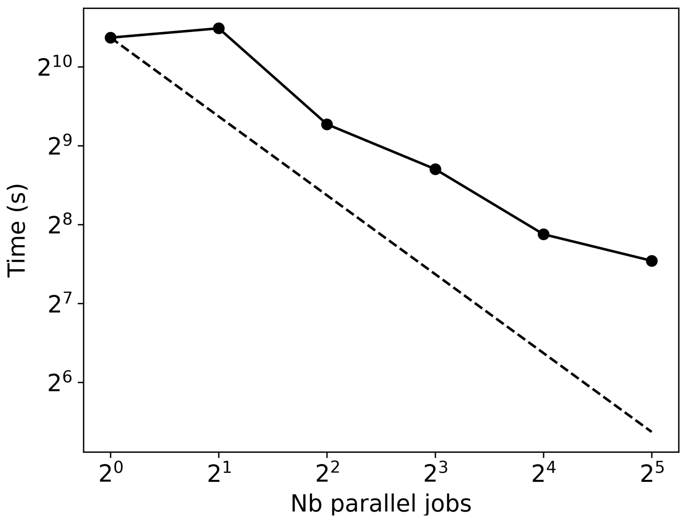


Figure 5.3: Comparison of *VirtualLab* scaling using pyina MPI (without independent master) (-o-) with perfect scaling (–)

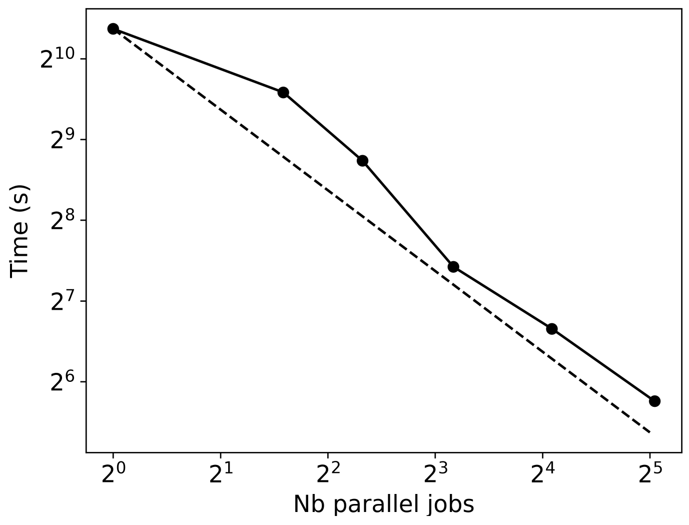


Figure 5.4: Comparison of *VirtualLab* scaling using pyina MPI (with independent master) (-o-) with perfect scaling (–)

5.3 HIVE Simulation

Modelling a multi-physics system such as HIVE is complex and requires the use of multiple modelling techniques. This section looks at the variety of modelling methods used and their integration with one another. Fig. 5.5 provides a visualisation of the boundary conditions in the HIVE simulation and their reference to the specific sections.

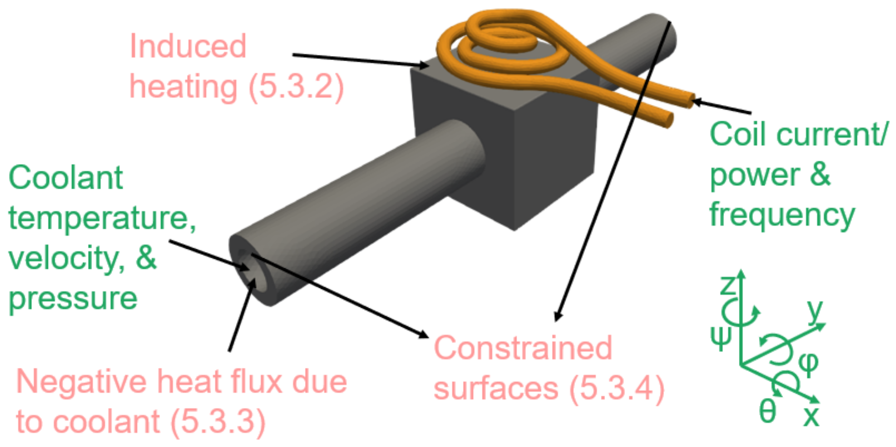

Induced heating (5.3.2)
Coil current/ power & frequency
Coolant temperature, velocity, & pressure
Constrained surfaces (5.3.4)
Negative heat flux due to coolant (5.3.3)
z
y
ψ
φ
θ
x

Figure 5.5: Visualisation of experimental parameters (green) and boundary conditions (red) for the HIVE simulation.

5.3.1 Component Geometry

The majority of components tested in HIVE are made up of three parts; a pipe, block, and tile (or armour). The work presented here and in chapters 6, 7 and 8 are performed on a titanium heat exchanger component designed as part of a project between Swansea University and UKAEA. An engineering drawing of the component is shown in fig. 5.6, with the respective dimensions given in table 5.1. The component mainly focused on in this is made of a single piece of titanium, meaning there are no joins between the pipe and the block, and the block and the tile. Chapter 8 will look at a component where the tile and block are two separate parts bonded together.

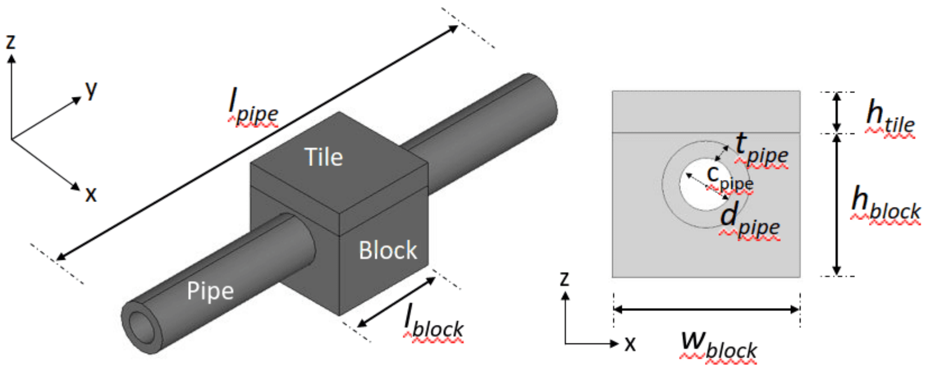


Figure 5.6: Engineering drawing of titanium component used in analysis.

Dimension	**Magnitude (mm)**
l_{block}	45
w_{block}	45
h_{block}	35
l_{tile}	45
w_{tile}	45
h_{tile}	10
l_{pipe}	200
d_{pipe}	12.7
t_{pipe}	4.15

Table 5.1: Dimensions of titanium component.

5.3.2 Induction Heating Model

Michael Faraday first demonstrated EM induction in 1831, with James Maxwell formulating this idea into the Maxwell-Faraday equation, see equation 5.1 [185]. This combined with the Gauss's law for electricity (eq. 5.2), Gauss's law for magnetism (eq. 5.3) and Ampere's law (eq. 5.4) complete the governing set of equations which describe modern EM theory, known as the Maxwell equations [186]. In these equations **E** is the electric field, **B** is the magnetic field, **J** is the electric current density, ρ_c is the electric charge density, ε_0 is the permittivity of free space and μ_0 is the permeability of free space.

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{5.1}$$

$$\nabla \cdot \mathbf{E} = \frac{\rho_c}{\varepsilon_0} \tag{5.2}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{5.3}$$

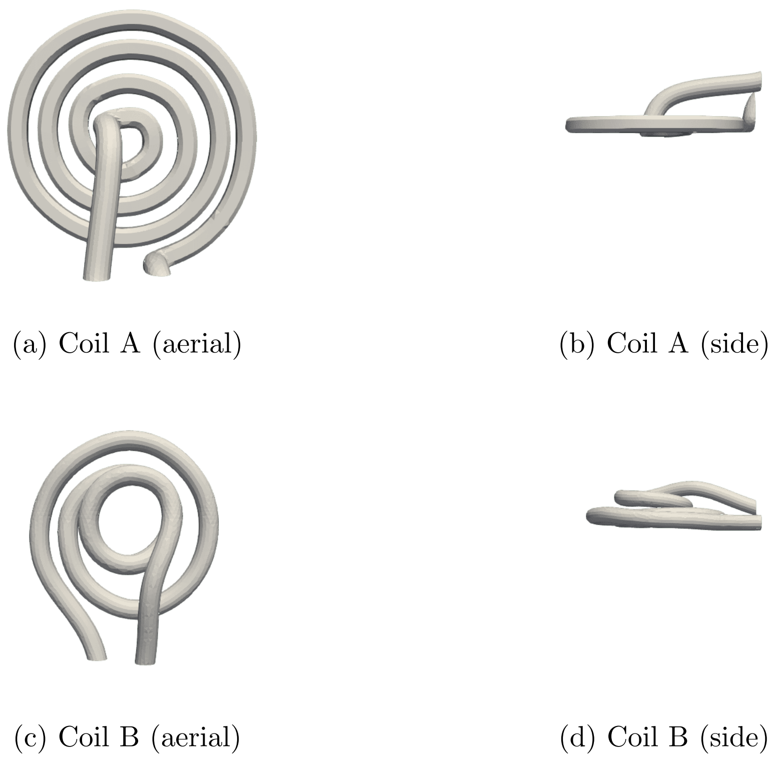
$$\nabla \times \mathbf{B} = \mu_0(\mathbf{J} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}) \tag{5.4}$$

In *ERMES* the FE method is applied to the 'double-curl' Maxwell equations, which is derived by taking the curl of equation 5.4 and 5.1 . Unlike many EM FE solvers, *ERMES* results are calculated at the nodes instead of the commonly used edge elements, because these can produce singularities due to ill-conditioned matrices [167].

Performing analysis of HIVE's induction heating system using *ERMES* firstly requires a mesh of the component, the induction coil and the surrounding vacuum. The design of the induction coil and positioning has a huge effect on the coupling efficiency, and thus heating delivered to the component. This work will perform analysis using two different coil designs, referred to as coil A and coil B, which are shown in fig. 5.7.

The positioning of the coil relative to the component can be specified using 6 degrees of freedom; 3 translational and 3 rotational, see fig. 5.5. Once the coil design and positioning are specified, a conformal mesh of the vacuum which surrounds both the coil and component is generated using *SALOME*.

Next, information relating to the analysis is required. BCs, such as parts of the geometry which are treated as perfect electrical conductors, must be assigned, along with the material properties of the component and induction coil. Along



(a) Coil A (aerial) (b) Coil A (side)

(c) Coil B (aerial) (d) Coil B (side)

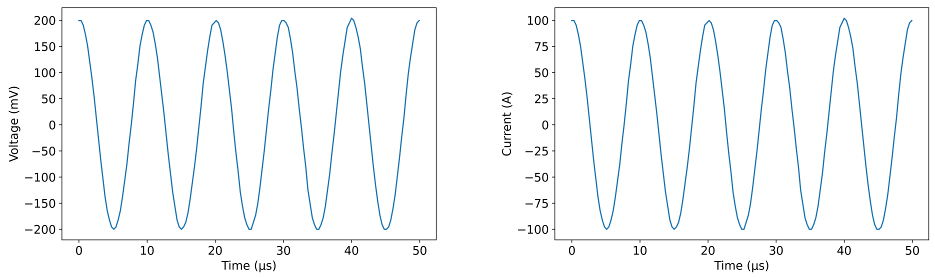
Figure 5.7: Coil designs considered for HIVE analysis.

with this, the material properties of the component are required, with EM analysis requiring values for the ρ, μ_r and electric permittivity, ε_r of a material.

The frequency and magnitude of the current in the coil is also required, however as discussed in section 4.3, the induction heating system used at UKAEA only measures the power it draws from the power supply, not the magnitude of the current. To accurately measure the magnitude of the current, the CWT Rogowski coil and RS PRO 2205A oscilloscope have been incorporated into the HIVE facility, enabling the values of the current and frequency to be measured near the induction coil terminal [187], [188].

A Rogowski coil is a wound coil which encloses a current carrying wire, which in this case the induction coil input terminal. The alternating current in the input terminal creates an alternating magnetic field, which induces a voltage in the Rogowski coil. The output of the Rogowski coil is integrated using an oscilloscope to retrieve the sinusoidal profile of the voltage. Each oscilloscope has a sensitivity value which describes how the integrated voltage is proportional to the current in the enclosed wire. The oscilloscope used has a coil sensitivity of 2 mV/A.

An example of the information collected using the Rogowski coil is shown in fig. 5.8a, where the inputs to the induction heating system were for a frequency of 101.3 kHz and power of 1kW. Taking into account the sensitivity of the oscilloscope, a plot of the current is shown in fig. 5.8b. This shows that the magnitude of the current is around 100 A, and also confirms that the frequency demanded by the induction heating system is correct, as each period takes around $10\mu s$, as expected.



(a) Voltage (b) Current

Figure 5.8: AC profiles recorded using Rogowski coil and oscilloscope.

As the coils used by HIVE are asymmetric, the *ERMES* analysis is performed in two stages. First, an electrostatic analysis is performed to determine the field **J** in the induction coil, followed by a full wave analysis which uses this to calculate the fields **E**, **B** and **J** in the component.

The heating delivered to a component results from the Joule heating, the differential form of which is given by eq. 5.5. This is the quantity of power which is delivered to a component per unit volume. Because *ERMES* is a linear solver, the solution fields for **E**, **J** and **B** all scale linearly with the magnitude of the current. As Joule heating is the product of **E** and **J** this field is proportional to the square of the magnitude of the current in the coil.

$$\frac{dP}{dV} = \mathbf{E} \cdot \mathbf{J} (W/m^3) \tag{5.5}$$

As discussed in section 4.3, the skin depth for most components tested in HIVE is less than 0.5 mm, therefore it is essential that this region of the component be well discretised. Due to the need for a conformal mesh of the surrounding vacuum, a global refinement of the component's mesh will substantially increase the time taken to produce the result. This not only increases the time taken to perform the FE analysis due to the increased number of degrees of freedom of the model, but meshing the finer conformal mesh is also time-consuming.

Prior to refining the mesh, it is important to note that, when using a standard CAD geometry, singularities will arise at sharp corners using *ERMES*. This is a well known issue in FE analysis, where a certain value will diverge towards infinity at sharp corners as the mesh refines [189]. There are a handful of ways in which this can be dealt with, the most common of which is the inclusion of a fillet at the corner [190]. A fillet is parameterised by a radius which effectively smooths the corner, an example of which is shown in fig. 5.9 which has been added to the CAD of the tested component.

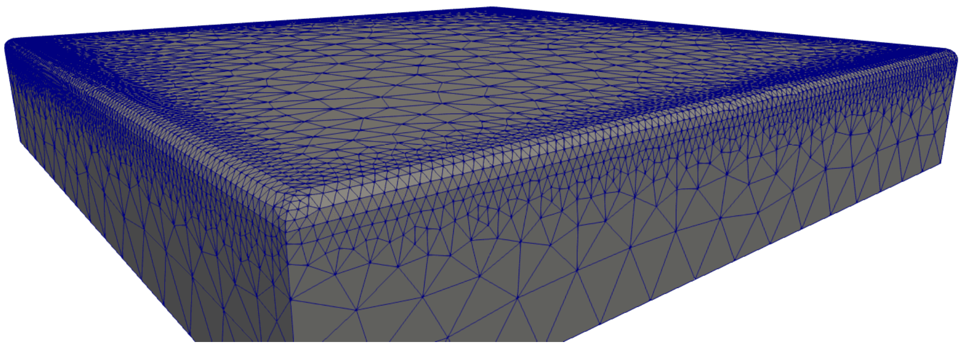
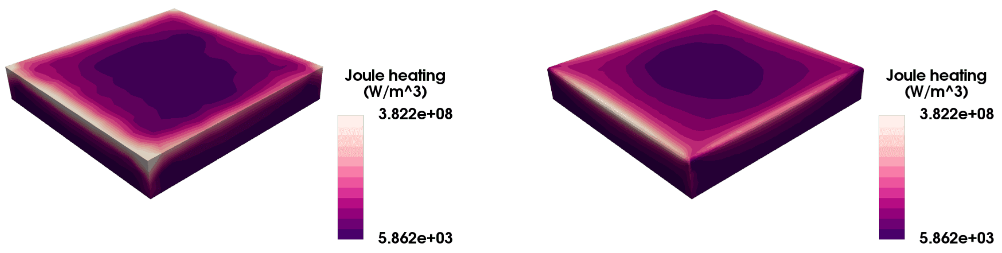


Figure 5.9: Mesh of fillet added to face adjacent to coil.

A comparison of the *ERMES* results generated with and without a fillet is shown in fig. 5.10. Note that this analysis has been performed at the lower frequency of 10 kHz in order that the magnitude of the skin depth is larger to highlight the difference.



(a) Sharp corner. (b) Filleted corner.

Figure 5.10: Reduction in peaks at corners by introducing fillet to coil adjacent surface.

Fig. 5.11 shows how the maximum Joule heating value reduces three-fold by removing the sharp corners for a variety of mesh sizes. The frequency for this and all remaining *ERMES* analyses are performed using the usual coil frequency of 101.3 kHz.

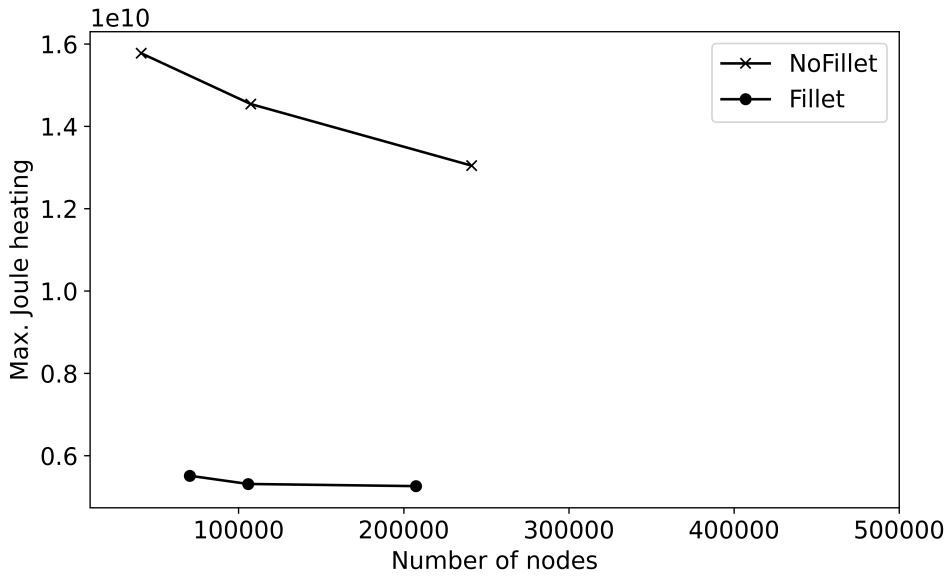
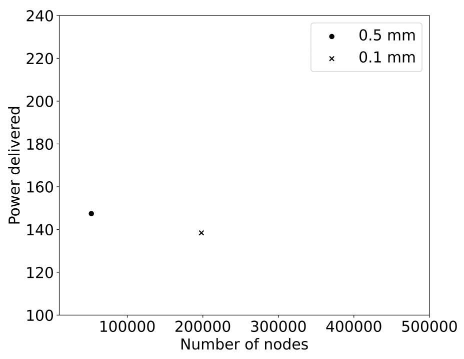


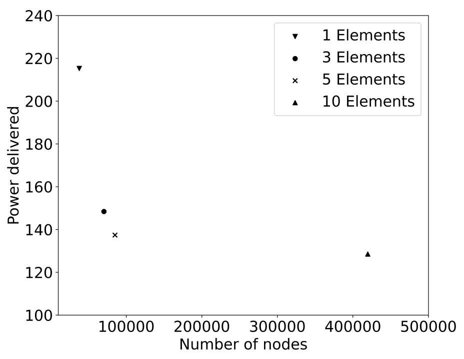
Figure 5.11: Maximum Joule heating value for fillet and non-fillet geometry versus mesh size.

As the inclusion of the fillet will inevitably lead to increasing the number of elements in the mesh, the first course of action is to investigate the mesh fidelity in this region. The power delivered to the component can be calculated by multiplying the volume of the elements by its corresponding Joule heating value and summing over all elements. As this is the key result extracted from *ERMES* analysis, this is what the mesh refinement will be judged against.

As the fillet is a non-physical feature, it is important that its inclusion does not affect the overall results. Due to the scale of the components considered, a maximum fillet radius of 0.5 mm is imposed, with *SALOME* enforcing a lower bound of 0.1 mm. The power delivered to the component using the minimum and maximum fillet sizes are shown in fig. 5.12a, highlighting that a smaller fillet size increases the number of nodes substantially for only a very marginal improvement in the result. Therefore, the fillet size of 0.5 mm is advised. The next consideration is the level of discretisation on the fillet. Fig. 5.12b shows the change in power when using 1, 3, 5 and 10 elements to discretise the fillet, with 5 providing the best compromise between accuracy and computational expense.



(a) Fillet size.



(b) Fillet discretisation.

Figure 5.12: Refinement of mesh at the fillet.

As the current carrying layers are only present in the 0.5 mm below the coil adjacent surface, it is sufficient to refine the mesh on this surface only. Smaller 2D elements here will ensure smaller 3D elements in the region directly beneath the surface, as required. Fig. 5.13 shows the results for three different mesh sizes on this surface; 0.5 mm, 0.3 mm and 0.2 mm, which equates to around 1, 2 and 3 elements through the anticipated skin depth region, respectively. These results indicate that a mesh size of 0.2 mm, yielding 2 elements in the skin depth, are required to model the current carrying region with sufficient accuracy.

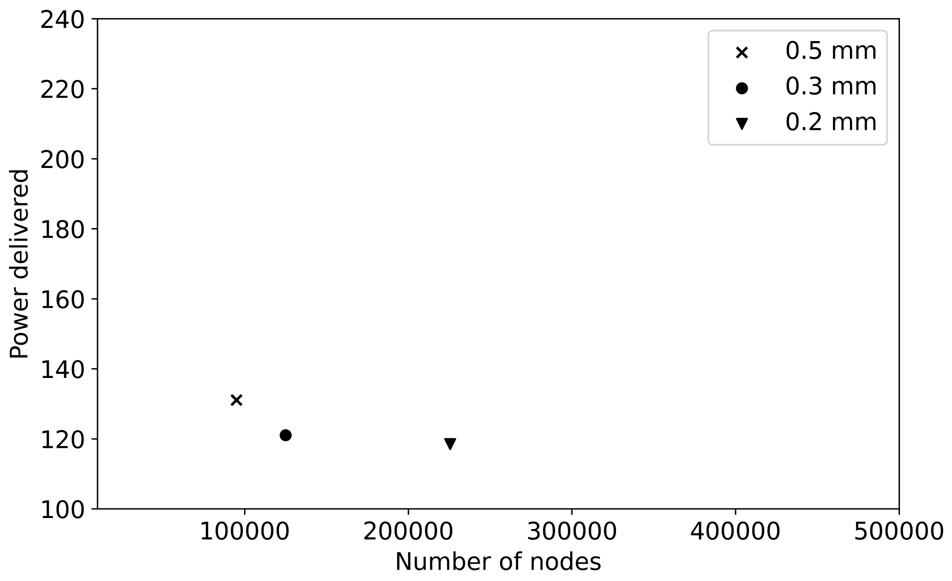


Figure 5.13: Mesh discretisation on surface adjacent to coil.

Combining each aspect of the mesh refinement results in a reduction in the mesh size by a factor of 3 compared with just applying a global refinement on the tile, as shown in fig. 5.14.

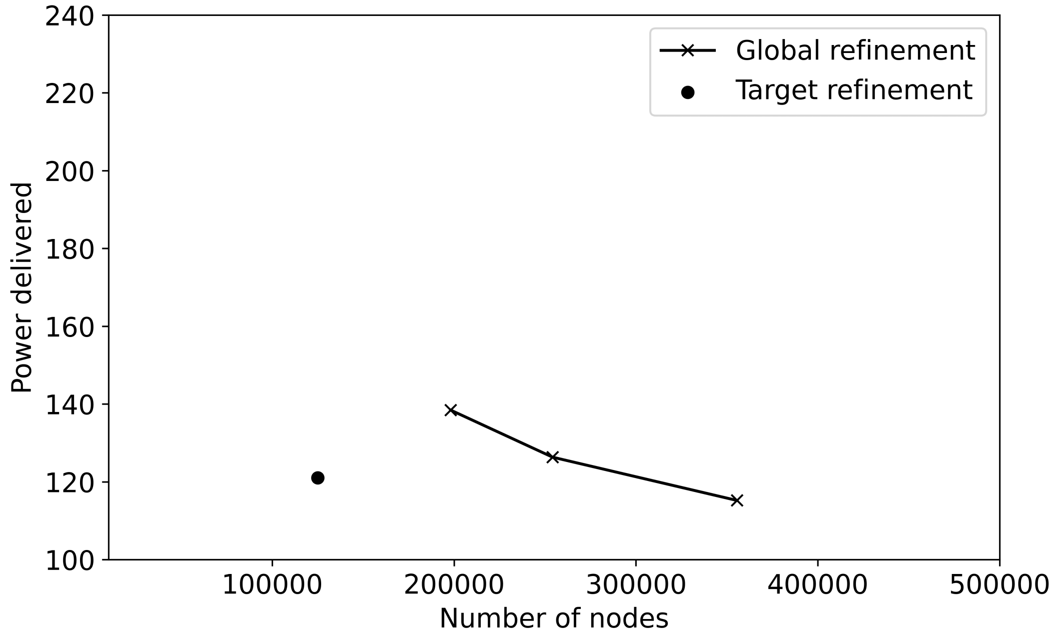


Figure 5.14: Reduction in mesh size using refinement algorithm compared with global tile refinement.

5.3.3 Coolant Model

The ability to model complex fluid flows has improved greatly with the advent of CFD. CFD's use is ubiquitous in a number of industries, including aerospace, automotive, marine and biology [191]–[194]. While it is an extremely useful tool to gauge key flow properties, it is computationally intensive, partly down to the mesh fineness required for accurate modelling of turbulent boundary layers.

Although the water travelling through the pipe in HIVE is a 3-dimensional, highly pressurised and turbulent flow, modelling its behaviour using CFD is unnecessary due to a large amount of research having been conducted in to the heat transfer properties of cooling pipes for a variety of different applications. These have shown that this problem can be simplified to a 1D problem defined using a boiling curve, which measures the heat flux, q, between the coolant and the pipe as a function of the wall (pipe) temperature, T_{wall}. An illustration of the boiling

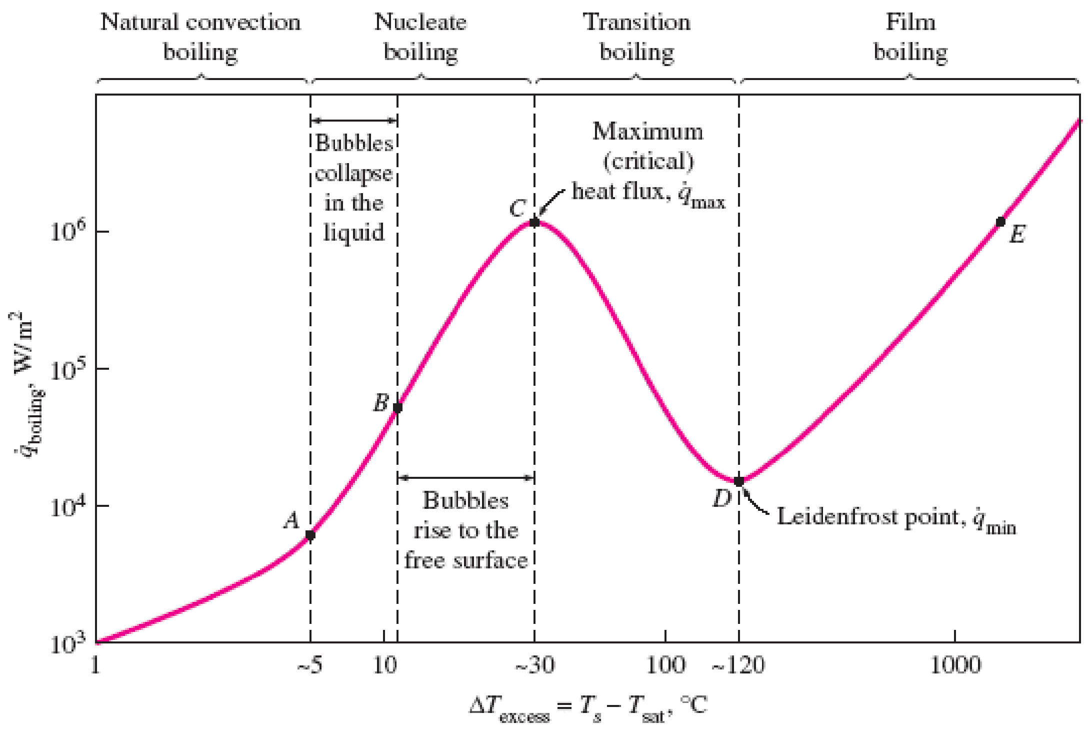


Figure 5.15: Boiling curve of water at 1 atm (0.1 MPa) [195].

curve of water at 1 atm is shown in fig. 5.15, where the different boiling regimes are highlighted.

Using experimental data, a number of correlations have been presented in literature to predict the quantity of this heat flux. These correlations generally depend on properties of the coolant, the geometry of the pipe and the wall temperature.

For the application of heat transfer, it is essential that the T_{wall} stays below the critical heat flux (CHF) temperature T_{CHF}. Above this temperature are the undesirable transition and film boiling regimes, which are characterised by the presence of large quantities of vapour between the wall and the fluid, thus dramatically reducing the heat transfer between the two. As a result, the correlations presented here are for the convection and nucleate boiling regimes of the boiling curve.

In order to model the coolant for different regimes requires a range of coolant properties, such as the saturation temperature, thermal conductivity and dynamic and kinematic viscosity of the fluid. The International Association for the Properties of Water and Steam (IAPWS) is an organisation established to provide internationally accepted formulation for the properties of water and light and heavy steam [196]. IAPWS97 [197] is a Python package which calculates key properties of a fluid for a given temperature, pressure and velocity, which are all known quantities set by HIVE operators.

When T_{wall} is lower than the saturation (boiling) temperature, T_{sat}, of the coolant, it is in the single phase regime. This can be broken down in to two distinct categories; natural convection, where any fluid motion is achieved by natural means (such as buoyancy effect), and forced convection, when the fluid motion is driven by external means such as a pump or fan, like it is in HIVE.

During the forced convection regime q is calculated by multiplying the heat transfer coefficient (HTC), h, by the difference between T_{wall} and the bulk temperature of the coolant, $T_{coolant}$, see eq. 5.6.

$$q = h(T_{wall} - T_{coolant}) \tag{5.6}$$

The HTC can be calculated using eq. 5.7, where Nu is the Nusselt number of the flow, k is the thermal conductivity of the fluid and l_{pipe} is the length of the pipe. The thermal conductivity of the fluid is available via the IAPWS97 package, while the pipe length for HIVE is a known quantity (l_{pipe} in table 5.1). The Nusselt number, which is a dimensionless number that measures the ratio of convective to conductive heat transfer, is unknown and must be estimated.

$$h = Nu\frac{k}{l_{pipe}} \tag{5.7}$$

A number of correlations are quoted in literature which estimate the value of the Nusselt number using other flow properties. In 1930 Dittus and Boelter proposed eq. 5.8, which is written in terms of the Reynolds number, $Re_{d_{pipe}}$, and the Prandtl number, Pr [198]. The Reynolds number is a commonly used dimensionless number which provides a ratio between the inertial and viscous forces within the fluid. The subscript to $Re_{d_{pipe}}$ signals that the diameter of the pipe, d_{pipe}, is used as the reference length. Similarly, the Prandtl number is dimensionless and measures the ratio of momentum diffusivity to thermal diffusivity. The range of conditions for which this correlation is valid are also included.

$$\begin{aligned} Nu =& 0.023 Re_{d_{pipe}}^{4/5} Pr^{2/5} \\ 0.6 \leq & Pr \leq 160, \\ & Re_{d_{pipe}} \geq 10,000, \\ & \frac{l_{pipe}}{d_{pipe}} \geq 10 \end{aligned} \tag{5.8}$$

When the difference between T_{wall} and $T_{coolant}$ is large, the Dittus-Boelter correlation shows poor results, with some stating that it is only valid for a difference of 6 °C [199]. Sieder and Tate accounted for this by including a viscosity correction factor, which is the ratio of the viscosity at the bulk temperatures, μ, and wall temperature, μ_{wall}, as shown in eq. 5.9 [200]. The other terms in this correlation are the same as that of Dittus-Boelter, but with minor changes to exponents and scaling factors. The inclusion of the viscosity correction factor also ensures that this correlation is applicable for a wider range of Prandtl numbers. It is advised that all properties of eq. 5.9, other than μ and μ_{wall} be evaluated at the film temperature, T_{film}, which is the average of T_{wall} and $T_{coolant}$[199].

$$\begin{aligned} Nu =& 0.027 Re_{d_{pipe}}^{4/5} Pr^{1/3} (\frac{\mu}{\mu_{wall}})^{0.14} \\ 0.7 \leq & Pr \leq 16700, \\ & Re_{d_{pipe}} \geq 10,000, \\ & \frac{l_{pipe}}{d_{pipe}} \geq 10 \end{aligned} \tag{5.9}$$

The predicted heat fluxes using the Dittus-Boelter and Sieder-Tate correlations are shown in fig. 5.16 for water at 0.5 MPa and 30 °C. As anticipated, the difference between the two becomes more evident at larger wall temperatures due to the viscosity correction factor included by Sieder-Tate. The heat flux transferred to the coolant increases for larger flow velocities, as expected.

The nucleate boiling regime is characterised by the fluid beginning to evaporate at the surface of the pipe. For small positive values of $\Delta T_{sat} = T_{wall} - T_{sat}$, the single phase correlations defined in eq. 5.8 and 5.9 are still valid. As this value increases, the fluid at the surface begins to evaporate, creating bubbles. This has the beneficial property of mixing the fluid, thus improving its heat transfer capabilities. Initially these bubbles will collapse after detaching from the surface, however as ΔT_{sat} increases through this regime the bubbles begin to coalesce into jet streams and in turn become slugs of vapour.

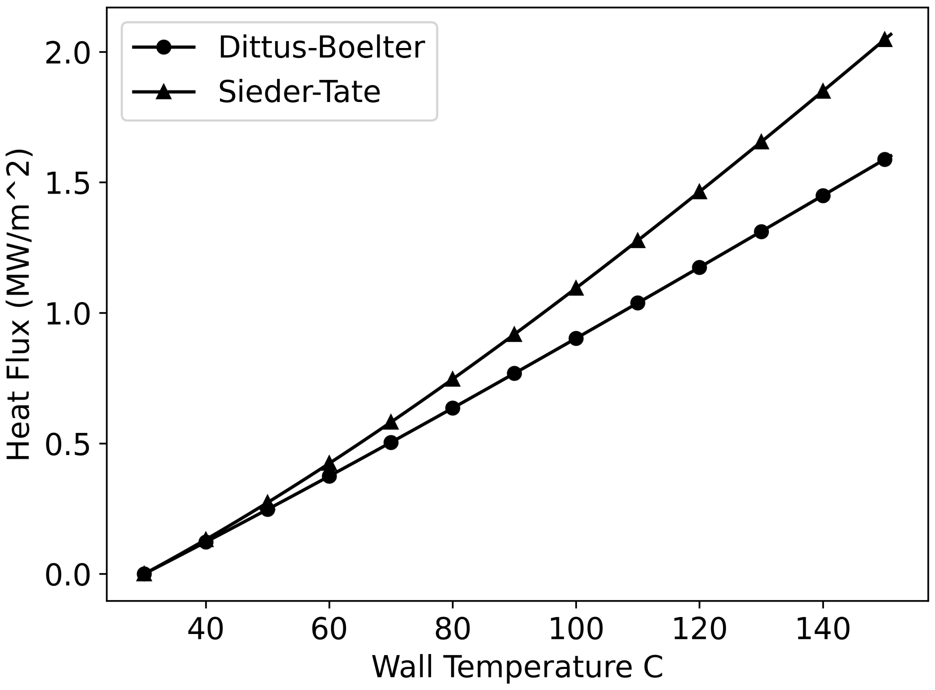


Figure 5.16: Forced convection heat flux calculated using the Dittus-Boelter and Sieder-Tate correlations for flow velocity of 3 m/s.

To accurately model the nucleate boiling regime, the conditions required to first achieve it must be deduced. This is referred to as the onset of nucleate boiling (ONB) and there are a number of correlations available in literature which estimate the heat flux required to achieve it. The most commonly used correlation is that of Bergles and Rohsenow, which is given by eq. 5.10, where $P_{coolant}$ is the coolant pressure and T_{ONB} the ONB temperature [201]. Note that their original formulation measured pressure in bar, however this formulation has been scaled by a factor of 10 so that it is measured in MPa.

$$\begin{aligned} q_{ONB} = 1082(10P_{coolant})^{1.156}(1.8(T_{wall} - T_{ONB}))^{\frac{2.16}{(10P_{coolant})^{0.0234}}} \\ 0.1 \leq P_{coolant} \leq 13.8 \end{aligned} \tag{5.10}$$

By equating the ONB correlation to the forced convection correlation, the wall temperature required to achieve this phenomenon is deduced. Above this temperature the flow is in the nucleate boiling regime, and must be modelled accordingly. A good review of the nucleate boiling correlations available in the literature can be found in [202].

For this work, the correlation developed by the Japan Atomic Energy Research Institute (JAERI) is used, which is given by eq. 5.11 [203]. Note that the factor of 10^6 converts q_{NB} to W/m^2C to match with the other correlations defined in this section. This correlation has been chosen because good agreement is shown against experimental results for a range of coolant properties similar to those available to HIVE [203], [204].

$$\begin{aligned} q_{NB} = 10^6 \exp^{\frac{P_{coolant}}{8.6}} &\left(\frac{T_{wall} - T_{ONB}}{25.72}\right)^3 \\ 30 \leq &T_{coolant} \leq 80, \\ 0.5 \leq &P_{coolant} \leq 1.6 \end{aligned} \tag{5.11}$$

To ensure a smooth transition between the forced convection and nucleate boiling regimes, an asymptotic model is used. These models contain heat fluxes from both forced convection, q_{FC}, and nucleate boiling, q_{NB}. The asymptotic model used in this work is that of Bergles-Rohsenow shown in eq. 5.12 [201]. By observation when $T_{wall} = T_{ONB}$, $q = q_{FC}$, and as the temperature increases q_{NB} dominates q_{FC}, meaning that $q \to q_{NB}$ as desired. The smooth transition between the two is shown in fig. 5.17.

$$q = q_{FC}\left(1 + \frac{q_{NB}}{q_{FC}}\left(1 - \frac{q_{ONB}}{q_{NB}}\right)^2\right)^{1/2} \tag{5.12}$$

The final consideration for the boiling curve is the estimation of the CHF, denoted by q_{CHF}. This value dictates the maximum amount of heat flux a cooling system can extract, which is achieved when $T_{wall} = T_{CHF}$. Due to the sudden drop in heat flux for temperatures above this value, ensuring the wall temperature is below T_{CHF} is a critical safety issue.

Because of the complexity of the physics which occur at the CHF, most correlations found in literature require a variety of coolant properties to predict q_{CHF}. Inasaka et Nairai modified the Tong 68 correlation to improve its accuracy and extend its application to a wider range of pressures, including those used by HIVE [205], [206]. The equation for this correlation is given in eq. 5.16, where x_{ex} is the vapour quality at the tube exit measured using the specific isobaric heat capacity, C_p, and the latent heat of evaporation, H_{fg}. The value for q_{CHF} is measured using the mass velocity, G, and the viscosity measured at the saturation temperature,

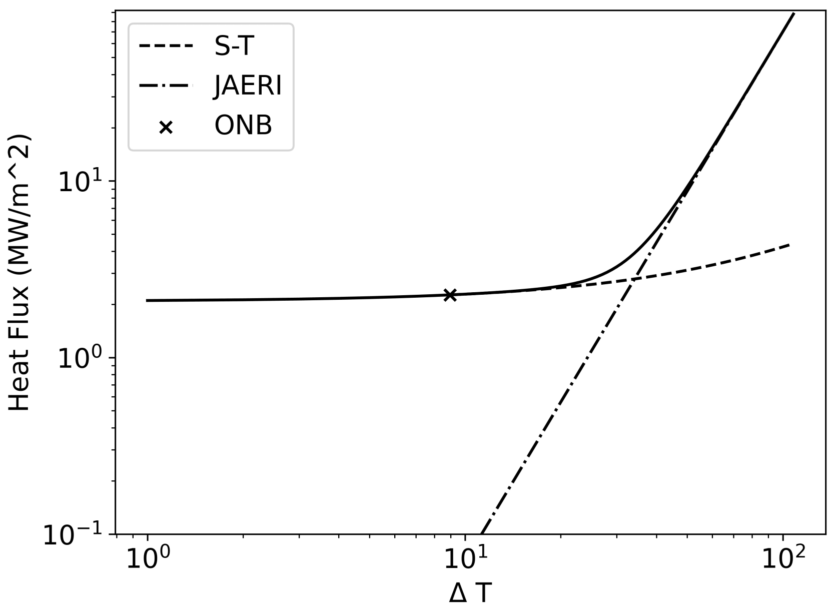


Figure 5.17: Asymptotic model of Bergles-Rohsenow for smooth transitioning between the forced convection and nucleate boiling regimes.

μ_{sat}. In the original Tong correlation $C = C_{Tong}$, whereas the modified Tong includes eq. 5.15.

$$x_{ex} = -\frac{C_p}{H_{fg}}(T_{sat} - T_{coolant}) \tag{5.13}$$

$$C_{Tong} = 1.76 - 7.433x_{ex} + 12.222x_{ex}^2 \tag{5.14}$$

$$C = C_{Tong}\left(1 - \frac{52.3 + 80x_{ex} - 50x_{ex}^2}{60.5 + (P_{coolant} \times 10^{-5})^{1.4}}\right) \tag{5.15}$$

$$q_{CHF} = H_{fg}\frac{CG^{0.4}\mu_{sat}^{0.6}}{d_{pipe}^{0.6}} \tag{5.16}$$

Combining this correlation with those calculated for the convection and nucleate boiling regimes results in a boiling curve of the expected heat flux for the range of admissible wall temperatures, which is shown in fig. 5.18.

Using these curves, it is possible to assess the effects that varying the three coolant parameters adjustable in HIVE will have. As anticipated, increasing the fluid velocity equates to a substantial increase in the heat flux absorbed by the coolant, see fig. 5.19. To extract $2MW/m^2$ of heat, the wall temperature for coolant pumped at 7 m/s is about half that of 1 m/s, however lower temperatures

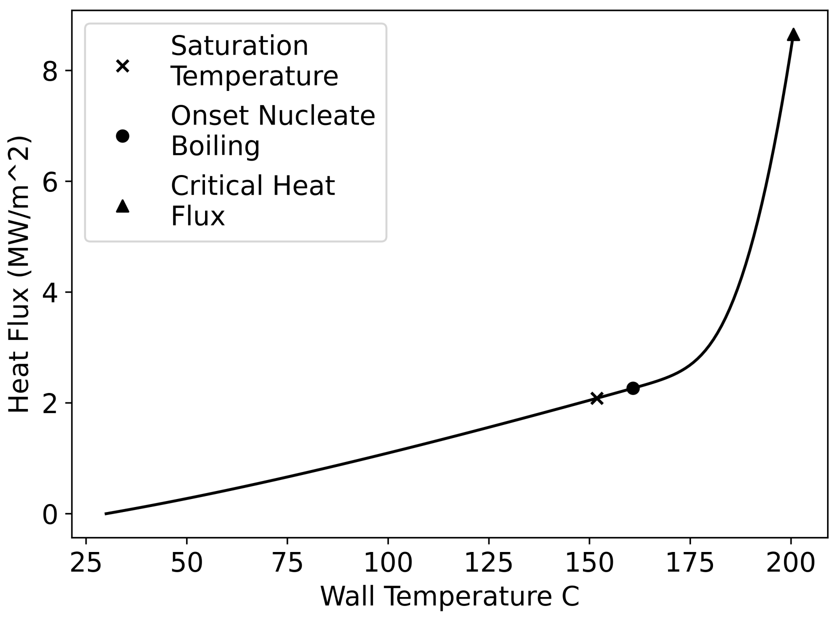


Figure 5.18: 1D boiling curve generated using forced convection and nucleate boiling correlations.

on the pipe may not be desirable as this creates a larger thermal gradient, which can lead to increased stresses.

As shown in fig. 5.20, varying the coolant pressure has no effect on the forced convection regime, however it does increase the values of T_{ONB} and T_{CHF}. This enables larger quantities of heat to be extracted by the coolant, which is desirable in cases where components are loaded with more extreme thermal loads. As expected, fig. 5.21 confirms that increasing the coolant temperature results in a reduction in the heat flux.

5.3.4 Thermo-mechanical Model

The thermo-mechanical model of HIVE is generated using *code_aster* and uses the results generated by the induction heating and coolant models. This work is interested in the steady-state behaviour of the components tested in HIVE, since these are the conditions they will be subjected to for long periods in a fusion device. As a result, this model ignores the transient nature of the experiment, reducing the computation time substantially.

Including the effects of the coolant is simple, as the boiling curve is applied to the surface of the pipe as a temperature dependent heat flux. The one simpli-

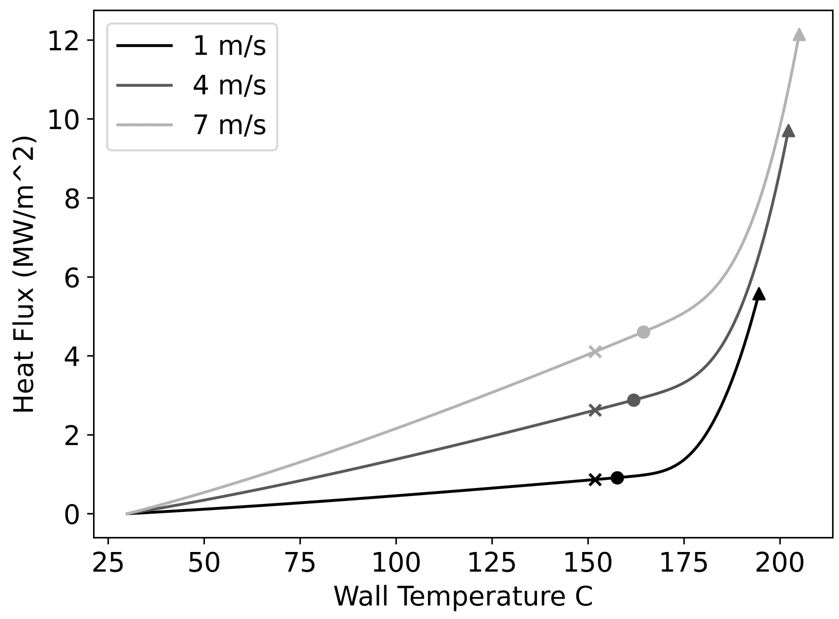


Figure 5.19: Variation in heat flux extracted by coolant pipe by varying the velocity.

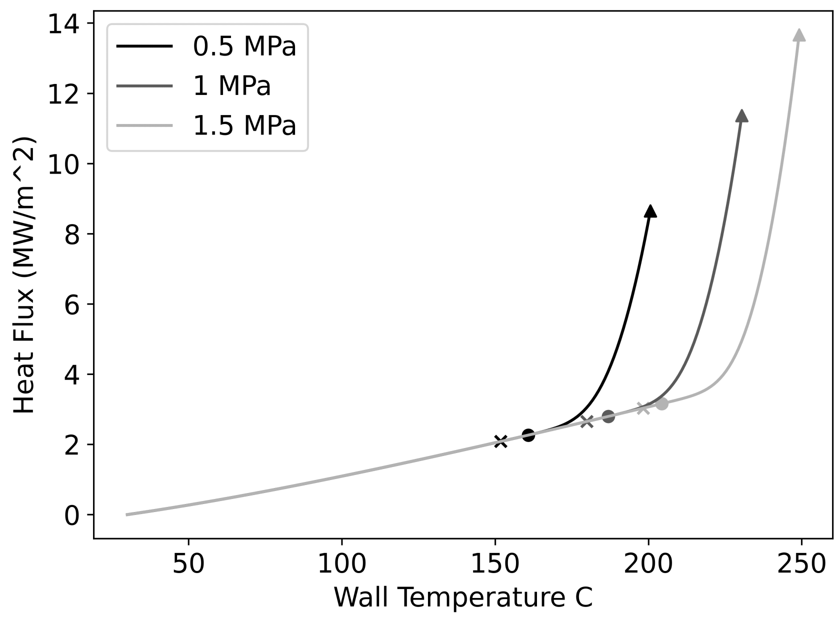


Figure 5.20: Variation in heat flux extracted by coolant pipe by varying the pressure.

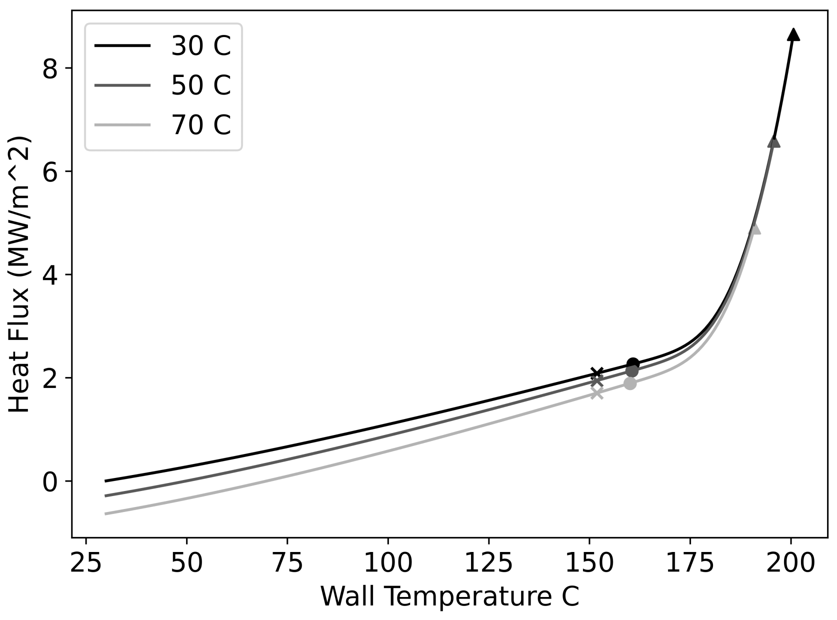


Figure 5.21: Variation in heat flux extracted by coolant pipe by varying the temperature.

fication made in this regard is that the temperature increase of the coolant along the pipe is ignored. That is, as the coolant travels along the pipe its temperature increases due to heat transferred from the component, however this change is negligible for a component with such a short path length and as such can be ignored.

Incorporating the Joule heating field produced by *ERMES* within *code_aster* is achieved by means of a heat source BC. Heat sources applied in *code_aster* can be temperature dependent, or temporally and spatially dependent. Due to their dependence on the geometry of the induction coil, the Joule heating profiles generated are irregular, meaning they cannot be represented as a function of the spatial coordinates x, y and z. This means that accurately applying the Joule heating loads can only be achieved by applying them to each element individually, which is extremely challenging.

Firstly, it requires that each volume element of the mesh be represented as an individual mesh group. Not only does this substantially increase the time taken to generate the component mesh and the *ERMES* mesh, but also rapidly increases the size of the file describing the mesh because of all the additional information regarding the groups. For example, even a very coarse mesh of the component has 144,855 volume elements, meaning that assigning each of these as an individual group would inflate the size of the mesh from 5.1 Mb to 130 Mb.

Secondly, *code_aster* is poorly suited for dealing with problems where a mesh has a huge number of groups. For example, it took more than 3.5 hours for *code_aster* to read in the aforementioned coarse mesh and over 4 hours to write the results to file. It also took over 4.5 hours to apply the Joule heating BC to each individual group. These time scales are more stark considering it takes only 10 s to solve the system of equations to calculate the temperature field (thermal analysis) and 20 s for the displacement, stress, and strain field (mechanical analysis). The likely cause of these time intensive operations is the use of inefficient search and sort algorithms to process the large quantities of data required for a problem such as this [207]. Note that the times quoted here used a single core on the Sunbird computer cluster.

Clearly, performing analysis in this manner is infeasible, therefore two different approaches were considered to address this issue; thresholding and clustering. The idea of thresholding is to only apply loads to the elements which have the greatest impact on the results. Given that the majority of the eddy currents are subjected to a very small region of the component, it is time-consuming and unnecessary to apply the Joule heating load to each of the 144,855 elements. Instead, the Joule heating loads are sorted in descending order, adding them until a certain percentage value of the total power is delivered.

By following the thresholding approach, the number of elements required can be reduced substantially. As is shown in fig. 5.22, to get 99% of the power 56,062 elements are needed, which is less than 40% of the 144,855 elements in the mesh. To ensure the anticipated power is delivered to the component, the values assigned to the 56,062 elements would be scaled by a factor of 1/0.99.

Unfortunately, the thresholding approach still yields large computation times, see table 5.2. While there is a reduction in the time compared with the full simulation, it is still high considering the size of the problem. It also leads to errors of up to 2% compared with the full simulation. Thresholding also has the undesirable feature of increasing the number of groups required with the size of the mesh, meaning it would be unusable for numerous cases.

The second approach avoids this issue by clustering similarly valued Joule heating values together in to a pre-defined, k number of groups using the k-means algorithm [208]. k-means is an iterative algorithm which aims to minimise the variance within each cluster. Each iteration consists of two steps; assignment and update. During the first step, each data point is assigned to the closest of

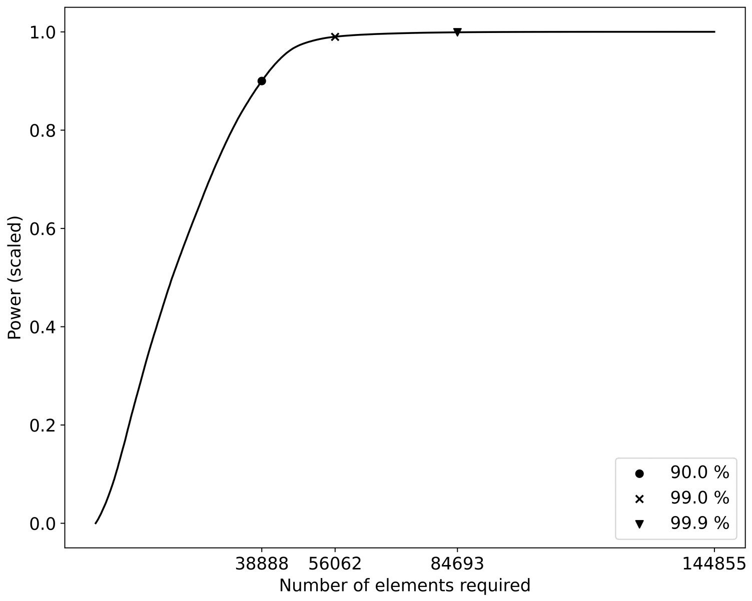


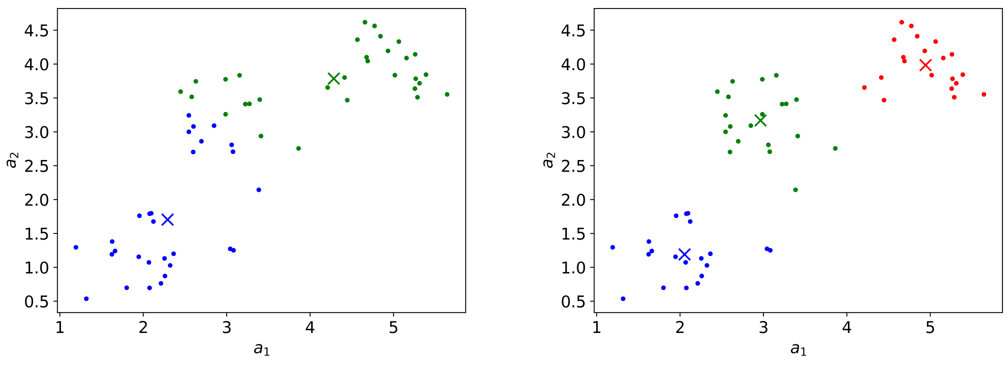
Figure 5.22: Power delivered to component using the most influential elements, with 90%, 99% and 99.9% thresholds highlighted.

the k cluster centroids, which are initially randomly chosen. The second step is then to update the cluster centroids, which is calculated as the average of all the points assigned to each cluster. This procedure is repeated until there is no further change to the value of the cluster centroids. An example of the use of the k-means algorithm for a 2D problem is shown in fig. 5.23 where 30 data points are clustered in to 2 and 3 groups.

How well the cluster centroids represent the original data is measured using the inter-cluster variance, or inertia, as it is commonly known. The formula for

	Nb. groups	Read mesh (s)	Setting BCs (s)	Writing results (s)	Total sim. time (s)	Max. err (%)
Full	144855	13133	17034	15657	45876	N/A
Thresh. (99%)	56063	1521	2035	1541	5135	1.17
Cluster	100	< 1	< 1	< 1	34	0.83
Cluster	1000	1	2	1	37	0.019
Cluster	10000	51	74	57	215	0.0018

Table 5.2: Comparison of thresholding and clustering for reducing simulation time.



(a) 2 clusters (b) 3 clusters

Figure 5.23: 30 2D points (o) assigned to cluster centres (x) using the k-means algorithm.

the inertia, I, of a dataset $X = \{x_1, ..., x_n\}$ clustered in to k groups is given in eq. 5.17, where c_j is the centroid of cluster j which has n_j points assigned to it. The i-th point of this cluster is denoted by x_{ji}.

$$I = \sum_{j=1}^{k} \sum_{i=1}^{n_j} |x_{ji} - c_j|^2 \tag{5.17}$$

There is a clear trade-off between reducing the inertia and increasing the number of clusters used. One method to choose the number of clusters is the 'elbow' method, which involves plotting the inertia scores and choosing the point at which the score levels out. Another, less interactive method is by using the goodness of variance fit (GVF) score, which scales the inertia by the sum of squares of the dataset, as shown in eq. 5.18. This metric is used in the Jenks natural breaks method, which pre-dates the k-means algorithm and is simply its 1D equivalent [209].

$$GVF = 1 - \frac{I}{\sum_{i=1}^{n} \left|x_i - \overline{X}\right|^2} \tag{5.18}$$

When the number of clusters is equal to n then each point will have its own, individual cluster whose centroid coincides with that point, resulting in an inertia value of 0 and a GFV of 1. On the other hand, the inertia attains its maxima when there is a single cluster, and is equal to the sum of squares. Since this is the denominator of the GVF score, this imposes a lower bound of 0 on its value.

The GVF value for the examples shown in fig. 5.23a and 5.23b are 0.3227 and

0.8043, respectively. Increasing the number of clusters to 4 (not pictured) will only increase the GFV to 0.8321. While visually it is clear that the data is best clustered in to 3 groups, the GVF score confirms this.

For HIVE, simulations were performed where the 144,855 Joule heating values were clustered in to 100, 1,000, and 10,000 clusters. Each of these 3 have GVF scores of greater than 0.9999, highlighting an excellent fit. The results for the clustering analysis are also shown in table 5.2. Using as little as 100 clusters yields more accurate results than the 99% thresholding along with orders of magnitude speed up in the computational time. Increasing to 1,000 clusters results in an order of magnitude improvement in the accuracy for almost no time penalty. Using 10,000 groups improves the accuracy slightly but increases the computation time exponentially. This increase is likely due to *code_aster* reaching a limit in terms of its memory allocation, with the need to load and unload data frequently.

Clearly, clustering is the better approach for incorporating the *ERMES* loading condition in to *code_aster*. All further analysis carried out in this work clusters the Joule heating loads in to 1,000 groups, which is performed using the SciKit Python package due to its improved scalability compared with PyTorch's implementation [210].

For the mechanical model, the component is constrained at the inlet and outlet of the pipe using the 3-2-1 method to avoid over constraining the part [211]. The displacement, stress, and strain fields of the component result from the thermal expansion of the component due to the temperature field. The effect of the coolant pressure on the pipe is ignored, which is a reasonable assumption considering the comparably low pressures.

In this workflow, the three FE simulations are weakly coupled, since they are solved independently of one another. For example, the *ERMES* simulation is performed without any knowledge of the temperature field, meaning that values for the temperature dependent material properties are taken for a generic reference value. Ideally, these would be strongly coupled, meaning that changes in the material properties arising from the temperature field and stress state would be reflected in the EM results, and vice versa.

There are a couple of ways in which this approximation could be improved, such as an iterative loop which passes the results back and forth until there is no further change, or performing a number of *ERMES* simulations for different temperature reference points, enabling the Joule heating loads to be applied as temperature dependent heat sources. For this work, however, the current implementation of a single *ERMES* simulation feeding in to a single *code_aster* simulation to calculate the temperature field and resulting mechanical field is deemed sufficiently accurate.

The procedure for the HIVE simulation is shown in fig. 5.24 as part of the wider workflow for surrogate model generation.

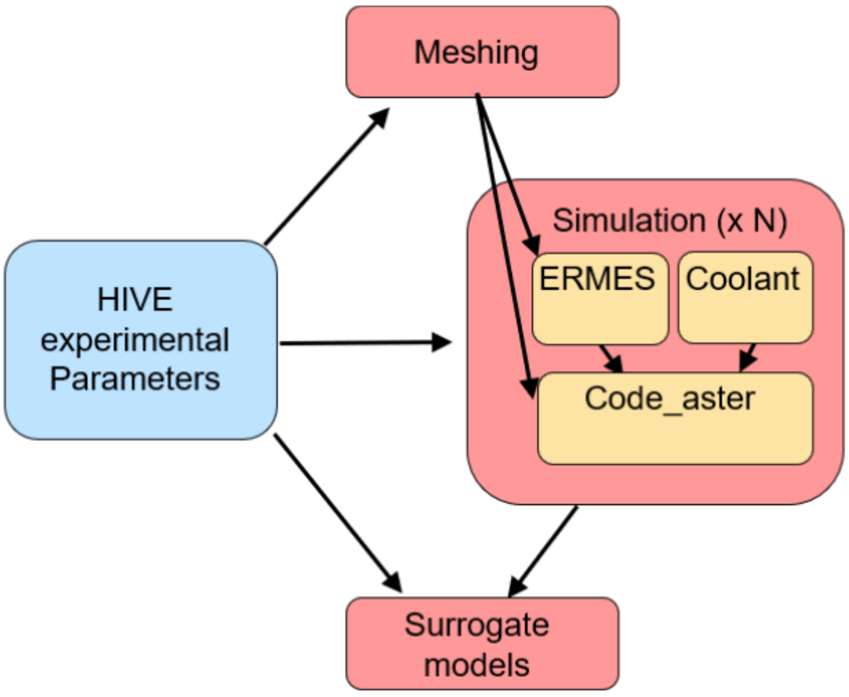


Figure 5.24: Workflow for virtual HIVE experiment and surrogate model generation.

5.3.5 Model Validation

In October 2022, 12 different heat exchanger components were tested in HIVE to assess the impact of known and unknown defects. Externally, each looked identical to the component shown in fig. 5.6, however many had voids machined in to the tile part of the component to assess their impact. Presented here are the results for the 'baseline' component, which is the solid titanium component with no voids and no bonds between the pipe, block, and tile. These experiments were carried out using coil A (fig. 5.7a for reference).

Unfortunately, it proved to be infeasible to bond the thermocouples to the component's external surface, therefore the thermocouple data was unavailable for validation. Instead, the terminals of the coil were shifted so that a side on view of the component was visible to the IR camera. For reference, fig. 5.25 shows the tested component through one of the other portholes of the vacuum vessel. Along with this, a small thermal mirror was placed behind the component so that the IR camera would record some data on the opposite, back face in addition to the front face.

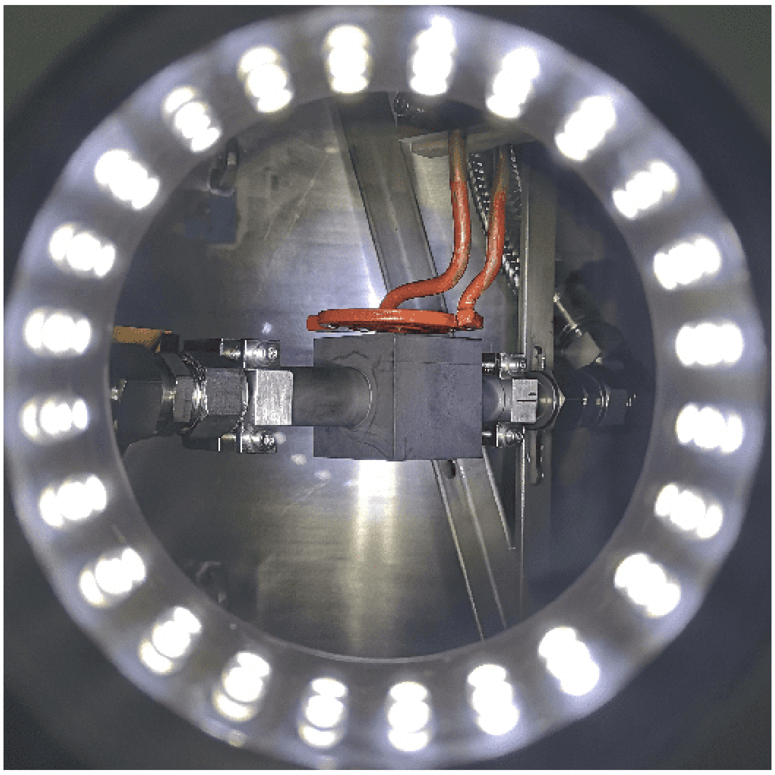


Figure 5.25: View of titanium component through a porthole

Using the IR software, regions of interest (ROI) on the front and back face are established, which are shown in fig. 5.26. Over these ROIs, the maximum temperature and average temperature are recorded over time. The ROI on the front face covers the entire face, making it simple to compare against the simulation results. The back facing ROI is slightly more ambiguous, especially in regard to the average, as it only covers a nondescript part of the back side. As a result, the accuracy of the model will be compared with the average and maximum temperature data recorded on the front face, and the maximum temperature data on the back side once steady state has been achieved.

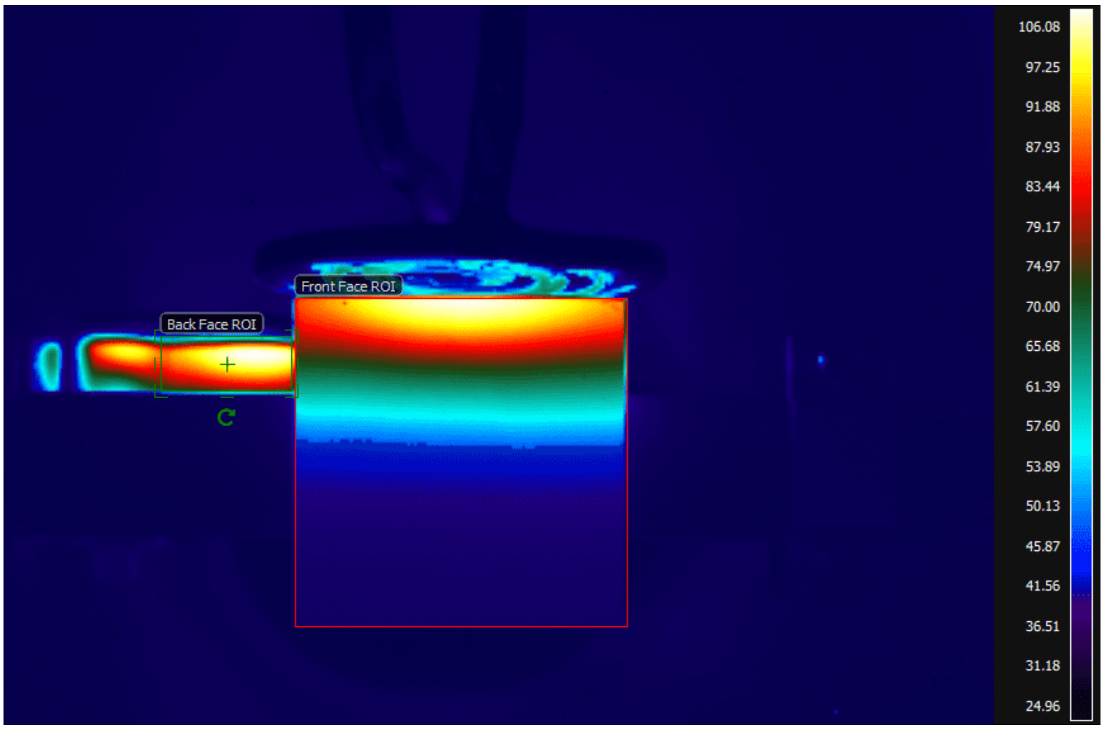


Figure 5.26: IR camera and the ROIs to record data.

A matrix of loading conditions were set up to validate the model. These include varying the coolant temperature from low (25 °C) to high (75 °C), and the induction heating power from 1 kW to 1.6 kW. Table 5.3 defines the parameters which vary over the four experiments, along with the coil frequency and current data recorded using the Rogowski coil.

Name	Coolant temp. (°C)	Induction power (kW)	Coil freq. (kHz)	Coil current (A)
Exp. 1	25	1.0	100.215	103.93
Exp. 2	25	1.6	100.23	175.73
Exp. 3	75	1.0	100.23	104.23
Exp. 4	75	1.6	100.22	171.6

Table 5.3: Parameters varied for the HIVE experiments.

The data collected from the 4 experiments are shown along with the equivalent simulation data and the percentage error in table 5.4. There is excellent agreement between the two, especially on the two metrics extracted from the more well-defined front face ROI. The errors are slightly larger regarding the maximum temperature on the back face ROI, however one reason for this could be that the true max. temperature isn't actually being recorded in this ROI.

Name	Front max.			Front avg.			Back max.		
	Exp. (°C)	Sim. (°C)	Err (%)	Exp. (°C)	Sim. (°C)	Err (%)	Exp. (°C)	Sim. (°C)	Err (%)
Exp. 1	115	115.32	0.28	54.9	56.54	2.99	120	125.35	4.46
Exp. 2	250	262.1	4.84	105	111.03	5.74	261	285.84	9.52
Exp. 3	154	159.43	3.53	99.3	103.67	4.4	159	168.45	6.26
Exp. 4	273	286.7	5.02	141	150.32	6.61	283	308.38	8.97

Table 5.4: Comparison of experimental and simulation results recorded over the two ROIs.

5.4 Chapter Summary

This chapter has presented the computational model of HIVE, which will be used to generate synthetic data for the construction of surrogate models in the following three chapters. This chapter discussed the solutions identified to certain challenges which were encountered during the development of this model. These included developing a tool which enabled *ERMES* to be used without relying on a commercial pre- and post-processing package, and identifying a way to more efficiently apply Joule heating loads to the thermo-mechanical model, which resulted in orders of magnitude speed up for almost no loss of accuracy. The developed model showed good accuracy compared with experimental results, with a maximum error of less than 10%.

The main novelty of this chapter is the *VirtualLab* package which has been presented. This package provides the framework to create fully automated workflows for computational engineering applications, and enables a variety of different software packages to be easily incorporated through the use of containers. Containers also enable the *VirtualLab* python package to be easily portable from one system to another, while also improving the reproducibility of data generated by the workflow. The techniques used to achieve high throughput parallelisation were also presented, highlighting excellent scaling on both single-node and multi-node problems.

Chapter 6

Optimisation of Hardware Configuration

6.1 Introductory Remarks

The goal of this chapter is to showcase the ease in which *VirtualLab* can generate and analyse data, providing insight relating to the HIVE facility without the need for prior knowledge.

There are a number of desirable ways in which a component in HIVE can be tested, with some of the most popular requests outlined in section 4.4. This chapter will focus on one of these; the uniformity in which heat loads are applied to a component. Often, however, the goal is not simply just to maximise the uniformity of the heating profile, but instead to do this subject to the fact that a certain power is also delivered to the component (a constrained optimisation problem). This example has been chosen first, as the heating profile subjected to the component only depends on a subset of the experimental parameters, specifically those relating to the design and positioning of the induction coil. As a result, the surrogate models presented here will have fewer inputs, making them faster to train, and making their use in optimisation algorithms easier to comprehend.

Firstly, section 6.2 presents the theory of how a uniform profile is defined for this work. Following this, the accuracy of both GPR and MLP for the generation of SV surrogate models is discussed in section 6.3. Here, a comparison of different sampling strategies will also be presented. Section 6.4 presents the use of both GPR and MLP for the construction of 2D FF surrogate models of the coil adjacent surface, where the accuracy of the PCA and autoencoders dimensionality reduction techniques are analysed. The performance of MLP algorithms without the use of dimensionality reduction is also presented. Section 6.5 then presents results for the inversely informed coil configuration and the possible insight this methodology can provide.

6.2 Profile Variation

In order to optimise the coil configuration with respect to the uniformity of the heating profile, first a method of quantifying the uniformity, or conversely variation, in the heating profile is required. As the induction heating is essentially a heat flux, it is only necessary to consider the profile on the coil adjacent surface.

One method of measuring the variation in this profile is the standard deviation of the Joule heating values. This, however, was found to be problematic as this quantity varies drastically with changes to the mesh refinement. Instead, a method was devised which looked at the gradient for each surface element when taking into account the Joule heating values. The Joule heating is added to the coordinates of an element in the direction of the unit normal to the surface, $\bar{\mathbf{n_o}}$, giving a new, augmented triangle. Taking the cross product of two vectors which make up any of the 3 edges of the triangle returns the vector $\mathbf{n_a}$, whose direction is normal to the surface and magnitude is proportional to the area of the triangle. The variation of that element is calculated as the magnitude of the cross product of $\mathbf{n_a}$ and $\bar{\mathbf{n_o}}$, as shown in eq. 6.1. This value is calculated for all elements which make up the coil adjacent surface and summed to give an overall value of the variation across the surface.

$$Variation = \mid \bar{\mathbf{n_o}} \times \mathbf{n_a} \mid \tag{6.1}$$

For example, consider a triangle with vertices at $(0, 0, 0)$, $(1, 0, 0)$ and $(0, 1, 0)$ whose unit normal is $\bar{\mathbf{n_o}} = (0, 0, 1)$. The Joule heating values for the three corners are 4, 2 and 6, respectively, giving the coordinates of the new triangle as $(0, 0, 4)$, $(1, 0, 2)$ and $(0, 1, 6)$. This augmented triangle element and the original are shown in fig. 6.1. The vectors $(1, 0, -2)$ and $(0, 1, 2)$ make up the edges from vertex 1 to 2 and 1 to 3 respectively. The cross product of these is $\mathbf{n_a} = (2, -2, 1)$, which when input in to eq. 6.1 with $\bar{\mathbf{n_o}}$ gives the variation for this element as $2\sqrt{2}$. If the Joule heating values were the same for the three corners of the triangle, then the variation is zero as $\mathbf{n_a}$ and $\bar{\mathbf{n_o}}$ are parallel.

An example of how the variation score varies for different profiles is shown in fig. 6.2 for a 1D case.

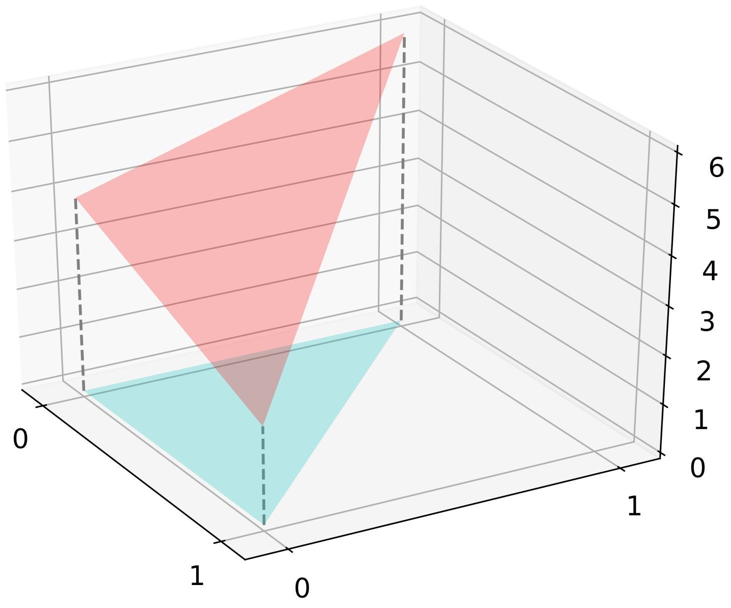
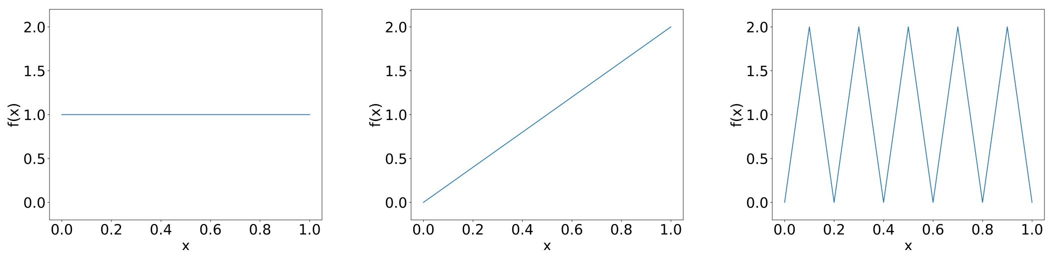


Figure 6.1: Original (blue) and augmented triangular element.



(a) Variation = 0 (b) Variation = 20 (c) Variation = 200

Figure 6.2: 1D representation of the change in variation score with different function profiles

6.3 SV Surrogate Models

As discussed in section 5.3.2, for each coil there are 6 degrees of freedom (DoFs) which specify its positioning; displacement in the x, y, and z direction, denoted by x_d, y_d and z_d, and rotation with respect to each axis, denoted by θ, ϕ and ψ, see fig. 5.5 for reference.

The effect of varying z_d is well understood in terms of the coils coupling efficiency (section 5.3.2). Moving the coil closer to the component will increase the power delivered, however it also results in a more non-uniform heating profile. The converse is true when moving the coil further away. The effect of varying the remaining DoFs, particularly when combined, is too complex for human intuition alone.

While ϕ will vary due to the tightening of the bolts which join the component to the coolant pipe, θ and ψ vary only slightly, therefore for simplicity these values are assumed to be zero for this investigation. This reduces the problem down to 4 DoFs, with an additional discrete DoF dependent on the coil design chosen.

Therefore, the surrogate models required for this section will have 4 inputs (the DoFs) and 2 outputs (the power and variation). As this problem relates to the Joule heating profile imparted on to the component, only data from the *ERMES* simulation is required. Calculating the power delivered to the component from this is simple and was outlined in section 5.3.2, while the variation is calculated using eq. 6.1.

To create surrogate models, data from across the 4D parameter space must be generated. The bounds for the four dimensions of the parameter space are shown in table 6.1, where the translational DoFs are measured in metres and rotational DoF in degrees. These values ensure a variable behaviour across the parameter space whilst ensuring that no combination would be infeasible, e.g. the coil touching a component. The reference point where (x_d,y_d,z_d) = (0,0,0) is shown in fig. 6.3.

DoF	**Lower bound**	**Upper bound**
x_d	-5E-03	5E-03
y_d	-15E-03	15E-03
z_d	3E-03	6E-03
ϕ	-5E+00	5E+00

Table 6.1: Parameter space considered for analysis.

2,000 points were chosen from across the parameters space using the Halton sequence, which were used to generate the training data, with a further 500 points chosen using random sampling for the test data. This data was collected using *VirtualLab* deployed on the Sunbird supercomputer, where 120 cores (3 nodes) were used, taking 324 minutes in total.

The results for coil A are presented here, however surrogate models for coil B have also been generated and will be used for the analysis in section 6.5. The performance of the surrogate models in this and the following two chapters are validated against the synthetic data collected using the computational model.

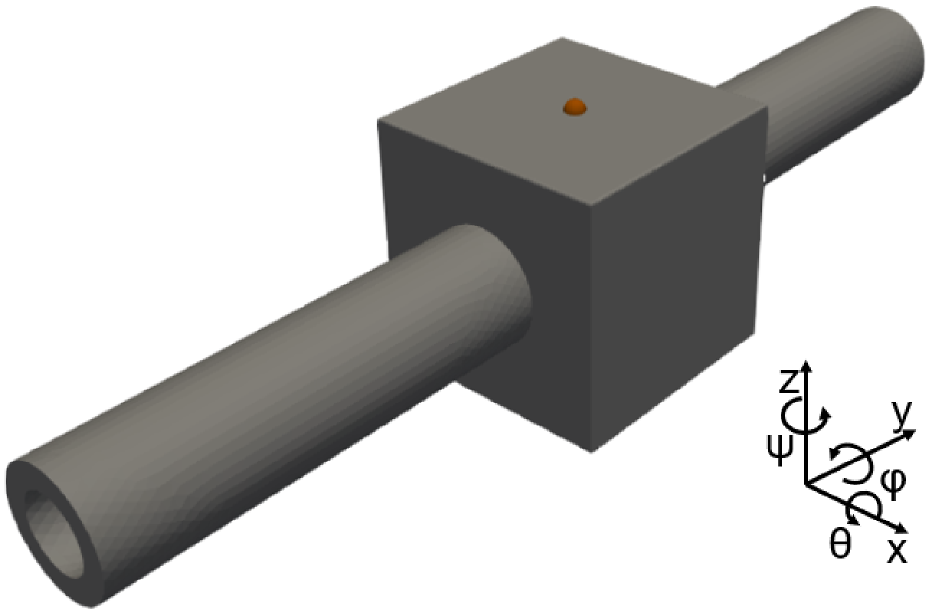


Figure 6.3: Reference point for where (x_d,y_d,z_d) = (0,0,0) along with frame of reference.

6.3.1 MLP

The most challenging aspect of using an MLP, or indeed any ANN algorithm, is deciding its architecture. There is no currently agreed upon best practice for the number of layers or number of neurons one should include in a model. Many researches use previously successful architectures with little scientific basis other than intuition. Some advise increasing and then decreasing the number of neurons as the layers move from input to output, while others advise keeping the number of neurons fixed [212]. It is often seen, however, that the number of nodes in a layer is of base 2, which helps speed up the parallelisation of the computations [213].

Fortunately, *VirtualLab* makes it easy to train multiple architectures in parallel to more quickly assess their performance. The various architectures are defined using the parameters file, allowing models to be created in an automated way. The architectures considered for this work, made up of 2 and 3 hidden layers, and are shown in table 6.2.

	Architecture	Nb. parameters
MLP_1	4-32-8-2	442
MLP_2	4-32-32-2	1282
MLP_3	4-64-16-2	1394
MLP_4	4-8-16-8-2	338
MLP_5	4-16-64-16-2	2242
MLP_6	4-32-32-32-2	2338
MLP_7	4-32-128-16-2	6482

Table 6.2: MLP architectures

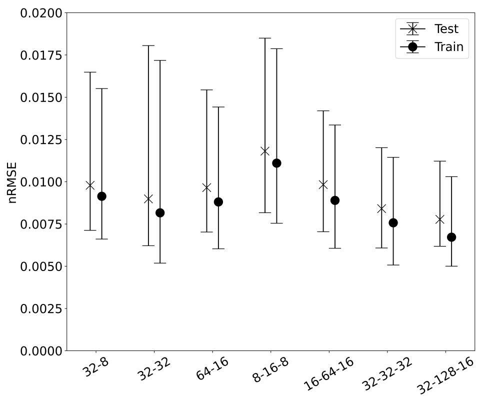
The activation function chosen for these models is swish (eq. 3.10), which is chosen due to its continuous differentiability and comparable performance with the commonly used leaky ReLU [214]. This model was trained for a maximum of 1,000 epochs, with training terminated if the model is deemed to be overfitting (see section 3.2.1). An epoch is a common ML term for the number of times the training data is iterated over during training. Along with this, a common batch size of 32 was used. All relevant hyperparameters used to train the models are given in table 6.3.

Training size	2,000
Activation function	Swish
Batch size	32
Epochs	1,000
Test-train split	0.2
Loss function	LSE
Optimiser	Adam (0.005)

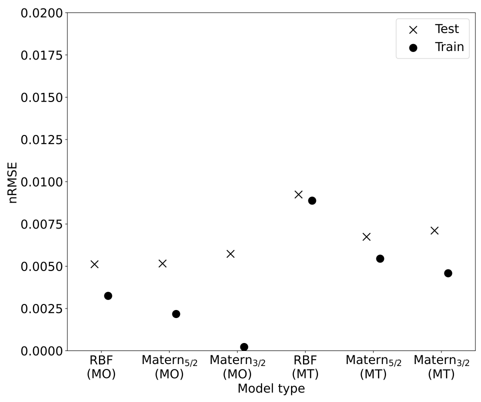
Table 6.3: Hyperparameters for MLP SV surrogate model.

As the performance of an MLP model can vary drastically depending on the initial weights, 10 differently initiated models are trained for each architecture. This enables a fair comparison between the different architectures by removing the variability of the initial weights. The mean nRMSE value for the 10 models for each architecture is shown in fig. 6.4a, along with the best and worst performing highlighted using the error bars.

The first thing to note is the variability in performance depending on the initial weights. For some models, the error is three times smaller for the best performing compared with the worst (i.e. MLP$_2$). There also seems to be very limited improvement by increasing the number of hidden layers from 2 to 3. Comparing MLP$_1$ with MLP$_2$ and MLP$_5$ with MLP$_6$ shows that models with an equal number of weights in each layer tend to perform better. Comparison between the values for the test data and the training data shows that the models generalise well and do not over-fit the data.



(a) MLP



(b) GPR

Figure 6.4: nRMSE on test and train datasets for different model types.

6.3.2 GPR

For the GPR based surrogate model, three different covariance kernel functions are considered; the RBF kernel along with the Matérn kernel with $\mu = 3/2$ and $\mu = 5/2$, denoted as Matérn$_{3/2}$ and Matérn$_{5/2}$ respectively. The RBF kernel is considered very general and good starting point when there is no assumed knowledge about the data [124]. The two Matérn kernels are chosen as these are similar to the RBF, but are able to fit more rapidly changing functional forms.

A MT-GPR (multi-task) and MO-GPR (multi-output) model are trained for each covariance kernel to compare the performance of each type. As GPR requires a matrix inversion based on the number of observations in the test dataset, the models are initially trained using only the first 200 observations from the data collected in section 6.3.1.

Training these models only require tuning 6 model parameters; 4 relating to the length scales (one for each DoF), the output scale and noise parameter. As a result, only a single model is trained for each type of GPR models.

Because the *ERMES* simulation is deterministic there is no noise in the data, therefore the initial noise value is set to 1×10^{-5} to avoid bad optima. Although it is possible to fix the noise value to zero, this would likely lead to poor performance as it would lead to over-fitting.

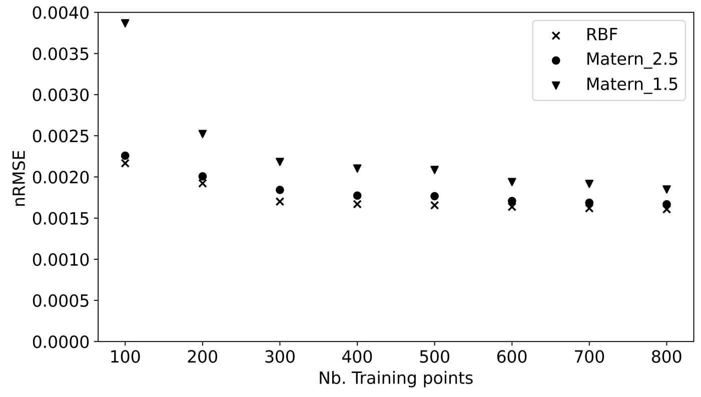
To allow for a fair comparison between the GPR and MLP models, the nRMSE is evaluated for the same test dataset used in section 6.3.1, with the results shown in fig. 6.4b.

The performance on the multi-output models are notably superior to the multi-task version, likely due to a lack of correlation between the two outputs. The models using the $\text{Matérn}_{3/2}$ kernel is the worst performing due to over-fitting of the training data, as confirmed by the very low nRMSE value on the training data. This could be rectified by placing a lower bound on the value of the noise when using the $\text{Matérn}_{3/2}$ kernel. The multi-output models for all three kernels outperform even the best MLP model, which is impressive considering that an order of magnitude fewer observations have been used to train the respective models.

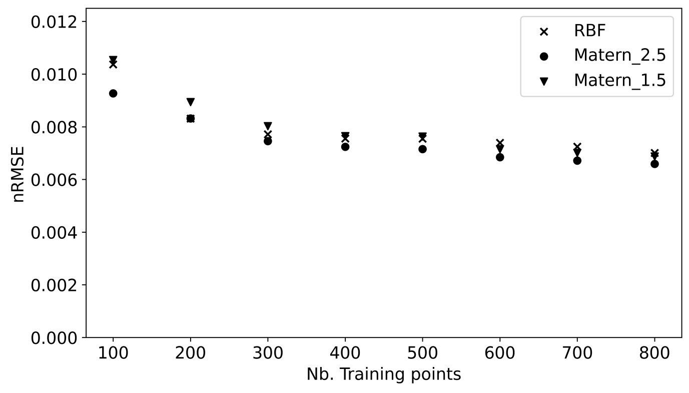
Looking at the makeup of the nRMSE values for the GPR models highlights a difference in the modelling of the power and variation values. Using the RBF kernel, over 80% of the nRMSE value is contributed to by the variation, with both Matérn kernels also showing a similar imbalance. The likely reason for this is that the variance is a less smooth, more rapidly changing profile compared with that of the power, making it a more challenging output to predict. This provides some insight as to why the MLP performs worse than the GPR, since both outputs share the majority of the weights of the model and are thus trying to predict drastically different profiles. The variation output contribution for the MLP models is between 55% and 62% of the total nRMSE, confirming that the model is sacrificing accurately modelling the power to better predict the variation.

In terms of their training time, the GPR models each take around 1 minute to train, with very little difference between the three kernels. Similarly, the smallest MLP (MLP_1 and MLP_4) takes around 1 minute, however the largest model (MLP_7) takes in the region of 10–15 minutes. These times have been recorded on the sunbird cluster.

The above GPR models discussed thus far use 200 observations, however there is more data available to assess whether the model performance can be improved. Fig. 6.5 shows the nRMSE for the power and variation for a variety of training dataset sizes, ranging from 100 to 800. With regard to the power, there is limited improvement when using more than 300 observations, while for the variation the score continues to decrease, albeit slowly, until the end, with the Matérn$_{5/2}$ the best performing kernel. Using the RBF kernel for the power and the Matérn$_{5/2}$ kernel for the variation using a training dataset of 500 observations results in an average nRMSE value of 4.4E-03 for both outputs using the test dataset, which is a substantial improvement over the MLP models using a quarter of the data.



(a) Power



(b) Variation

Figure 6.5: Test nRMSE value for models with different kernels trained using varying number of training observations.

6.3.3 Comparison of Sampling Schemes

As discussed in section 3.4, the data collection strategy chosen has a significant impact on the accuracy of the model. The work presented in this section thus far used data collected via the pre-determined Halton sequence. This section will compare this with a model generated using adaptive sampling strategies. Since GPR was considerably the best performing surrogate model, only this method will be investigated.

As the SV surrogate model considered here has two outputs, the first consideration is how to combine them in to a single value, from which the next sampling point will be chosen. The GPR models presented in this section struggled to accurately predict the variation, therefore the next sampling point will be chosen solely on this output, with the power ignored.

The initial model is trained using the first 40 observations of the training data used in section 6.3.1 and 6.3.2 (10 for each dimension of the parameter space). The performance of two adaptive schemes are assessed; the MMSE and EIGF, the theory for which are discussed in section 3.4. To take advantage of *VirtualLab*'s parallelisation, the next sampling points are collected in batches of 20. Fig. 6.6 shows the nRMSE for the variation model using data collected via the Halton sequence and the two adaptive schemes.

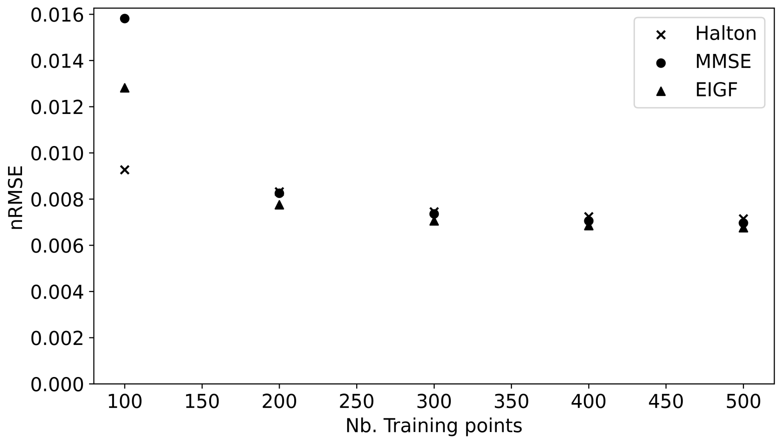


Figure 6.6: Comparison of pre-determined and adaptive sampling

The initial poor performance of the adaptive scheme is due to many of the earlier observations being assigned to the boundary of the parameter space. For example, of the first 20 points chosen by the scheme, 16 are on the corners of the 4D hyperspace, with the remaining 4 on the edges of the domain. When there are a total of 200 training observations, the EIGF scheme becomes the best performing, however after this all three are nearly indistinguishable.

While the adaptive model does result in a very marginal improvement over the pre-determined for training dataset sizes of over 500 observations, being limited to running 20 simulations at a time meant that collecting the additional 460 observations (500 minus initial 40) here took longer than collecting the 2000 observations did using the pre-determined Halton sampler (120 at a time).

The use of adaptive sampling schemes, however, can be extremely useful, especially if the goal is to search for a specific value, e.g. global minima. The improved accuracy of the models near these important features often comes at the cost of reduced accuracy elsewhere. Much of the work in this thesis requires knowledge of inverse solutions throughout the parameter space, therefore the Halton scheme is better suited for this application.

6.4 FF Surrogate Models

While the variation score modelled in the previous section is useful for optimisation purposes, it is also beneficial to view the Joule heating profile which will be delivered to the coil adjacent surface (from which the variation score can be deduced). This section investigates the accuracy of MLP and GPR models in predicting this 2D profile compared with that generated using the FE model. This will enable HIVE operators to quickly visualise the profile the coil will impart on the component to help with their decision-making.

The data required to train the surface models are extracted from the previously performed simulations. There are 10,093 nodes which make up the mesh of the coil adjacent surface, with values for each needing to be predicted using a ML model. As discussed previously, modelling this number of outputs using the GPR algorithm requires reducing the dimensionality of the data. This section will investigate the use of both the PCA and auto-encoder algorithms for this purpose. The use of MLPs with and without the use of data compression algorithms are also presented. The work of this section uses the data generated by the Halton sequence.

6.4.1 Dimensionality Reduction

PCA is an extremely popular tool for reducing the dimensionality of data. The most common method of choosing the number of PCs is based on the retention of variance in the data, as outlined in eq. 3.27, with 99% or 99.9% popular choices. This work presents a novel method for choosing the number of PCs based on the reconstruction error of the test dataset.

Since PCA is a lossy compression algorithm, reconstructing the data will invariably disagree with the original representation. Using the training data, the 2,000 PCs are identified, with fig. 6.7 showing the reconstruction error for the training and test dataset. The reconstruction error is the nRMSE between the original and reconstructed dataset. What this highlights is while the error continues to decrease for the training dataset, it plateaus for the test dataset. This is because with the 2,000 PCs the training data can be perfectly reconstructed, as this is the data used to identify the PCs. This is not true for the test dataset, with the original data never perfectly reconstructed. As a result, using increasing numbers of PCs to represent the data will only result in improvements in the accuracy of the training data, which is superfluous.

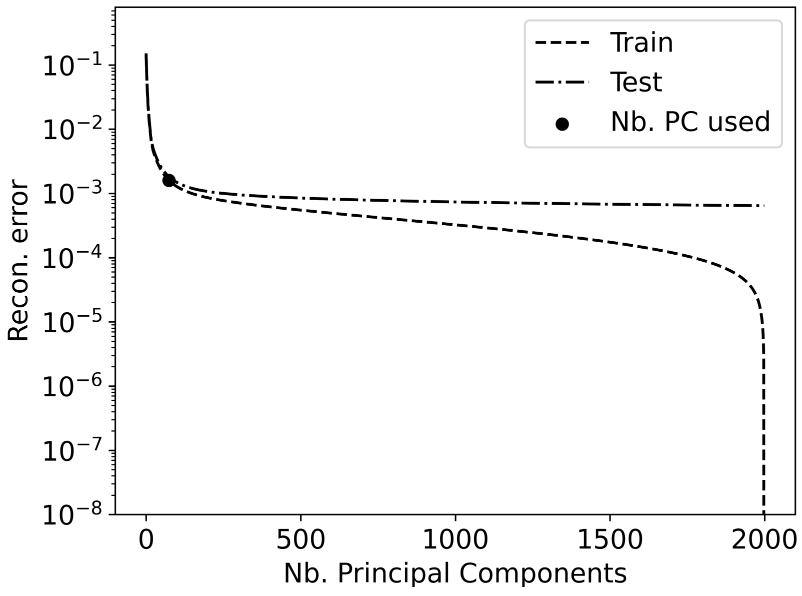


Figure 6.7: Reconstruction error on the training and test dataset for increasing number of PCs.

A more efficient approach is to choose the number of PCs for which there is only a limited improvement in the test reconstruction error thereafter (convergence of the reconstruction error). There are many ways to define convergence, however in this work it is defined as the point at which the percentage change in the loss is smaller than 1% for 3 consecutive points. 74 PCs is required to reach this convergence threshold, which is also highlighted in fig. 6.7, yielding an nRMSE value of 1.81E-03 on the test dataset. In terms of retaining variance in the data, 9 PCs are needed to retain 99%, 19 for 99.9% and 59 for 99.99%, which have nRMSE values of 1.62E-02, 6.07E-03 and 2.18E-03, respectively.

If all 2,000 PCs are used, the reconstruction error is 8.42E-04. While this is a reduction of around 50% in the reconstruction error compared with using 74 PCs, it requires training an additional 1926 outputs, which would far outweigh any potential benefit in terms of the accuracy.

To analyse the performance of autoencoders in compressing the data, two different architectures were considered. The first compresses the data down to 100 dimensions using the architecture 10093-1024-100-1024-1093. Given the number of nodes in the input and output layer, it is only feasible to have a single hidden layer in the encoder and decoder. The size of 1024 was chosen as it has base 2 and is around an order of magnitude different from the two layers either side of it. Even so, this model still has over 20 million weights. A second model which compresses the data to 500 dimensions was also considered, the architecture of which is 10093-2048-500-2048-10093 and has over 43 million weights.

The nRMSE for the test data reconstructed using the first model is 1.43E-02, which is an order of magnitude worse than the PCA when it uses fewer (74) dimensions to compress the data. Along with this, training this autoencoder took around 30 minutes, whereas performing the PCA analysis takes a matter of seconds. The second model, which in theory should perform better since the encoded data is represented by 500 dimensions instead of 100, performs substantially worse. This is likely due to the model being caught in a bad optima, however it is infeasible to train multiple models to overcome this due to the long training time, with this second model taking over 50 minutes.

Given its simplicity and low reconstruction error, the PCA algorithm will be used to reduce the dimensionality on all full field surrogate models used in this work.

6.4.2 MLP

One of the advantages of MLPs is their ability to produce models which have numerous outputs. This means that it is not strictly necessary to reduce the size of the output for an MLP model, although doing so would allow a smaller model with fewer trainable parameters, thus reducing the training time. In this section, the performance of two different types of MLPs are investigated; one which compresses the output using PCA and another which maps to the original, full output. To distinguish between them, the former will be referred to as a PCA-MLP, and the latter the Full-MLP.

Three different architecture of hidden layers are considered for both model types; a single hidden layer with 512 nodes, 2 hidden layers each with 64 nodes, and one with 3 hidden layers with 16, 64 and 256 from the input to the output. All of these models will have the same 4 inputs as those in the previous section, while the number of outputs will depend on the model type. The Full-MLP will have 10,093 nodes in the output layer, while the PCA-MLP model will have 74 nodes in the output layer, which is then be reconstructed to 10,093 using the PCA algorithm.

As in section 6.3.1, 10 differently initiated models are trained for each architecture to allow a fair comparison between each. Again, the entire dataset of 2000 observations is used to train both types of MLPs, while the other hyperparameters associated with the model and its training are the same as that used in section 6.3.1, see table 6.3. Again, the generality of all models, both MLP and GPR, are measured using the test dataset consisting of 500 observations.

The nRMSE values for the Full-MLP models are shown in table 6.4, while the PCA-MLP models are shown in table 6.5.

Architecture	Nb. Weights	Train nRMSE	Test nRMSE	Training time (m)
4-512-10093	5,180,269	8.2103E-03	8.6481E-03	68:51
4-64-64-10093	660,525	8.1520E-03	8.9601E-03	10:01
4-16-64-256-10093	2,611,709	5.9031E-03	6.5751E-03	32:17

Table 6.4: Full-MLP FF surrogate model accuracy.

In terms of the Full-MLP models, the deeper model performs best, even though it has around half the number of trainable parameters compared with the shallowest model. This improved performance could be attributed to the gradual geometrical increase in the size of the layers, as opposed to the larger jump seen in the first. Comparing the scores on the test and train data show that all models generalise well.

For the PCA-MLP models, the first point to note is that, by compressing the output layer, the number of trainable parameters associated with the model reduces by around two orders of magnitude. As expected, this has a large effect on the time taken to train these models. These models, however, perform worse than their full-field counterpart, although the difference is not surprising given the difference in model sizes. Comparison of the train and test nRMSE values again confirm that the models are not overfitting the data.

Architecture	Nb. Weights	Train nRMSE	Test nRMSE	Training time (m)
4-512-74	40,522	1.1420E-02	1.2458E-02	01:47
4-64-64-74	9,290	3.1738E-02	4.0155E-02	01:18
4-16-64-256-74	36,826	2.0900E-02	2.2609E-02	01:44

Table 6.5: PCA-MLP FF surrogate model accuracy.

As is standard practice for ML models, all inputs, and outputs have been scaled to the range [0,1] to avoid a mismatch in the magnitude of the gradients, thus speeding up training. Doing so, however, means that during the update of the weights, the error on all outputs are considered equal. For most MLP models this is perfectly reasonable, however for the PCA-MLP this approach is unwise, as it is more important to accurately model the more dominant PCs than those which contribute less to the overall prediction. One way to improve this is by weighting the outputs in terms of their importance.

Since the PCs identified using the PCA algorithm are optimal in terms of retaining variance in the data, it follows that the more dominant PCs will span a larger range. This is confirmed in fig. 6.8, where the range of data for each PC is plotted. As a result, not normalising the outputs naturally encodes the importance of the more dominant PCs. The PCA-MLP models were trained again without normalising the data in the output layer, with the results shown in table 6.6. The accuracy of these models is similar to the Full-MLP, with two of the three models returning a lower nRMSE value, but for the same low computational cost as the PCA-MLP.

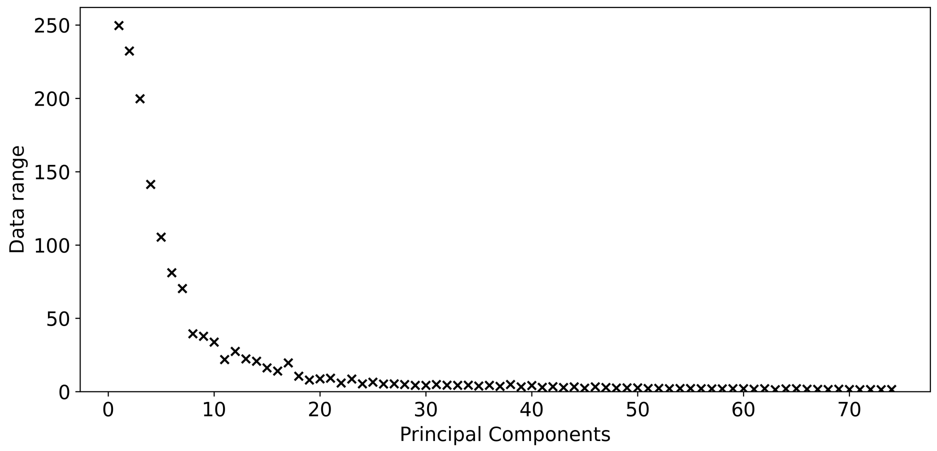
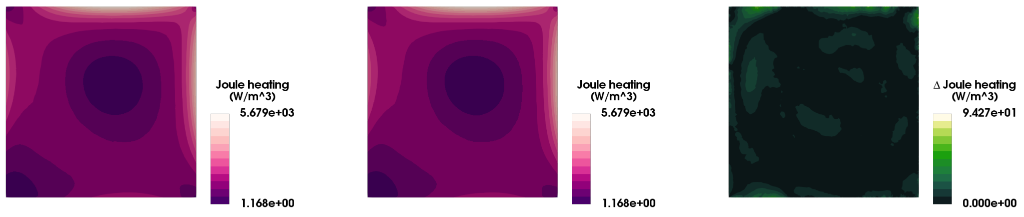


Figure 6.8: Range of data on each principal component

Architecture	Nb. Weights	Train nRMSE	Test nRMSE	Training time (m)
4-512-74	40,522	5.2170E-03	5.9145E-03	01:39
4-64-64-74	9,290	6.8840E-03	7.5769E-03	01:19
4-16-64-256-74	36,826	8.3239E-03	8.9146E-03	01:46

Table 6.6: Non-normalised PCA-MLP FF surrogate model accuracy.

With the best performing model identified, the surrogate Joule heating profile on the coil adjacent surface can now be visualised. Using an example from the test dataset, fig. 6.9a shows the 2D profile on the coil adjacent surfaces generated by the *ERMES* simulation, while fig. 6.9b shows the profile generated by the surrogate model. Fig. 6.9c shows the difference between the two, highlighting a maximum error of 94.3 W/m^3, which is less than a 3% error.



(a) FE model (b) Surrogate model (c) Absolute difference

Figure 6.9: Comparison between the FE model and MLP model for the Joule heating profile on the coil adjacent surface.

6.4.3 GPR

Similar to section 6.3.2, only 500 of the 2,000 available training data will be used to generate the FF surrogate using GPR. A separate PCA analysis was conducted using these 500 observations, with 71 PCs required to achieve the desired level of convergence, three fewer than that used for the MLP.

As highlighted previously, the more dominant PCs - those linked to higher eigenvalues - capture more generalized features of the underlying data, giving them a smoother profile. The less dominant PCs identify more specialised features, which are noisier and thus have a more rapidly changing profile. The drastically different profiles over the PCs mean it is unsuitable to model these using the multi-task GPR, and instead only the multi-output GPR algorithm is used. Similar to the section 6.3.2, 3 different models are trained which use the RBF, Matérn$_{5/2}$ and Matérn$_{3/2}$ kernel on all outputs, the accuracy of which are shown in table 6.7.

Kernel	Train nRMSE	Test nRMSE	Training time(m)
RBF	1.8996E-03	6.1062E-03	13:35
Matérn$_{5/2}$	2.0169E-03	3.5367E-03	12:42
Matérn$_{3/2}$	1.7030E-03	3.8334E-03	12:05

Table 6.7: GPR FF surrogate model accuracy.

The first thing to note is that both Matérn models outperform all three MLP models (table 6.6), with RBF outperforming two of them. A potential reason for the poorer performance of the RBF kernel is due to its smooth profile being unable to generalise well for the noisier PCs. This is confirmed in fig. 6.10, where the nRMSE for the test and train data is plotted for each PC, highlighting the overfitting for the latter PCs. The Matérn$_{3/2}$ kernel overfits the data slightly more than Matérn$_{5/2}$, which is unsurprising given its sharper profile.

One way to improve the overfitting shown by all three models would be to increase the lower bound on the noise associated with these outputs. Caution should be taken with this approach, as applying this to the more dominant PCs will likely worsen their performance, which in turn would worsen the overall model performance. As a result, no lower bound on the noise will be enforced as the model performance is deemed sufficiently accurate.

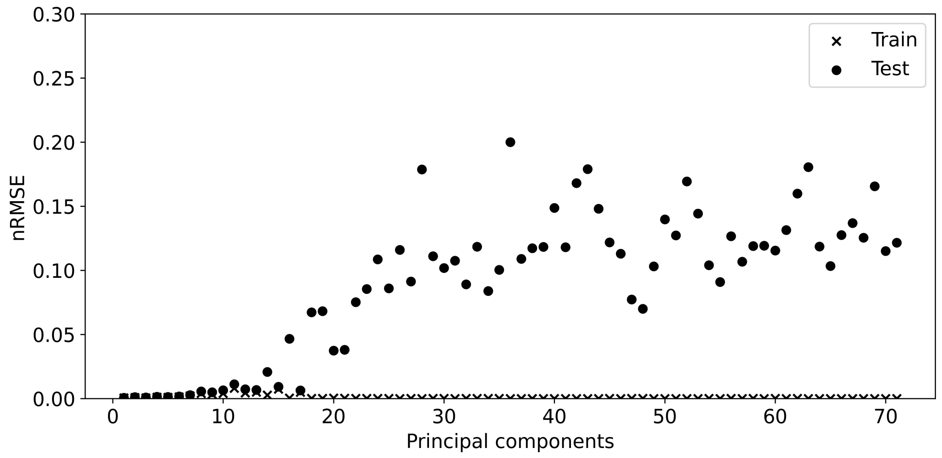
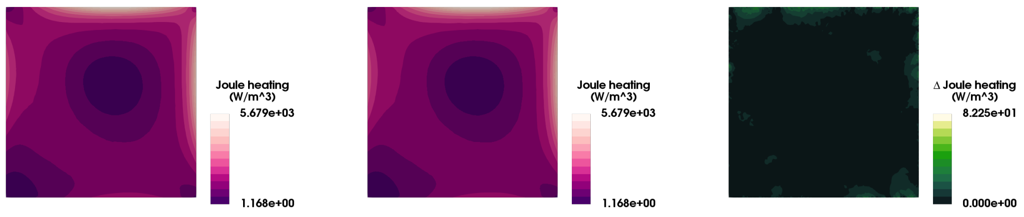


Figure 6.10: nRMSE for each PC used in RBF GPR model

As all 71 outputs have been modelled using the 3 different kernels, the best performing model for each PC could be used to give the best overall model. Choosing those which give the lowest test loss results in an overall loss of 3.4848e-03, which is only marginally better than the 3.5367e-03 for the Matérn$_{5/2}$ kernel. Given the additional effort involved in using a model with different kernels for each output, the modest reduction in nRMSE is not worth it.

Fig. 6.11 shows a comparison of the Joule heating profile predicted by the *ERMES* simulation and the surrogate model for the same testcase as that used for fig. 6.9. The maximum error using the GPR model is 82.3 W/m^3, down from 94.3 W/m^3 for the MLP model, which is expected given the improvement in the model accuracy.

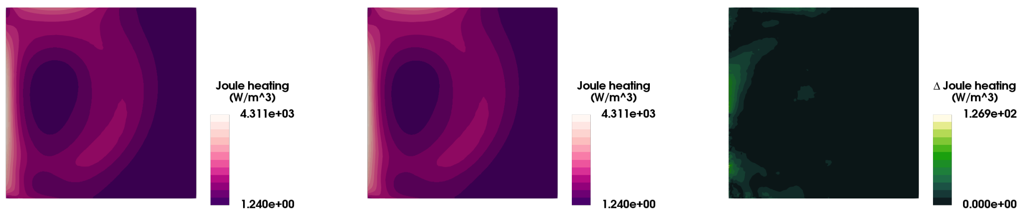


(a) FE model (b) Surrogate model (c) Absolute difference

Figure 6.11: Comparison between the FE model and Matérn$_{5/2}$ GPR model for the Joule heating profile on the coil adjacent surface.

To draw comparison between the FF surrogate models presented in this section with the SV surrogate models from section 6.3, the variation score is calculated from the output of the FF surrogate using eq. 6.1. For the Matérn$_{5/2}$ GPR model, the nRMSE measured on the test dataset is 6.850E-03, which is less than that for the best performing SV surrogate model, which was 7.153E-03. For the best performing MLP model, the nRMSE for the prediction of the variation is 9.9294E-03.

As GPR has shown to be the best method, a second surrogate model has been trained using data collected from the FE model using coil B. Again, this data was collected using the Halton sequence, where 500 observations have been collected for training. The same parameter space was used for collecting this data as was used for coil A, see table 6.1. The nRMSE on the test dataset (200 observations) is 3.0929E-03, which is similar to that for coil A. A comparison of the Joule heating profile generated by the *ERMES* simulation and the surrogate model is shown in fig. 6.12.



(a) FE model (b) Surrogate model (c) Absolute difference

Figure 6.12: Comparison between the FE model and Matérn$_{5/2}$ GPR model for the Joule heating profile on the coil adjacent surface (Coil B).

6.5 Experiment Insight

This section uses the surrogate models generated in section 6.3 and 6.4 to solve the inverse problems faced by HIVE, and to provide a greater insight of what behaviours are possible.

6.5.1 Identifying Extrema

Knowledge of the extrema within the parameter space is extremely useful to assist with decision-making. These extrema are the combination of the 4 DoFs which minimise and maximise the values of the power and variation. Using the *MultiSLSQP* package presented in section 2.4.3, 100 different initial points are used to identify the minima and maxima for both outputs within the parameter space. As discussed in section 2.4, while using more initiated points will increase the likelihood of identifying the global optima, it is not guaranteed. For this 4D hyperspace, 100 differently initiated points is deemed sufficient.

The values of the DoFs for the four extrema cases estimated using the surrogate model for coil A are shown in table 6.8. These inverse solutions are useful to provide a 'sanity check' for the methodology given our intuition about the problem. As expected, the power attains its maxima when the coil is as close to the component as possible ($z_d = 3.00E - 03$), while it attains its minima when it is at its furthest ($z_d = 6.00E - 03$).

Output	Extrema	x_d	y_d	z_d	ϕ
Power	Min.	-5.00E-03	1.50E-02	6.00E-03	2.24E-01
	Max.	3.52E-03	-1.74E-03	3.00E-03	-5.00E+00
Variation	Min.	-5.00E-03	1.50E-02	6.00E-03	5.00E00
	Max.	-1.75E-03	-1.84E-03	3.00E-03	-5.00E+00

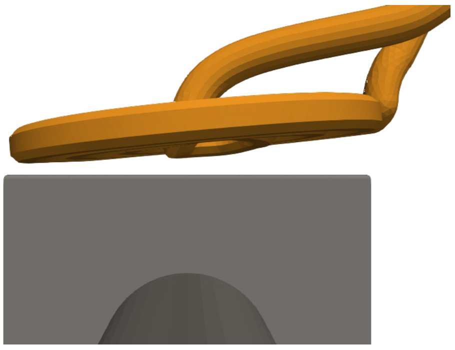
Table 6.8: Inversely informed coil parameters which minimum & maximum the power and variation using the GPR surrogate model for coil A.

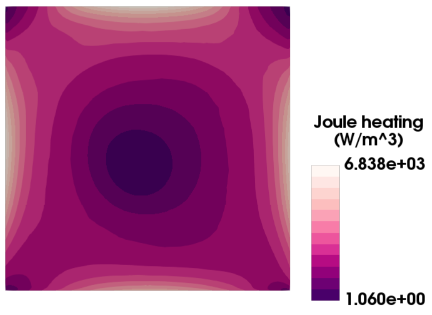
The inverse solutions provided in table 6.8 are used as inputs in to an *ERMES* simulation, with the resulting power and variation values calculated and compared with those predicted by the surrogate model. These are summarised in table 6.9, where excellent agreement is shown in predicting the values at the extrema, with 0.51% the largest percentage error. The predictions for the variation tend to be largest, which is unsurprising given that this is a more rapidly changing function. Note that these and all other *ERMES* simulations in this section were performed using a current in the coil of 300 A, unless otherwise stated.

		Power (W)			Variation		
		Surrogate (°C)	FE (°C)	Err. (%)	Surrogate (°C)	FE (°C)	Err. (%)
Power	Min.	221.06	222.01	0.02	45.22	44.99	0.51
	Max.	525.61	526.44	0.16	94.37	94.42	0.05
Variation	Min.	222.18	22.29	0.05	44.12	44.08	0.09
	Max.	515.07	515.80	0.14	97.10	97.41	0.32

Table 6.9: Comparison of power and variation predicted using the GPR surrogate model and FE simulation at inversely informed extrema (coil A).

Fig. 6.13a shows the placement of the coil with respect to the component which delivers the maximum quantity of power. The resulting Joule heating profile on the coil adjacent surface with coil A placed at this location is shown in fig. 6.13b, which was generated using the *ERMES* simulation.


(a) Coil placement.



(b) Joule heating profile.

Figure 6.13: Max. power extrema.

The same analysis was performed for coil B, with the values for the power and variation at the four extrema given in table 6.10. The power delivered to the component and the magnitude of the variation are both less using coil B compared with coil A. This is unsurprising, as coil A was designed with the goal of improving the coupling efficiency of the induction heating system, which it has achieved.

Output	Extrema	Power (W)	Variation
Power	Min.	94.72	21.64
	Max.	231.44	35.90
Variation	Min.	129.46	18.63
	Max.	207.54	55.25

Table 6.10: Power and variation predicted using the GPR surrogate model at inversely informed extrema (coil B).

As discussed in section 5.3.2, the Joule heating is proportional to the square of the current in the coil. Since the power and variation are both linear functions of the Joule heating field, these both are also proportional to the square. Therefore, by increasing the current for coil B by a factor of 1.5 (up to 450 A) the values of both power and variation will increase by a factor of 2.25, resulting in values of power and variation which are similar to those for coil A with a current of 300 A.

It would have taken a number of experiments for HIVE operators to learn the different power scales required for coil A and coil B, however through the use of surrogate models, this difference is more easily measurable.

6.5.2 Creating a Performance Envelope

More often than not, experiments in HIVE will aim to find a balance between the power and variation. There are numerous combination of the 4 DoFs which will deliver a certain power to the component, i.e. the multiple solutions to the inverse problem. Each combination will impart a different Joule heating profile on to the component, and thus have different variation scores. To better emulate fusion like conditions, it is desirable to have the heating as uniform as possible. This can be achieved by placing an equality constraint on the power while minimising the variation using the *MultiSLSQP* package.

Table 6.11 shows the coil configurations which minimise the variation (best) and maximise it (worst) while ensuring the at least power 400 W is delivered to the component using coil A.

	x_d	y_d	z_d	ϕ	Variation
Best	-3.00E-03	-4.27E-03	4.35E-03	5.00E+00	6.93E+01
Worst	-4.71E-03	-1.44E-02	3.00E-03	-5.00E-03	8.74E+01

Table 6.11: Best & worst configuration for delivering 400 W of power to the component (coil A).

Repeating this process and fixing the power to values ranging from the minimum, 221.06 W, to the maximum, 525.61, yields an envelope of the variation available for the different powers, which is shown in fig. 6.14. This envelope allows the user to identify the different ways in which the component can be loaded, allowing more informed decisions to be made.

The same analysis was conducted for coil B, where the power ranges from 213.12 W to 520.74 W (the min. and max. value when the current is 450 A). The envelope for coil B is displayed in fig. 6.15, along with the envelope for coil A for ease of comparison. Interestingly, power and variation seem to be less correlated for coil B compared with coil A. coil B is able to provide more uniform heating to the component than coil A for the majority of the power range. That said, doing so will require a 50% increase in the power supply due to its poor coupling efficiency.

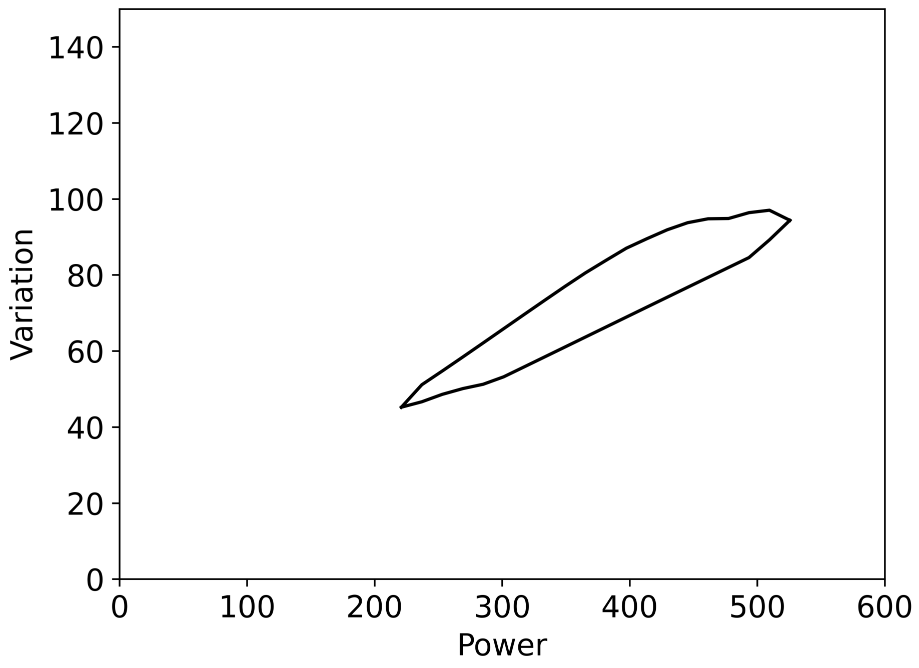


Figure 6.14: Envelope of possible outcomes for power and variation using coil A

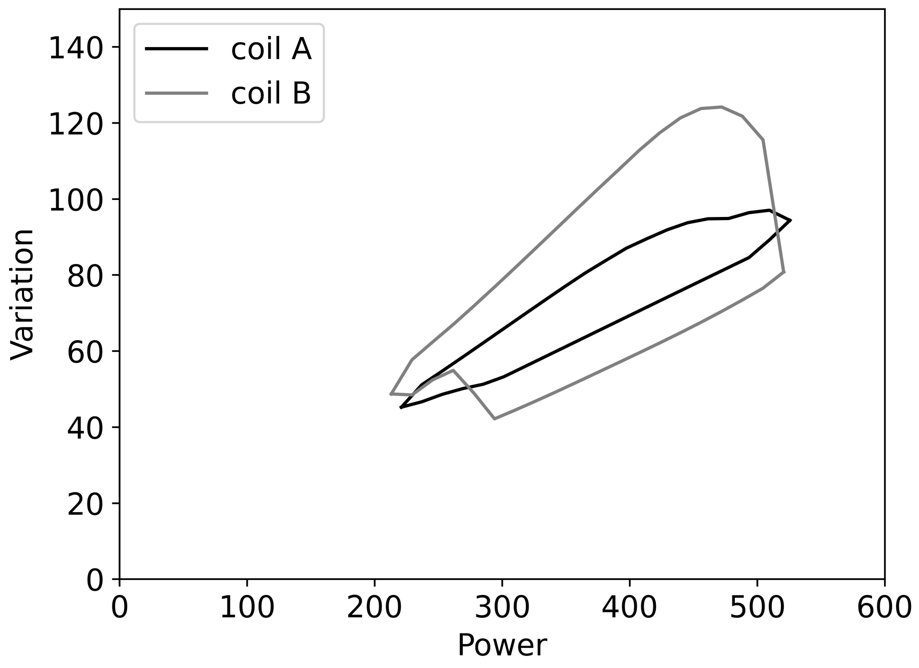


Figure 6.15: Performance envelopes for coil A (300 A) and coil B (450 A)

6.6 Chapter Summary

This chapter has shown how ML can provide a great deal of insight regarding the HIVE experiment without the need for any prior knowledge or experience. 1000 simulations (500 for each coil) are sufficient to train surrogate models to perform this analysis, all in less than two hours using the *VirtualLab* package (using 120 cores). As the workflow is entirely automated, this can be scheduled to run overnight, with the analysis ready for the following day of testing. This would enable HIVE to test components much more rapidly and avoid unnecessary experiments. The contents of the *VirtualLab* run file necessary to create the surrogate models and perform this analysis is shown in listing A.1 in the appendices.

While the analysis carried out in this chapter have been for two coils which are actually used in experiments, it can just as easily be performed for conceptual coil designs. This would enable their relative strengths, weaknesses, and performance envelope to be identified before manufacturing, saving both time and money.

VirtualLab makes generating surrogate models extremely easy, whether it's SV or FF surrogates. This means that the user doesn't require knowledge of the ML algorithms itself, enabling those who are unfamiliar with this field to still take advantage of the opportunity they provide. This chapter discussed the generation of 2D FF surrogate models, with the PCA algorithm shown to be much more dependable than auto-encoders for the task of data compression. A novel approach to choosing the number of PCs to retain was also presented, along with guidance on how to use PCA with MLPs for improved performance. This chapter also showed that FF surrogates are able to outperform SV surrogates in predicting key performance indicators of a simulation, with the former able to more accurately predict the variation of the heating profile.

Chapter 7

Smarter Component Testing

7.1 Introductory Remarks

This chapter builds on the knowledge of the previous chapter through the development of more complex FF surrogate models to assist the generation of more informative DoPEs. As outlined in section 4.4, a comprehensive DoPE requires identifying experimental parameters which deliver various thermal and stress outcomes to specific parts of the component. This means that not only does the surrogate model need to predict the behaviour throughout the 3D component, but also requires more inputs than those of the previous chapter, since a greater number of experimental parameters affect the thermal and stress state of the component.

The combined use of two FF surrogate models enables smarter testing by taking into account the stress state of the component during an experiment. This is something which the operators of HIVE currently have no means of inferring, but is essential in informing the next iteration of the component design. The use of surrogates is not only more accurate, fast, and robust, they also remove the need for inefficient trial & error during experiments and operator knowledge of the facility.

The use of 3D FF surrogate models for SAO purposes is believed to be a novel contribution to the field, as is the combination of two such models for solving constrained optimisation problems. The outline of this chapter is as follows; firstly, section 7.2 presents the different surrogate models developed for predicting the temperature field and Von Mises stress field throughout the component, along with their accuracy compared with the computational model of HIVE. Following this, these surrogate models are used to identify experimental parameters which make up a more comprehensive DoPE, an example of which can be found in section 7.3.3. In section 7.4, the surrogate models are used to assess the sensitivity of the experimental parameters with regard to the thermal and stress behaviour of the component, highlighting their versatility to a broader range of tasks. Here, it is shown that the parameter sensitivity which would be inferred from thermocouple data does not necessarily translate to quantities such as the maximum temperature. Finally, section 7.5 showcases the flexibility of the developed *VirtualLab* workflow for DoPE construction by performing the same analysis on a new, multi-material component geometry.

7.2 FF Surrogate Models

As shown in section 5.3.3, changing the temperature of the coolant and its velocity through the pipe will affect the overall component behaviour, however, changing the coolant pressure does not significantly affect the amount of heat which will be extracted from the component. Instead, higher coolant pressure enables higher coolant temperatures to be used. For this work high coolant temperatures (in excess of 100 °C) are not considered, therefore the coolant pressure is fixed at 0.5 MPa for this analysis.

This means that for a given coil design, there are 7 experimental parameters which impact the component's behaviour. The ranges for the 4 DoFs relating to the coil configuration are the same as those used in chapter 6. The coolant temperature, $T_{coolant}$ ranges from 30 °C to 80 °C, the flow rate of the coolant ranges from 5 l/min to 55 l/min which for the given pipe diameter of 12.7 mm equates to a coolant velocity, $V_{coolant}$, ranging between 0.6578 m/s and 7.2363 m/s. The current in the coil, I, ranges from 100 A to 600 A. For reference, a summary of the parameter space is given in table 7.1.

Choosing the same ranges for the coil configuration DoFs as the previous chapter means that the previously performed *ERMES* simulations are still valid. *VirtualLab* makes it easy to perform particular sections of the workflow and combining them with previously generated data. This means that the data required to perform this analysis only requires performing the 1D coolant analysis and the *code_aster* thermo-mechanical simulation.

This is another example of where using pre-determined sampling strategies are beneficial. As the data hasn't been collected for a specific goal, it is useable

Parameter	**Lower bound**	**Upper bound**
x_d	-5.00E-03	5.00E-03
y_d	-1.50E-02	1.50E-02
z_d	3.00E-03	6.00E-03
ϕ	-5.00E+00	5.00E+00
$T_{coolant}$	3.00E+01	8.00E+01
$V_{coolant}$	6.58E-01	7.24E+00
I	1.00E+02	6.00E+02

Table 7.1: Parameter space considered for design of physical experiments.

for a wider range of applications, whereas an adaptive data sampling approach would likely have biased the data collection to a specific region, reducing its global accuracy.

Values from the next 3 dimensions of the Halton sequence are used to vary the values for the coolant temperature, coolant velocity and coil current. The surrogate models generated in this chapter use 500 observations for training and 200 for testing the generality of the model.

7.2.1 Temperature Field

To generate a surrogate of the 3D temperature field, denoted by T_{surr}, the temperature at all 90,432 nodes of the mesh must be modelled. Similar to the FF surrogate model in section 6, the number of PCs used is the number required for convergence of the reconstruction error on the test dataset. For this dataset, this is achieved using 95 PCs, which are modelled using the MO-GPR algorithm, with the RBF, Matérn$_{5/2}$ and Matérn$_{3/2}$ covariance kernel functions again used to assess the performance of each.

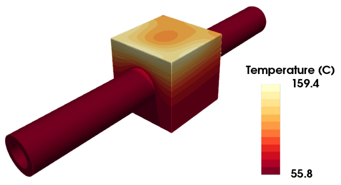
The nRMSE values for surrogates of the temperature field for coil A and coil B are shown in table 7.2. For both coils, the RBF kernel gives the lowest score, with the other two suffering a little more from overfitting. The nRMSE scores for all models suggest they are sufficiently accurate for the intended application.

To visualise their accuracy, a comparison of the temperature profile calculated by the best performing surrogate model (RBF) and the *code_aster* FE simulation is shown in fig. 7.1 for coil A and fig. 7.2 for coil B. Both simulations are conducted using the same set of experimental parameters given in table 7.3 for the two different coil designs.

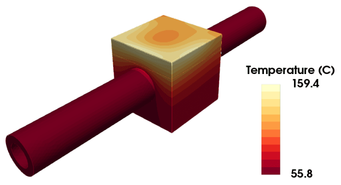
	kernel	Test nRMSE	Train nRMSE
	RBF	1.48E-03	4.61E-04
Coil A	Matérn$_{5/2}$	1.54E-03	1.09E-04
	Matérn$_{3/2}$	2.42E-03	7.84E-05
	RBF	1.41E-03	3.94E-04
Coil B	Matérn$_{5/2}$	1.53E-03	4.64E-05
	Matérn$_{3/2}$	2.53E-03	3.84E-05

Table 7.2: Accuracy of temperature surrogate model, T_{surr}, using different kernels.

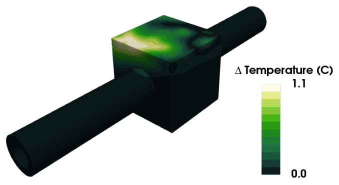
The nRMSE for these two cases are 4.11E-03 and 4.94E-03, which is larger than the average value over the entire test dataset. Even on a worse performing example, the largest percentage error between the two is less than 1%, highlighting very good agreement. This is confirmed in fig. 7.1c and fig. 7.2c, where the maximum absolute temperature difference is 1 °C.



(a) FE model

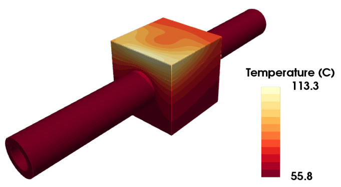


(b) Surrogate model

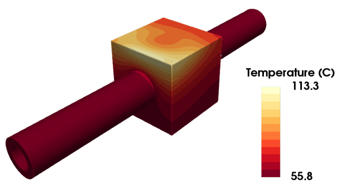


(c) Absolute difference

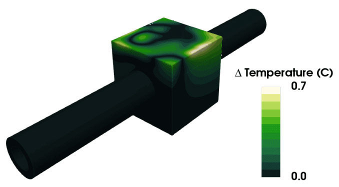
Figure 7.1: Comparison between the FE model and RBF GPR model for the 3D temperature profile (coil A).



(a) FE model



(b) Surrogate model



(c) Absolute difference

Figure 7.2: Comparison between the FE model and RBF GPR model for the 3D temperature profile (coil B).

Parameter	Value
x_d	4.34E-04
y_d	-6.65e-03
z_d	4.27E-03
ϕ	3.45E+00
$T_{coolant}$	5.58E+01
$V_{coolant}$	4.41E+00
I	1.14E+02

Table 7.3: Input parameters for example shown in fig. 7.1 and fig. 7.2

7.2.2 Von Mises Stress Field

While there are a number of different mechanical metrics by which to judge a component's performance, the most commonly used is the Von Mises stress. The Von Mises stress, σ_{VM}, is a function of the stress tensor, used to predict whether a ductile material will yield or not. The formula for the Von Mises stress is given by eq. 7.1, where σ_{xx}, σ_{yy} and σ_{zz} are the normal stresses and σ_{xy}, σ_{yz} and σ_{zx} are the shear stresses.

$$\sigma_{VM} = \sqrt{\frac{(\sigma_{xx}-\sigma_{yy})^2+(\sigma_{yy}-\sigma_{zz})^2+(\sigma_{zz}-\sigma_{xx})^2+6(\sigma_{xy}^2+\sigma_{yz}^2+\sigma_{zx}^2)}{2}} \tag{7.1}$$

The benefit of using the Von Mises stress is that the surrogate model, denoted by VM_{surr}, will only need to predict a scalar value at each node, instead of a field made up of normal and shear stresses. To achieve convergence of the reconstruction error, 72 PCs are required, which are again modelled using the three different kernels. The results for each kernel for the two different coils are shown in table 7.4.

	kernel	Test nRMSE	Train nRMSE
Coil A	RBF	1.76E-02	6.57E-03
	Matérn$_{5/2}$	1.61E-02	4.53E-03
	Matérn$_{3/2}$	1.63E-02	4.09E-03
Coil B	RBF	2.19E-02	7.96E-03
	Matérn$_{5/2}$	2.08E-02	5.02E-03
	Matérn$_{3/2}$	2.12E-02	4.68E-03

Table 7.4: Accuracy of Von Mises surrogate model, VM_{surr}, using different kernels.

The first thing to note is that this surrogate model is not as accurate as that of the temperature field, with the nRMSE values around an order of magnitude worse for both coils. The reason for this is that the PCA algorithm is unable to reconstruct the Von Mises data to the same level of accuracy as it did the temperature data. This is highlighted in fig. 7.3, where the reconstruction error for the temperature data is nearly two orders of magnitude lower than it is for the Von Mises data. One reason for this could be that the Von Mises stress is six stress components combined in to one (see eq. 7.1), which could make it noisier and thus more difficult to accurately project on to linear subspaces.

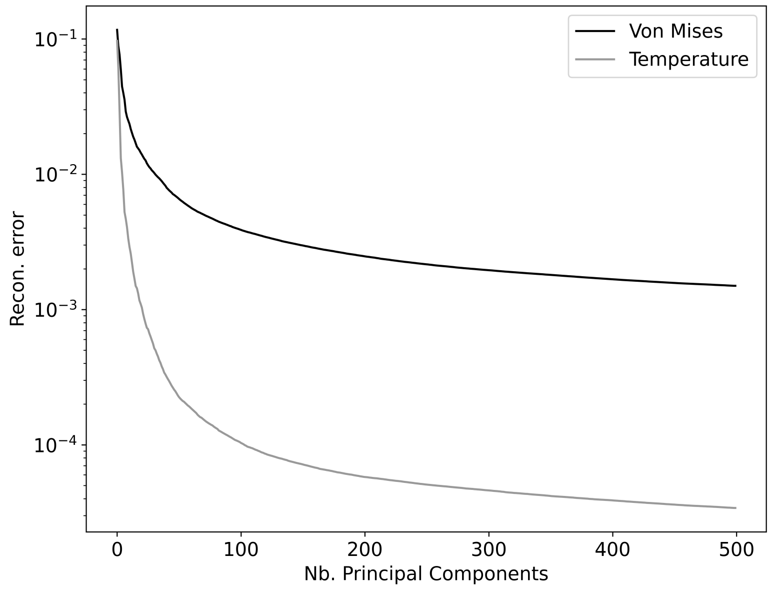
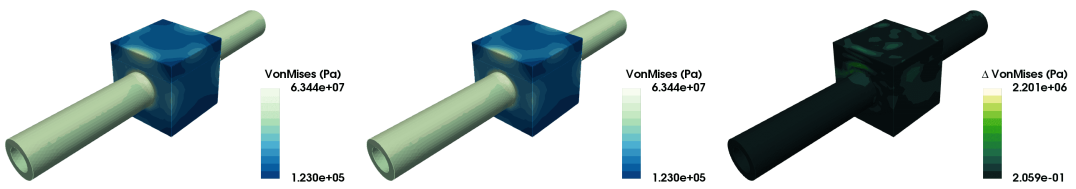


Figure 7.3: Reconstruction error v the number of PCs used for the temperature and Von Mises data (test dataset).

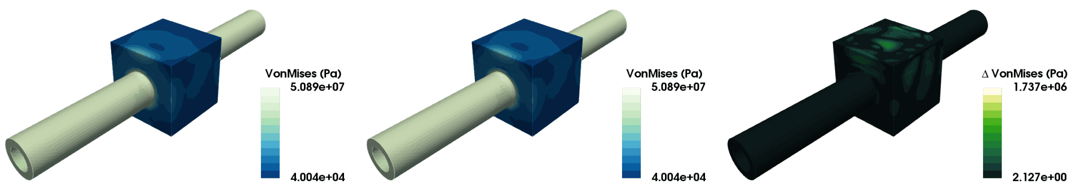
This means that the accuracy of the Von Mises surrogate model will never be as accurate as the temperature profile using the PCA algorithm. Due to the size of the input and output layer, training an autoencoder is infeasible due to it having many hundreds of millions of weights, with no guarantee of an improvement in accuracy, as was seen in section 6.4.1.

The corresponding Von Mises stress fields to the temperature profiles shown in fig. 7.1 and 7.2 are shown in fig. 7.4 and 7.5 respectively. The predicted Von Mises field has been generated using the Matérn$_{5/2}$ kernel, as this was the best performing of the three. Even though the surrogate model of the Von Mises field is not as accurate as the temperature field, the largest absolute error is still only around 2 MPa, equating to around a 7% error. This is still deemed sufficiently accurate for this application, especially considering that an FE simulation of HIVE takes around 15 minutes while the surrogate model takes less than 2 seconds to generate the stress field.



(a) FE model (b) Surrogate model (c) Absolute difference

Figure 7.4: Comparison between the FE model and RBF GPR model for the 3D temperature profile (coil A).



(a) FE model (b) Surrogate model (c) Absolute difference

Figure 7.5: Comparison between the FE model and RBF GPR model for the 3D temperature profile (coil B).

7.3 Constructing a DoPE

7.3.1 Temperature Related

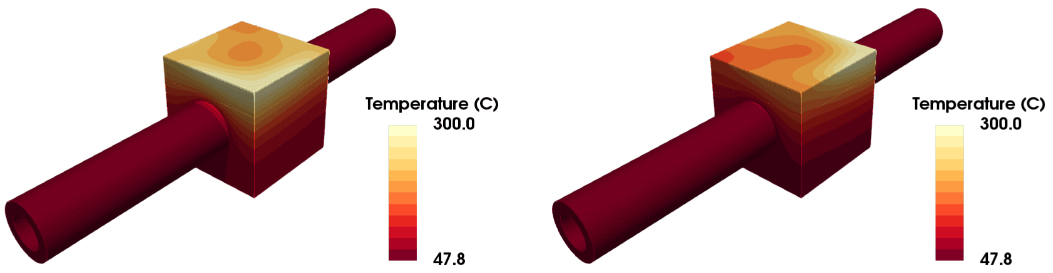
The first task is to identify the experimental parameters which will ensure the maximum temperature for a component reaches a certain value. This is achieved using eq. 7.2, where T^* is the desired temperature and $\mathcal{P}$ is the parameter space where admissible solutions are searched.

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{P}}(\max(T_{surr}(\mathbf{x})) - T^*)^2 \tag{7.2}$$

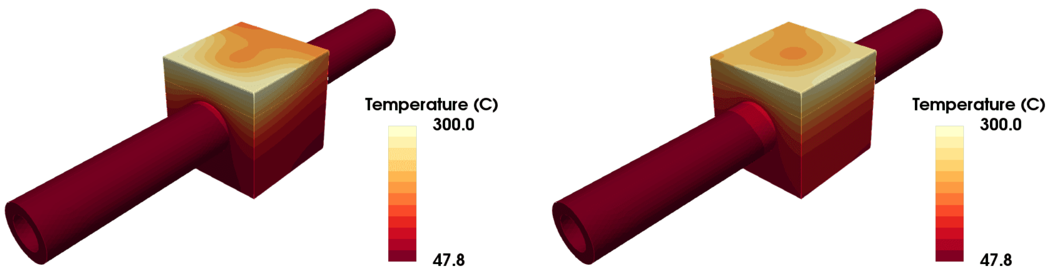
When components are tested for this purpose in HIVE, it is often beneficial to gradually heat the component. This not only provides the client - those who have designed the component under testing - with data for different loading conditions, but also potentially avoids scenarios where no usable data is collected because of component failure, e.g. debonding at the interface of two materials. This is achieved by identifying the experimental parameters which deliver a certain lower temperature to the component, and then gradually increasing the power delivered to the induction heating system.

In this example, the component is to be initially heated up to achieve a maximum temperature of 300 °C, which is then gradually increased to 700 °C in increments of 100 °C.

All 7 experimental parameters affect the temperature profile of the component, therefore there are numerous combination of these which will deliver the desired behaviour. Fig. 7.6 shows four of the different temperature profiles which deliver a maximum temperature of 300 °C, which have been generated using the surrogate model using inverse solutions generated from eq. 7.2.



(a) Profile 1 (b) Profile 2



(c) Profile 3 (d) Profile 4

Figure 7.6: Example of temperature profiles which deliver a maximum temperature of 300 °C.

For simplicity, the most desirable heating profile is chosen, which in this case is deemed to be Profile 4 (fig. 7.6d). The experimental parameters used to achieve this temperature profile are shown in table 7.5. The next step is to identify the current which will deliver 400 °C. This is again achieved using eq. 7.2, with $T^* = 400$ and all experimental parameters other than the coil current staying fixed at the values given in table 7.5. This ensures that only the coil current will need to be updated during the incremental loading of the component. The estimate current required to deliver 400 °C is 208.19 A, increasing from the 168.02 A required to deliver the maximum temperature of 300 °C.

Parameter	**Value**
x_d	3.86E-03
y_d	-2.62E-03
z_d	2.87E-03
ϕ	4.26E+00
$T_{coolant}$	6.65E+01
$V_{coolant}$	5.14E+00
I	1.68E+02

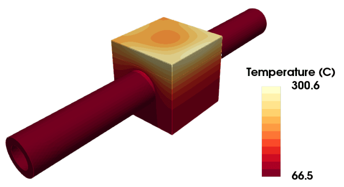
Table 7.5: Chosen experimental parameters to deliver 300 °C to a component.

The same procedure is followed for 500 °C, 600 °C and 700 °C, with the resulting currents given in table 7.6. The non-linearity of the experiment is again highlighted here, with an increase of 40 A required to increase the maximum temperature from 300 °C to 400 °C, but only an increase of 35 A to increase from 600 °C to 700 °C.

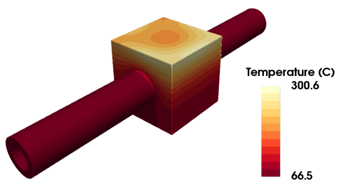
Temperature (°C)	**Coil current (A)**
300	168.02
400	208.19
500	246.14
600	282.63
700	317.88

Table 7.6: Coil current required to reach increasing maximum temperatures.

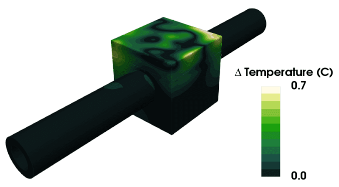
Verification of these inverse solutions is achieved by performing an FE simulation using the advised experimental parameters. The simulated temperature field for the 300 °C experimental parameters is shown in fig. 7.7a and for the 700 °C experimental parameters in fig. 7.8a. The first thing to note is that the maximum temperature for both simulations are extremely close to what was desired, with less than a 1 °C difference for both examples. Alongside these are the temperature field predicted by the surrogate model (fig. 7.7b and 7.8b) and the absolute difference between the two (fig. 7.7c and 7.8c), highlighting very good agreement with a maximum error of 3.5 °C.



(a) FE model

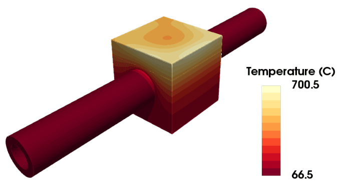


(b) Surrogate model

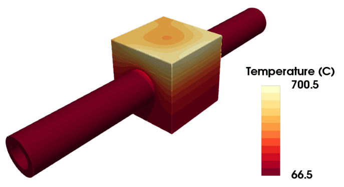


(c) Absolute difference

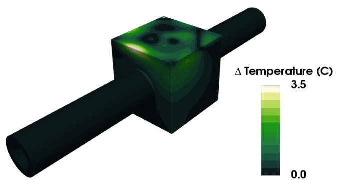
Figure 7.7: Validation of inverse solution to deliver a maximum temperature of 300°C.



(a) FE model



(b) Surrogate model



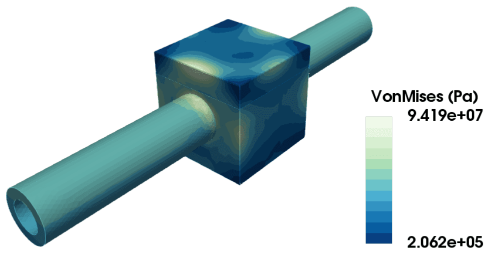
(c) Absolute difference

Figure 7.8: Validation of inverse solution to deliver a maximum temperature of 700°C.

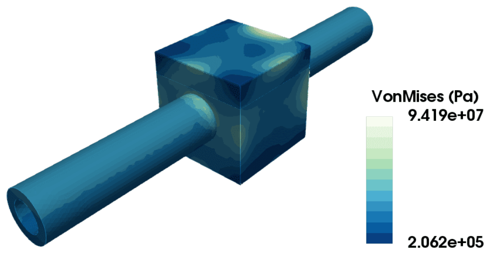
7.3.2 Stress Related

The previous example only considered the temperature profile of the component when deciding the best experimental parameters to choose, however using the surrogate Von Mises stress field model a second metric by which to choose the appropriate parameters is available. Instead of choosing those which gives the ‘best’ looking temperature profile, it is possible to choose the experimental parameters which stresses the component in a certain way, such as minimising or maximising stresses, along with delivering a desirable thermal response.

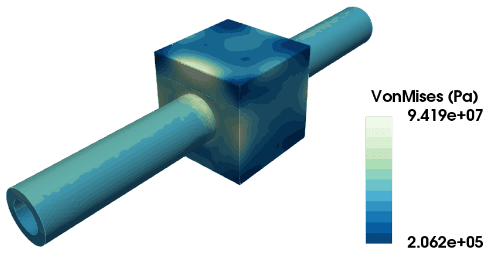
The previous example showed there are numerous temperature profiles which delivered a maximum temperature of 300 °C to the component, however each of these will yield a drastically different Von Mises stress profile. Fig. 7.9 shows the corresponding Von Mises stress field for the temperature fields shown in fig. 7.6, showing that for these four examples alone, the maximum Von Mises stress the component is subjected to ranges from 93 MPa to 110 MPa.



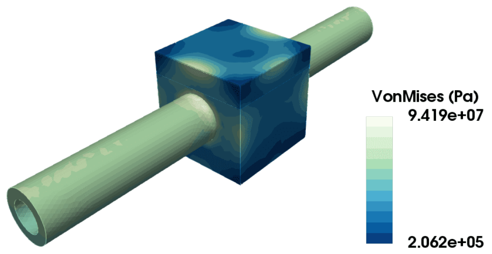
(a) Profile 1



(b) Profile 2



(c) Profile 3



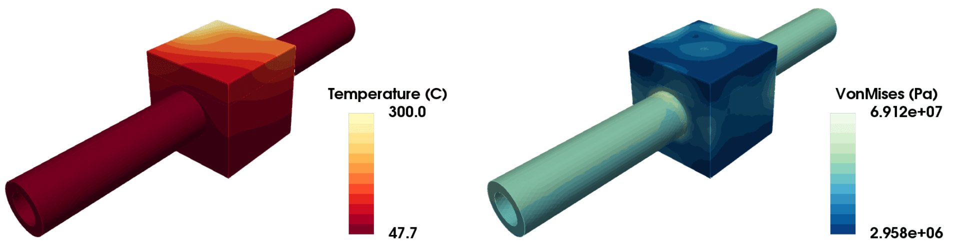
(d) Profile 4

Figure 7.9: Corresponding Von Mises stress fields to the temperature profiles shown in fig. 7.6.

Often it will be desirable to heat a component to reach a certain temperature but by minimally stressing the component. This will require identifying a set of experimental parameters which ensures that the maximum temperature in the component reaches T^*, while also minimising the maximum Von Mises stress a component is subjected to. This is summarised as the constrained optimisation problem outlined in eq. 7.3.

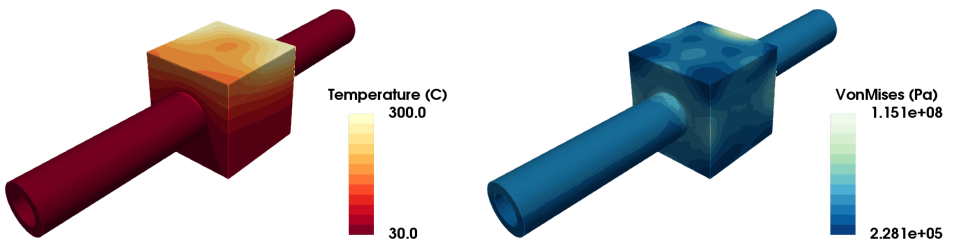
$$\begin{aligned}\mathbf{x}^* &= \arg\min_{\mathbf{x}\in\mathcal{P}}(\max(VM_{surr}(\mathbf{x}))) \\ s.t \quad &\max(T_{surr}(\mathbf{x})) = T^*\end{aligned} \tag{7.3}$$

Using the *MultiSLSQP* package to solve this problem with $T^* = 300$ provides a set of experimental parameters, given in table 7.7, which has a maximum Von Mises stress of 69 MPa. Performing the same analysis with the goal of maximising the objective instead results in a Von Mises stress profile with a maximum value of 115 MPa, nearly double that of the lower bound. Again, the experimental parameters used to achieve it are shown in table 7.7. The temperature and Von Mises stress profile for these experimental parameters is shown in fig. 7.10 and 7.11 for the minimum and maximum Von Mises cases respectively.



(a) Temperature (b) Von Mises

Figure 7.10: Profiles which minimisethe maximum Von Mises stress of a component heated to 300 °C.



(a) Temperature (b) Von Mises

Figure 7.11: Profiles which maximisethe maximum Von Mises stress of a component heated to 300 °C.

Parameter	Min.	Max.
x_d	-5.00E-03	9.81E-04
y_d	1.50E-02	8.53E-03
z_d	3.00E-03	4.49E-03
ϕ	-5.00E+00	6.41E-01
$T_{coolant}$	4.77E+01	3.00E+01
$V_{coolant}$	6.58E-01	6.99E+00
I	1.48E+02	1.97E+02

Table 7.7: Experimental parameters which minimise and maximise the maximum Von Mises stress of a component heated to 300 °C.

Similarly to the work carried out for the power and variation in the previous chapter, a plot of the component's maximum Von Mises stress range versus its maximum temperature can be constructed. This is shown in fig. 7.12, where the upper and lower bounds of the maximum Von Mises stress are shown for both coil A and coil B for maximum component temperatures ranging between 300 °C and 700 °C. In this regard, both coils perform very similarly, with coil A having a slightly smaller range of values.

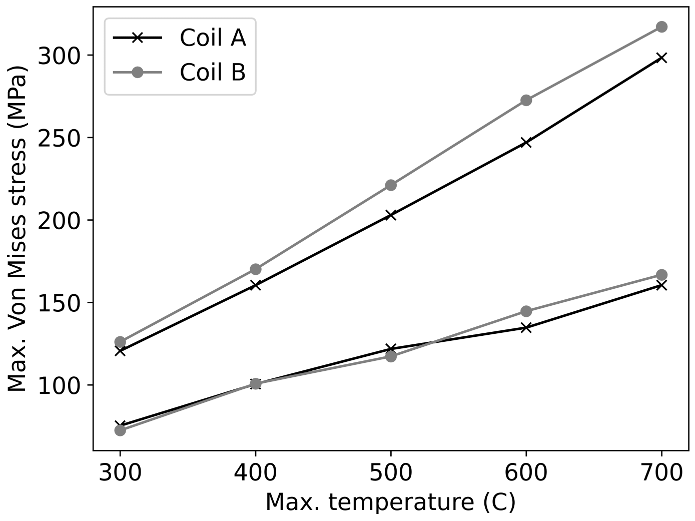


Figure 7.12: Envelope of performance of maximum Von Mises stress versus maximum temperature within the component.

7.3.3 DoPE Example

The two inverse problems discussed in this section thus far are just two examples of desirable results one would like to achieve from an experiment in HIVE. There are, however, numerous other desirable experiments which may need to be performed to qualify a component's suitability or verify the computational model, for example. Below is an example list of different experimental outcomes which are desired when testing a component.

1. Delivering the maximum temperature.
2. Delivering the maximum Von Mises stress.
3. Achieving 100 °C in the pipe while minimising the component's maximum temperature.
4. Delivering the minimum Von Mises while achieving 300 °C in the component.
5. Delivering the most uniform heating profile while also ensuring at least 500 °C is delivered to the component

The experimental parameters required to achieve these results are shown in table 7.8, which also briefly describes the expected behaviour. Experiment 1 is that which delivers the behaviour defined in point 1 in the list immediately above. For this sample, the maximum temperature is the infeasible number of 2045 °C, about double that of the melting point of titanium.

Certain experiments are useful as a sanity check for the method. For example, to achieve experiment 3 it is logical that the optimal solution is to get the pipe as hot as possible using the coolant parameters. This is highlighted in the results, where the coolant temperature is at its maximum (80 °C) and the velocity at its minimum (0.658 m/s), thus allowing the pipe to reach 100 °C with the maximum component temperature only needing to reach 140.14 °C.

This framework makes it easy to construct a much more insightful and varied DoPE than previously possible using the currently available data in HIVE and human intuition. This enables a much more comprehensive suite of tests to be performed, allowing better characterisation of a component to assist the next iteration of the design cycle.

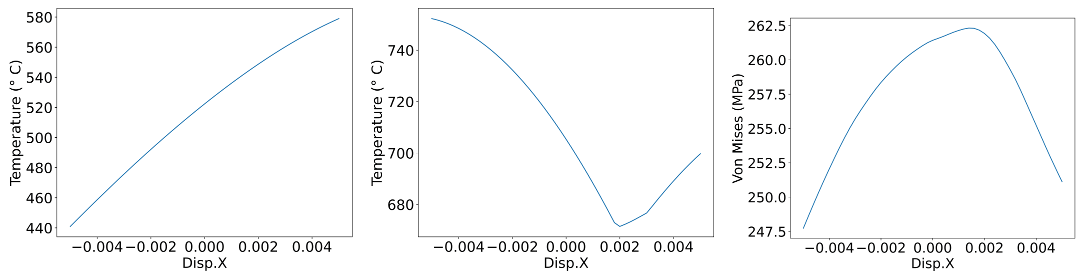
Exp.	y_d	y_d	z_d	ϕ	$T_{coolant}$	$V_{coolant}$	I	Anticipated result
1	-5E-03	-1.16E-02	3.00E-03	-5.00E+00	8.00E+01	6.58E-01	6.00E+02	A max. temperature of 2045.42 °C
2	2.61E-03	-3.50E-03	3.00E-03	-5.00E+00	3.00E+01	7.24E+00	6.00E+02	A max. Von Mises stress of 513.98 MPa
3	3.91E-03	-2.92E=03	6.00E-03	-5.23E-01	8.00E+01	6.58E-01	1.00E+02	A max. temperature of 140.14 °C.
4	-5.00E-03	1.50E-02	3.00E-03	-5.00E+00	4.77E+01	6.58E-01	1.48E+02	A max. Von Mises stress of 69.26 MPa
5	-5.00E-03	1.50E-02	6.00E-03	5.00E+00	4.19E+01	5.77E+00	3.044E+02	A max. temperature of 500.00 °C

Table 7.8: DoPE for titanium component and anticipated outcome

7.4 Parameter Sensitivity

Prior to testing a component in HIVE, it is useful to understand the sensitivity of the component's response to changes in the experimental parameters. Performing parameter sensitivity analysis is much easier and faster when surrogate models are available. Here, the value for one parameter is varied across its range while the others remain fixed, enabling its impact to be isolated. In the analysis carried out below, the values for the fixed parameters are set to the mid-pint value of their individual range, e.g. the coolant temperature is fixed at 55 °C, the mid-point between 30 and 80 °C. For brevity, the sensitivity of experimental parameters is only detailed for coil A, however similar behaviour was observed for coil B.

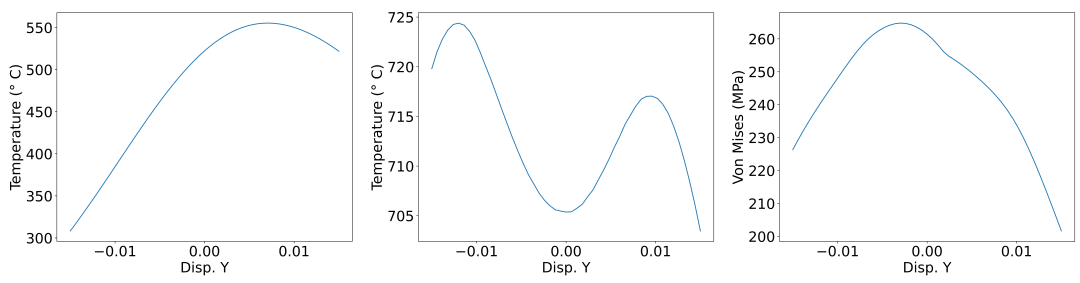
Until now, the effect of varying HIVE's parameters on the result have been measured by observing the changes to the thermocouple data collected during the physical experiment. To emulate this, a point on the side of the component (where thermocouples would usually be attached) is chosen, where temperature data is predicted using the surrogate model. Fig. 7.13a to 7.19a plots the change in temperature at this point with the varying of each experimental parameter. As expected, decreasing the distance between the coil and the component increases the temperature (fig. 7.15a), as does increasing the coil current (fig. 7.19a). Even these relatively simplistic curves demonstrate the nonlinearity of the problem, whereby decreasing the coolant temperature by 50 °C only reduces the measured point temperature by around 30 °C. A similar temperature change is observed when increasing the coolant velocity from its minimum up to its maximum.



(a) Temperature (TC) (b) Temperature (max.) (c) Von Mises (max.)

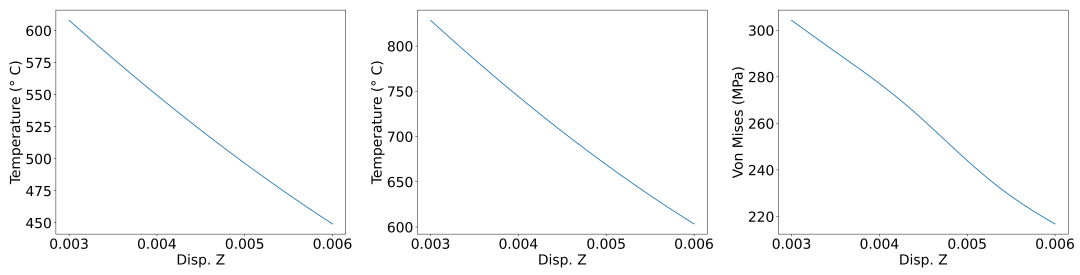
Figure 7.13: Sensitivity to changes in x displacement (x_d)

Having a surrogate model presents the opportunity to also quickly monitor how the maximum temperature is affected by changes to the experimental parameters, which is shown in fig. 7.13b to 7.19b. The first thing to note is the difference in the profiles between the point measurement and maximum tempera-



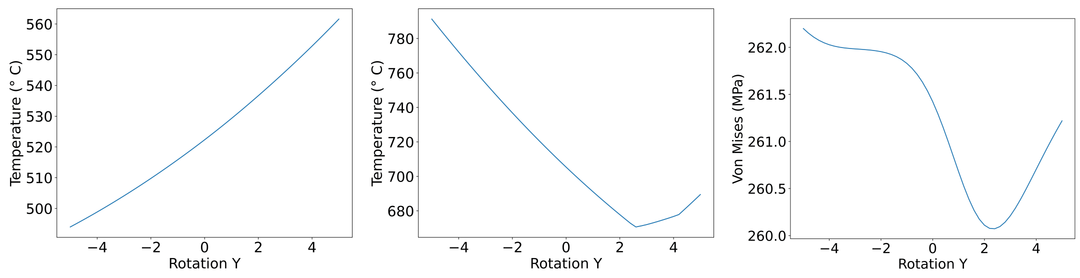
(a) Temperature (TC) (b) Temperature (max.) (c) Von Mises (max.)

Figure 7.14: Sensitivity to changes in y displacement (y_d)



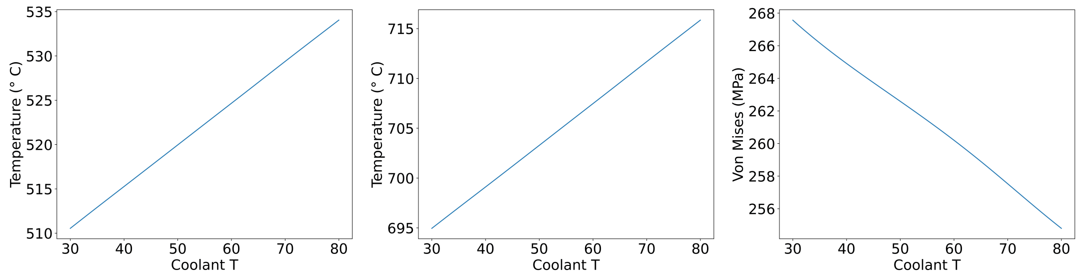
(a) Temperature (TC) (b) Temperature (max.) (c) Von Mises (max.)

Figure 7.15: Sensitivity to changes in z displacement (z_d)



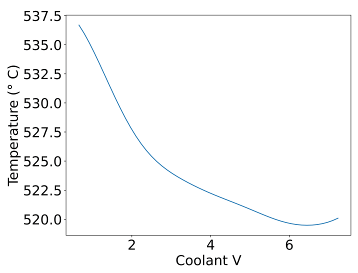
(a) Temperature (TC) (b) Temperature (max.) (c) Von Mises (max.)

Figure 7.16: Sensitivity to changes in rotation around y-axis (ϕ)

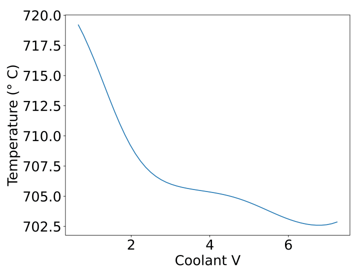


(a) Temperature (TC) (b) Temperature (max.) (c) Von Mises (max.)

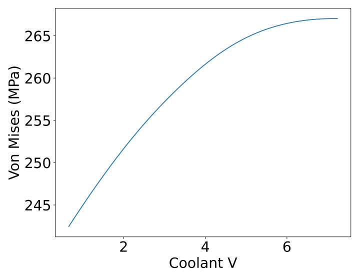
Figure 7.17: Sensitivity to changes in coolant temperature ($T_{coolant}$)



(a) Temperature (TC)

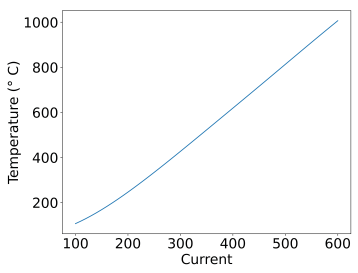


(b) Temperature (max.)

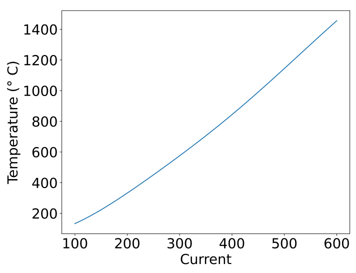


(c) Von Mises (max.)

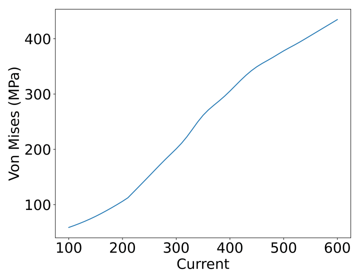
Figure 7.18: Sensitivity to changes in coolant velocity ($V_{coolant}$)



(a) Temperature (TC)



(b) Temperature (max.)



(c) Von Mises (max.)

Figure 7.19: Sensitivity to changes in coil current (I)

ture for certain variables, namely x_d, y_d and ϕ. It's also unsurprising that changes to z_d and I have a much bigger impact on the maximum temperature as opposed to the point measurement. While the coolant temperature and velocity have similar profiles to that seen previously, the effect that each one has on the maximum temperature is less balanced, with increasing $V_{coolant}$ having a much greater effect than decreasing $T_{coolant}$.

Interestingly, increasing the current by a factor of 6 - which leads to 36 times more power delivered to the sample - only leads to around an order of magnitude increase in the temperature. This is caused by the non-linearity of the problem, specifically the dependence of the pipe wall temperature on the amount of heat extracted by the coolant and the non-linear material properties used in the simulation.

This type of insight would take a significant amount of time for a new operator of HIVE to develop. Additionally, these relationships may be different for components with different geometries or made with different materials. These types of issues are avoided using this methodology, allowing insight to be gained in a much faster and more robust manner.

As HIVE's current setup has no way of measuring the mechanical performance of a component, there is no understanding of the effect that changing experimental parameters has on the stress state of a component. Using the surrogate model of the Von Mises stress field, the sensitivity of stresses with respect to the experimental parameters at any point in the component can be observed.

The first thing to note is the differing impacts that certain parameters have on the maximum temperature and maximum Von Mises stress of the component. For example, varying the value of y_d from -0.015 to 0.015 only has a modest impact on the maximum temperature, however this has a much greater impact on the Von Mises stress, see fig. 7.14c. Similarly, the coolant velocity, which had a bigger impact on the maximum temperature than the coolant temperature did, has a smaller impact than it in terms of the maximum Von Mises stress.

This is extremely useful, previously immeasurable, knowledge to the operators of HIVE. Having a thorough understanding of how the experiment parameters affect the temperature and stress at any point in the component can greatly help during a component's experimental campaign. Previously, understanding of only a limited part of this insight would have been available, and would have taken several months of testing to appreciate.

7.5 Alternative Component

To demonstrate this workflow and the power of *VirtualLab*, this same analysis is performed for a different component. The component in question was designed as part of the Additive Manufacturing Aiming towards Zero Waste and Efficient Production on High-Tech Metal Products (AMAZE) project [215], and consists of a copper pipe and block with a tungsten tile. This design has a block which is longer than the tile, as shown in fig. 7.20.

7.5.1 Surrogate Models

For this analysis 1400 simulations were performed, 700 each for coil A and coil B, with 500 of these observations used for training and the remaining 200 reserved for testing the model. The parameter space the data was collected in is the same as that used for the titanium sample, see table 7.1. 12 GPR FF surrogate models were trained, 6 for each coil design, 3 for the temperature field and 3 for the Von Mises field for models using the RBF, Matérn$_{5/2}$ and Matérn$_{3/2}$ covariance kernels.

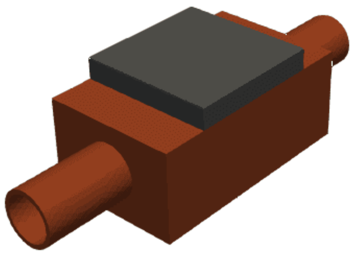


Figure 7.20: CAD of AMAZE component tested in HIVE

The 1400 simulations and surrogate models were generated using the sunbird cluster, where 100 cores were used to generate the data and perform the analysis in under three hours. This job was set to run overnight, meaning that the surrogate models were ready by the morning to assess their performance. The accuracy of the temperature surrogate models is shown in table 7.9, with the Von Mises surrogates shown in table 7.10.

Similarly to the titanium component, the RBF model is the most accurate with regard to the temperature field. Interestingly, the Von Mises surrogate models are much more accurate than for the titanium component (table 7.4), which is attributed to the lower reconstruction error for this data (6.21E-04) compared with the previous (1.81E-03, see fig. 7.3). For both coils, the performance of the RBF and Matérn$_{5/2}$ kernels are comparable.

	kernel	Test nRMSE	Train nRMSE
Coil A	RBF	3.04E-03	2.13E-03
	Matérn$_{5/2}$	3.41E-03	1.73E-04
	Matérn$_{3/2}$	4.10E-03	1.73E-05
Coil B	RBF	2.50E-03	2.01E-03
	Matérn$_{5/2}$	2.63E-03	1.94E-03
	Matérn$_{3/2}$	3.66E-03	1.94E-03

Table 7.9: nRMSE of temperature surrogate models of the AMAZE component.

	kernel	Test nRMSE	Train nRMSE
Coil A	RBF	6.24E-03	3.80E-03
	Matérn$_{5/2}$	6.36E-03	3.74E-03
	Matérn$_{3/2}$	7.28E-03	3.63E-03
Coil B	RBF	8.78E-03	5.03E-03
	Matérn$_{5/2}$	8.12E-03	4.45E-03
	Matérn$_{3/2}$	9.25E-03	4.38E-03

Table 7.10: nRMSE of the Von Mises stress surrogate models of the AMAZE component.

7.5.2 DoPE

Using the best performing surrogate models, the DoPE for the AMAZE component can be constructed. The desired experiments for this component are those outlined in section 7.3.3, however certain modifications have been made due to infeasible temperatures. As tungsten has a much lower electrical resistivity than titanium, much less heat is generated for this component. This is highlighted in table 7.11, where it is shown that the maximum temperature this component can reach for the given parameter space is 169.06 °C (experiment 1), over an order of magnitude less than for the titanium component. Interestingly, the maximum stress of this component (348.61 MPa, experiment 2) is not much less than the titanium component (512.98 MPa). This could be due to increased stresses due to different thermal expansion of the materials.

Satisfying experiment 3 is also possible, where again the coolant temperature is at the maximum temperature along with its velocity at the lowest value. Unlike the titanium component, z_d cannot be at its furthest point away from the component (6.00E-03) as this would not deliver the sufficient heating loads to the component to reach 100 °C within the pipe. Due to the lower maximum possible temperature of the component, the desired temperature for experiment 4 and 5 has been changed to 100 °C (from 300 °C and 400 °C respectively).

Exp.	y_d	y_d	z_d	ϕ	$T_{coolant}$	$V_{coolant}$	I	Anticipated result
1	-4.06E-03	-1.50E-02	3.00E-03	-5.00E+00	8.00E+01	6.58E-01	6.00E+02	A max. temperature of 169.06 °C
2	-3.06E-03	-6.88E-03	3.00E-03	5.00E+00	8.00E+01	6.58E-01	6.00E+02	A max. Von Mises stress of 348.61 MPa
3	-3.47E-03	-1.50E-03	5.77E-03	4.80E+00	8.00E+01	6.58E-01	5.01E+02	A max. temperature of 111.90 °C
4	3.54E-03	-1.50E-02	3.00E-03	-5.00E+00	3.00E+01	3.61E+00	6.00E+02	A max. Von Mises stress of 82.27 MPa
5	-3.81E-03	3.02E-03	6.00E-03	5.00E+00	6.74E+01	1.16E+00	5.36E+02	A max. temperature of 100 °C

Table 7.11: DoPE for AMAZE component and anticipated outcome.

7.6 Chapter Summary

This chapter has demonstrated that the use of FF surrogate models can provide a much more comprehensive DoPE for a component in HIVE compared with the current approach. Previously, experimental parameters were estimated based on the sparse thermal experimental data of the previous experiment, however by using surrogate models it is possible to identify the parameters which deliver a certain temperature and/or stress to any part of the component. Along with being faster, more robust and accurate, this method also reduces the reliance on operator's previous experience. For example, it would have taken several attempts for a HIVE operator to learn the difference in the quantity of power a tungsten component requires in comparison with a titanium one, however by using surrogate models this is achieved without the need to perform a single physical experiment.

The automated workflow developed using *VirtualLab* means that this entire analysis can be performed in a matter of hours, which is hugely beneficial for HIVE to test components in a high throughput manner. This also removed the need for inefficient trial & error approaches to estimating experimental parameters, something which is not only time-consuming but also costly. Coupling in this reduced testing time with the smarter DoPE means that experiments in HIVE provide much greater understanding of a component's strengths and limitations, thus enabling more informed decisions to be made during the next iteration of the design cycle.

This chapter also highlighted the alternative ways in which the generated surrogate models can be used to gain insight. Section 7.4 highlighted that the sensitivity of parameters learnt from simulated thermocouple data, which is the only measure which HIVE operators currently have, are often underestimated compared with the maximum temperature of the component. Moreover, the Von Mises stress surrogate model was able to provide parameter sensitivity for a metric which HIVE operators currently have no method of inferring. Previously, understanding of only a limited part of this insight would have been available, and would have taken several months of testing to appreciate.

The contents of the *VirtualLab* run files necessary to create the temperature and Von Mises surrogate models and perform the inverse analysis can be found in listings A.2 and A.3 in the appendices.

Chapter 8

Enhancing Sparse Experimental Data

8.1 Introductory Remarks

Currently, HIVE has no method of using the sparse experimental data to infer more insightful properties about a component, such as the maximum temperature of the component or its stress state. As a result, only limited understanding of a component's suitability is currently gained from each experiment. As outlined in section 1.2.2, many problems in engineering which use proxy measurements require highly controlled environments and geometries, however given the complexity of the HIVE experiment and variations in component geometry this is not possible. This chapter demonstrates a novel approach to solving the proxy measurement problem through the use of surrogate models, enabling a limited amount of pointwise temperature data to be used to infer the temperature and stress field throughout the component. This knowledge enables far more informed decisions to be made during the design cycle of the component, thus speeding up their development (due to fewer number of iterations required).

As outlined in section 4.3.3, the diagnostics in HIVE are limited, with only a few thermocouples available to collect pointwise surface temperature data, therefore it is essential that these provide as much insight as possible. This chapter also investigates the optimisation of sensor placement in HIVE, which is a novel contribution to the IHTP field as outlined in the work of Tamaddon-Jahrromi *et al.* [23].

The outline of this chapter is as follows; firstly, section 8.2.1 will provide a proof of concept of the ability to use thermocouple data to infer the temperature and stress field throughout the component. Following this, section 8.2.3 uses gradient-free optimisation algorithms to reduce the number of thermocouples required by identifying their optimal location. Following this, section 8.3 will show that the thermocouple data can also be used to estimate uncertain properties of the component. Here, the thermal contact conductance between two bonded surfaces is investigated, where it is shown that encoding prior knowledge of the experiment in the methodology can substantially improve its accuracy.

8.2 Thermocouple Informed Temperature Field

In certain circumstances, the temperature at thermocouple locations extracted from the simulation won't equate to those recorded by the experiment. This could be due to differences in the component performance resulting from its manufacture, e.g. defects, different material properties, or errors in the simulation due to the use of incorrectly estimated parameters. Even in these circumstances, the full temperature field throughout the component can still be inferred by using the inverse solution given by eq. 8.1 and using it as the input in to the computational (or surrogate) model. In this equation, $T_{exp}^{TC^{(i)}}$ is the data recorded by the i-th thermocouple during the experiment ($i = \{1, .., N_{TC}\}$) and $T_{surr}^{TC^{(i)}}$ is the temperature predicted at the location of that thermocouple using the surrogate model.

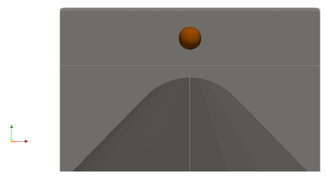
$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{P}} \sum_{i=1}^{N_{TC}} (T_{surr}^{TC^{(i)}}(\mathbf{x}) - T_{exp}^{TC^{(i)}})^2 \tag{8.1}$$

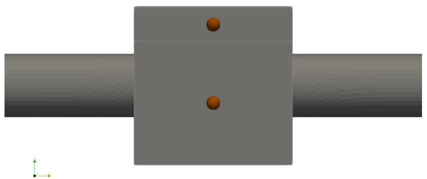
As discussed in section 5.3.5, thermocouples could not be bonded to the surface of the components, meaning that this experimental data is unavailable. Instead, the values for T_{exp}^{TC} are extracted from the temperature field generated by a simulation, i.e. using simulated experimental data. The advantage of this approach is that the temperature field which this method is trying to predict is known (from the simulation), meaning that its accuracy can be measured. This would not have been possible using actual experimental data. Also, simulated experimental data is faster and easier to produce, and enables a wide variety of different loading conditions from across the parameter space to be tested.

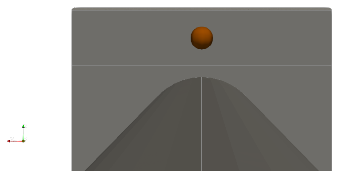
The value of $T_{surr}^{TC^{(i)}}$ is calculated as a linear combination of the temperatures at the nodes of the surface element which the thermocouple is located within. Therefore, given that the outputs of T_{surr} are differentiable with respect to its inputs, then it follows that T_{surr}^{TC} are similarly differentiable. As a result, eq. 8.1 can be solved using the *MultiSLSQP* package.

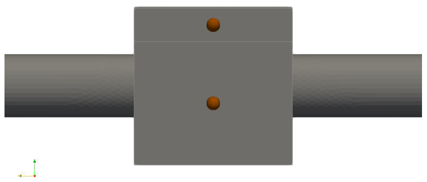
8.2.1 Proof of Concept

In this proof of concept, 7 thermocouples are attached to 7 different surfaces of the component. As previously discussed, thermocouples can't be placed on the coil adjacent surface, and the two surfaces of the block where the pipe intersects are also unsuitable due to difficulty in joining thermocouples there. Therefore, thermocouples are joined to the mid-point of the 4 surfaces that make up the sides of the tile along with 3 surfaces of the block, 2 on the front and back facing sides and 1 on the base, see fig. 8.1.


(a) $TC^{(1)}$


(b) $TC^{(2)}$ and $TC^{(5)}$


(c) $TC^{(3)}$


(d) $TC^{(4)}$ and $TC^{(6)}$

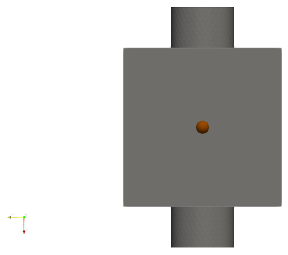

(e) $TC^{(7)}$

Figure 8.1: Location of 7 thermocouples on the surface of a component.

	Surface	**Positioning parameters**
$TC^{(1)}$	Tile Side 1	(0.5,0.5)
$TC^{(2)}$	Tile Side 2	(0.5,0.5)
$TC^{(3)}$	Tile Side 3	(0.5,0.5)
$TC^{(4)}$	Tile Side 4	(0.5,0.5)
$TC^{(5)}$	Block Side 2	(0.5,0.5)
$TC^{(6)}$	Block Side 4	(0.5,0.5)
$TC^{(7)}$	Block Base	(0.5,0.5)

Table 8.1: Description of 7 thermocouple locations.

A description of the thermocouple placement used in this example is given in table 8.1, where each location is described by the surface it is attached to and its positioning on that surface. This positioning is described using 2 parameters over the range [0,1], one for each direction of the 2D surface, with (0.5,0.5) placing the thermocouple at the centre of its respective surface. The origin for each surface is the corners closest to the origin of the coordinate system.

A simulation is performed using the experimental parameters given in table 8.2, with the resulting temperature field shown in fig. 8.2. The temperatures at the 7 thermocouple locations are extracted from this simulation and are given in table 8.3.

Parameter	**Simulation**
x_d	-6.83E-04
y_d	1.32E-02
z_d	5.45E-03
ϕ	-1.64E+00
$T_{coolant}$	4.16E+01
$V_{coolant}$	1.21E+00
I	4.02E+02

Table 8.2: Experimental parameters used for testcase.

Using the data given in table 8.3 as T_{exp}^{TC} in eq. 8.1 yields several sets of viable experimental parameters, each resulting in the desired temperature at the thermocouple location, highlighting the non-uniqueness of inverse problems. Because of correlation between certain input parameters, there are a number of different combinations of experimental parameters which will yield the exact same temperature profile. For example, increasing the thermal loading effect by moving the coil closer to the component can be offset by reducing the power delivered to the induction heating system.

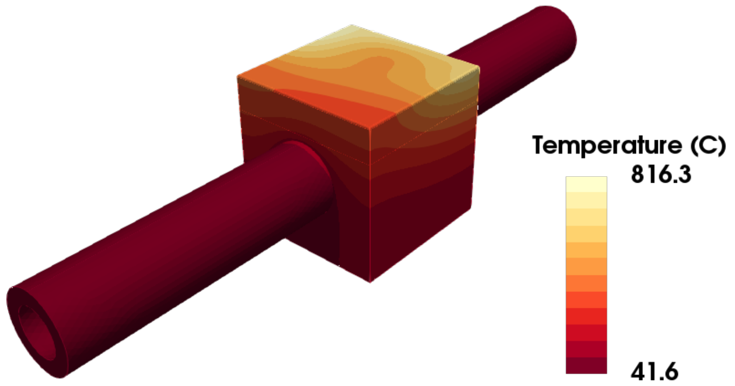


Figure 8.2: FE simulated temperature profile using parameters in table 8.2

	Temperature (°C)
$T_{exp}^{TC^{(1)}}$	331.46
$T_{exp}^{TC^{(2)}}$	418.23
$T_{exp}^{TC^{(3)}}$	664.84
$T_{exp}^{TC^{(4)}}$	493.90
$T_{exp}^{TC^{(5)}}$	203.27
$T_{exp}^{TC^{(6)}}$	203.42
$T_{exp}^{TC^{(7)}}$	201.96

Table 8.3: Simulated experimental data.

As it's the predicted field which is of interest and not the experimental parameters themselves, only those which give distinct temperature profiles are retained as inverse solutions. For this work, two temperature fields are considered distinct if the average percentage difference between the temperature values at the nodes of each is greater than 2.5%. This metric has been devised by observing different temperature profiles and analysing their level of difference. As a result of this, there are 4 distinct temperature profiles which match the recorded thermocouple data, all of which are shown in fig. 8.3 along with their error from the desired temperature profile (which was shown in fig. 8.2). The experimental parameters to achieve these four profiles are given in table 8.4.

These have been generated by using the surrogate and simulated experimental data to identify the 7 admissible BCs, which are used to generate full temperature field profiles for each using the surrogate model. From these, only distinct temperature profiles are retained along with the BCs needed to achieve them.

Parameter	**Profile 1**	**Profile 2**	**Profile 3**	**Profile 4**
x_d	-6.89E-04	-6.81E-04	-7.29E-04	-1.57E-03
y_d	1.31E-02	1.30E-02	1.32E-02	1.33E-09
z_d	5.09E-03	4.97E-03	5.37E-03	5.91E-03
ϕ	-1.55E+00	-1.53E+00	-1.46E+00	3.18E-01
$T_{coolant}$	3.93E+01	3.18E+01	4.96E+01	5.90E+01
$V_{coolant}$	1.10E+00	8.20E-01	2.04E+00	6.95E+00
I	3.89E+02	3.85E+02	4.01E+02	4.23E+02

Table 8.4: Experimental parameters used for testcase.

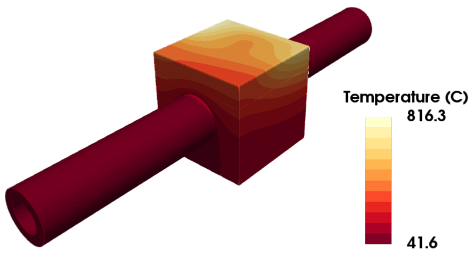
In terms of the temperature profile through the block and tile of the component, all four profiles are very similar. The reason that these are considered distinct profiles is because of the different temperatures along the pipe, caused by the different coolant temperatures estimated by the optimiser. The value of the coolant ranges from 31.8 °C to 59 °C, with these differences offset by changes to the coolant velocity, meaning that the same quantity of energy would be extracted from the component.

As discussed in section 4.3, the coolant parameters are measured at the inflow to the vacuum vessel, therefore there is a high level of confidence about what the temperature in the pipe actually is. This can be encoded in to the search for admissible temperature fields be fixing the value of the coolant temperature to that used to generate the experimental data during the optimisation routine. This means that in this example, $T_{coolant}$ was assumed to be 41.6 °C (see table 8.2) and wasn't able to change while searching for the inverse solution given by eq. 8.1. This idea will be discussed in more detail in section 8.2.4.

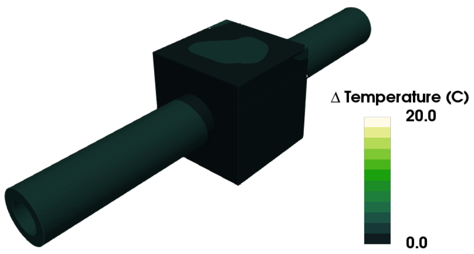
Adding in this extra criteria reduces the number of admissible solutions to one, which is shown in fig. 8.4. There is very good agreement between the predicted temperature field and the desired temperature field (shown in fig. 8.2), with a maximum absolute error of less than 3 °C, equating to an error of less than 0.5%.

The Von Mises stress profile based on these inversely informed experimental parameters is shown in fig. 8.5 alongside the stress profile predicted using the FE model and the absolute error difference between them. Again, excellent agreement is shown, with a maximum error of less than 1%.

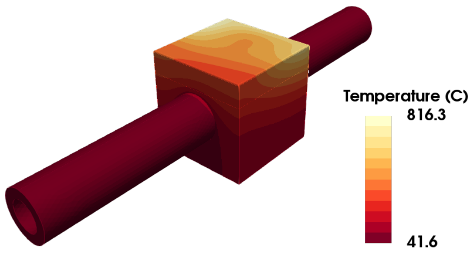
This demonstrates that this method can take sparse experimental data collected by HIVE and use it to inform the temperature and stress profile throughout the component. This provides substantially more insight and understanding of the component, helping inform better decision-making during its design cycle.



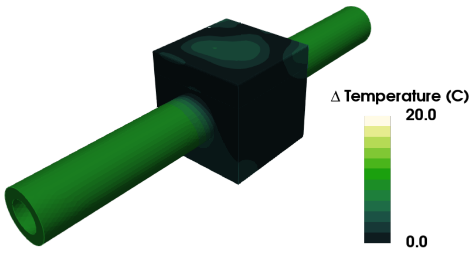
(a) Temperature profile 1



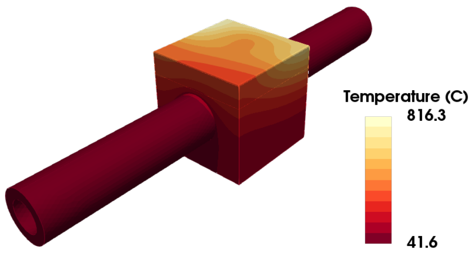
(b) Absolute difference for profile 1



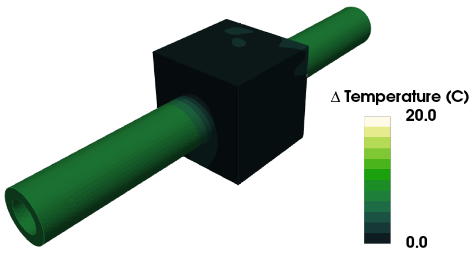
(c) Temperature profile 2



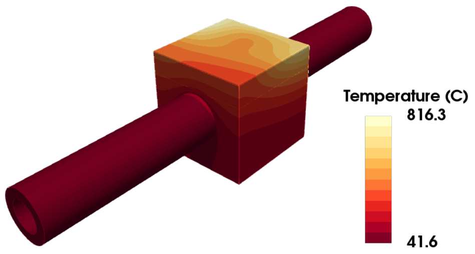
(d) Absolute difference for profile 2



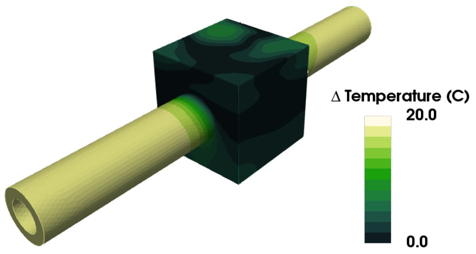
(e) Temperature profile 3



(f) Absolute difference for profile 3

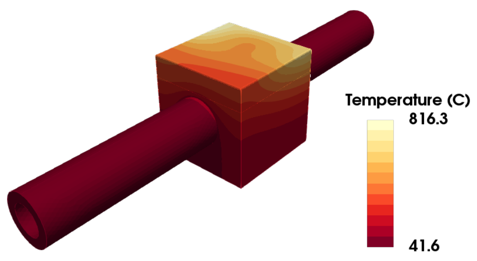


(g) Temperature profile 4

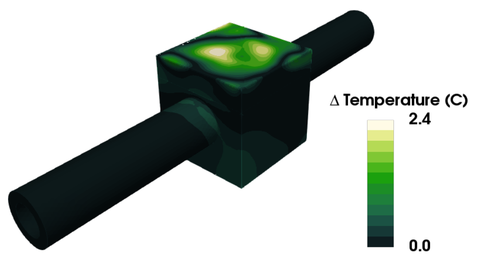


(h) Absolute difference for profile 4

Figure 8.3: Different admissible temperature fields predicted by the GPR surrogate model using 7 thermocouple measurements and their error from the FE model (fig. 8.2)

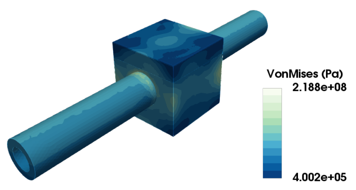


(a) Temperature profile

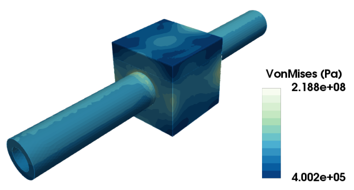


(b) Absolute difference from FE model

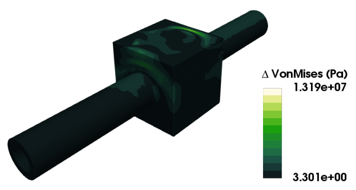
Figure 8.4: The only admissible temperature field predicted by the GPR surrogate model using 7 thermocouple measurements and known $T_{coolant}$ along with its error from the FE model (fig. 8.2)



(a) FE model



(b) Surrogate model



(c) Absolute difference

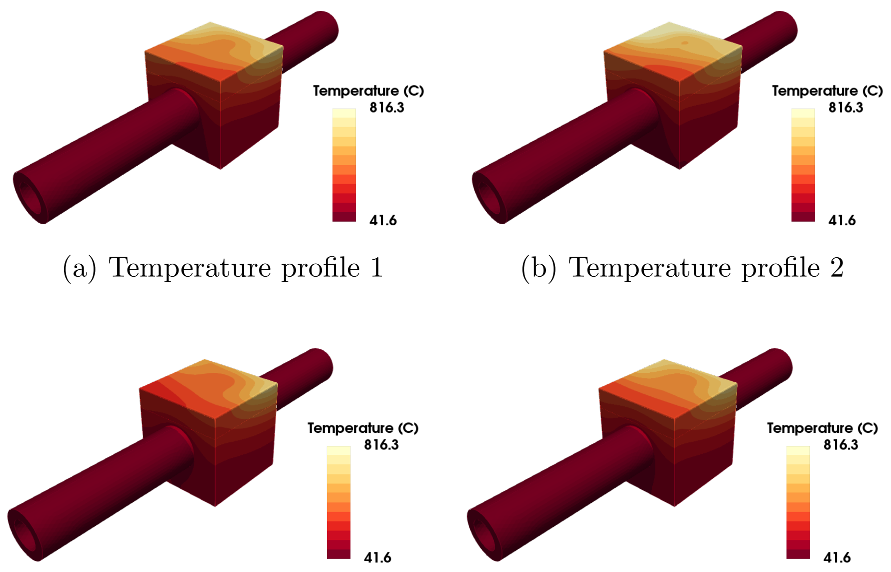
Figure 8.5: Comparison between the FE model and inversely informed surrogate model for the Von Mises stress profile using seven temperature measurements.

8.2.2 Thermocouple Sensitivity

As joining thermocouples to the component is a time-intensive task, it is beneficial to reduce the number required. Doing so not only saves money by freeing up the time of the HIVE operators for other tasks, but also enables a higher throughput of component testing, thus speeding up the design cycle of the component in question.

Consider the same example as the previous section but with only 3 thermocouples; $TC^{(2)}$, $TC^{(5)}$ and $TC^{(7)}$. With fewer known temperature values from thermocouples, there are more admissible temperature fields which fit the data. In this case, there are 5 different temperature profiles which match the data recorded at the thermocouple locations, four of which are shown in fig. 8.6. On the other hand, choosing $TC^{(2)}$, $TC^{(3)}$ and $TC^{(6)}$ results in fewer (3) admissible temperature fields.

Clearly this is just a single testcase, however to gain an unbiased measure of how informative a set of thermocouples are for informing the full temperature



(a) Temperature profile 1 (b) Temperature profile 2

(c) Temperature profile 3 (d) Temperature profile 4

Figure 8.6: Example of four different admissible temperature profiles when only three thermocouples are used.

field a larger number of testcases will need to be considered. The suitability of a configuration of thermocouples, denoted as $C = \{TC^{(1)}, ..., TC^{(N_{TC})}\}$, is calculated using eq. 8.2, where M_i is the number of admissible temperature fields, which is averaged over 10 randomly selected scenarios from across the parameter space.

$$L_{TC}(C) = \frac{1}{10}\sum_{i=1}^{10} M_i(C) \tag{8.2}$$

Using this definition, for $C = \{TC^{(2)}, TC^{(5)}, TC^{(7)}\}$ the value of L_{TC} is 5.6, while for $C = \{TC^{(2)}, TC^{(3)}, TC^{(4)}\}$ this value is 3.3.

8.2.3 Sensor Placement Optimisation

The sensitivity of the placement of thermocouples to the number of temperature fields naturally raises the question of their optimal configuration. To achieve this, a thermocouple configuration C^* must be identified which minimises the value of L_{TC}. Clearly, this objective function is not differentiable, meaning that solving this optimisation problem will require using the gradient-free genetic algorithm

To enable the genetic algorithm to be used in this context, the configuration of the thermocouples must be encoded as a genetic sequence. The positioning of a thermocouple is characterised by the name of the surface and two parameters relating to its placement on said surface. The latter two of these do not require encoding as these are continuous values in the range from 0 to 1, however the 7 surfaces must be represented as integer values instead of strings. With reference to table 8.1, 'Tile side 1' is encoded as 1 up to 'Block base' encoded as 7. This means that the combination of the three thermocouples given in table 8.5 would be encoded as (2, 0.8, 0.2, 6, 0.1, 0.1, 7, 0.4, 0.9)

	Surface	**Positioning parameters**
$TC^{(1)}$	Tile Side 2	(0.8,0.2)
$TC^{(2)}$	Block Side 4	(0.1,0.1)
$TC^{(3)}$	Block Base	(0.4,0.9)

Table 8.5: Example of three thermocouple locations.

In genetic algorithms, the evaluation of the loss function for a member of the population is usually termed as their fitness value. For this problem, the optimal fitness value is 1, giving a natural lower bound to the optimisation algorithm.

The hyperparameters of the genetic algorithm are summarised in table 8.6. A population size of 30 is chosen, as this is mid-point of the range of 20 to 40 advised by Grefenstette in [216]. The 6 fittest members of the population are used to produce the offspring of the next generation, with this value chosen as 20% of the entire population size. Choosing to carry the parents forward to the next generation means that 24 new offspring will be produced for each new generation.

Population size	30
Crossover probability	1
Mutation probability	0.05
Number of parents retained	10
No. generations	50

Table 8.6: Genetic algorithm parameters used for analysis.

Since the parents genes will be retained in the next generation, the crossover probability is chosen as 1 to ensure that all offspring will differ from their parents. This is supported by Grefenstette, stating that population sizes such as these benefit from high crossover probabilities. The advised mutation probability of 0.05 is used, meaning that for a genetic sequence of 9 chromosomes like the above, there is a 37% $(1 - (1 - 0.05)^9)$ probability of mutation in an offspring.

The genetic algorithm is performed for a maximum of 50 generations, however if a fitness value of 1 is identified for a member of the population, or if there is no improvement in the best fitness value for 10 generations, the algorithm will terminate to reduce computational expense.

A number of different attempts of this optimisation algorithm were performed with alternatively seeded initial population, with best scoring combination given in table 8.7, with their placement on the component shown in fig. 8.7. The best value for the objective (fitness) function (eq. 8.2) is 1.4, with the evolution of the best fitness value with each generation shown in fig. 8.8

	Surface	**Positioning parameters**
$TC^{(1)}$	Tile Side 1	(0.7195, 0.5077)
$TC^{(2)}$	Tile Side 3	(0.8173, 0.2861)
$TC^{(3)}$	Tile Side 3	(0.2269, 0.5513)

Table 8.7: Optimal combination of three thermocouples.

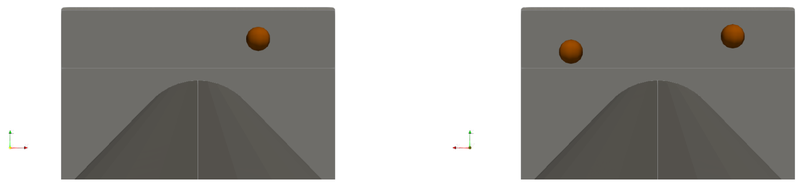


Figure 8.7: Visualisation of the 3 optimally placed thermocouples.

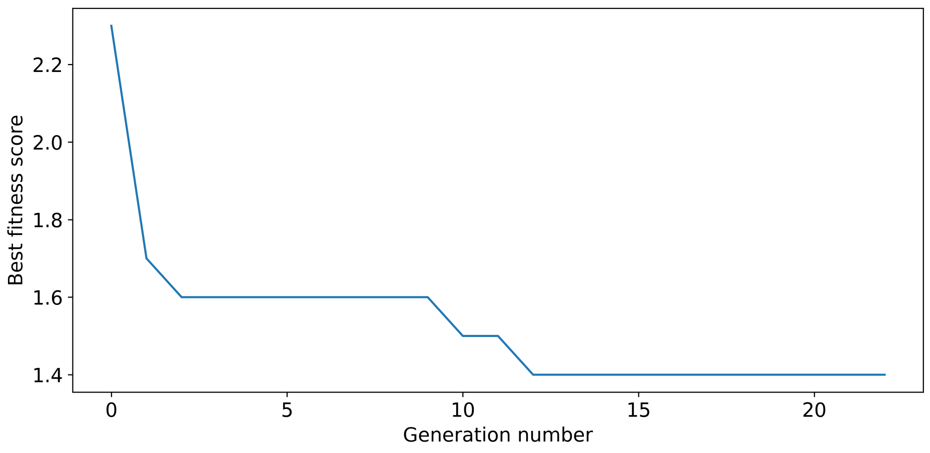


Figure 8.8: Evolution of fitness value with each generation

Although this is an optimal configuration, a fitness value of more than 1 means there are certain cases where more than 1 distinct temperature field was identified. As a result, using only 3 thermocouples to record data is too few a number. Performing the same optimisation routine again for the placement of 4 thermocouples results in a number of combinations which deliver the desired fitness value of 1, one of which is summarised in table 8.8. In scenarios where there are a number of combinations which perform equally well, then it is sensible to choose the combination which is the most logistically feasible to achieve.

	Surface	**Positioning parameters**
$TC^{(1)}$	Tile Side 1	(0.6007, 0.8659)
$TC^{(2)}$	Tile Side 3	(0.7311, 0.1611)
$TC^{(3)}$	Tile Side 4	(0.9835, 0.0794)
$TC^{(4)}$	Block Base	(0.8070, 0.3944)

Table 8.8: Optimal combination of four thermocouples.

It's important to highlight that there is a degree of sensitivity to the placement of the parameters, especially for the case when three thermocouples are used. Adding an element of Gaussian noise (mean zero and standard deviation 0.02) to the placements of the optimal locations results in an increase in the number of admissible fields from the optimal 1.4 to 1.8. Performing the same analysis for 4 thermocouple case would still result in an optimal score of 1, however.

8.2.4 Including Prior Knowledge

Until this point, the only assumed knowledge has been the temperature at the N_{TC} thermocouple locations and the temperature of the coolant. It was shown in section 8.2.1 that adding the knowledge of the $T_{coolant}$ had a big impact in terms of the number of admissible temperature fields.

There are certain parameters which HIVE operators have a high level of confidence for, such as the temperature of the coolant, while there are others which are harder to accurately measure, such as the coil positioning. This knowledge can be encoded in the search for an inverse solution by limiting the range where admissible solutions are sought. These ranges are decided based on the level of confidence the HIVE operator has about the experimental parameter at hand. If an operator believes the value for the i-th parameter, p_i, is around t_i, then solutions for this parameter can be sought in the range $[t_i - r_i, t_i + r_i]$, where the value of r_i is decided based on the level of confidence. The range $[t_i - r_i, t_i + r_i]$ is a subspace of the global range of admissible solutions, denoted as $[p_i^L, p_i^U]$, where p_i^L and p_i^U are the upper and lower bound of the parameter space for p_i, respectively.

In this work, the value of r_i is defined using eq. 8.3, where c_i is the level of confidence the operator has about the value of the parameter. For example, consider the scenario where the HIVE operator believes the value for z_d is 0.004 and is 70% confident about it. Given that the size of the parameters space is 0.003, then the range in which solutions are sought for z_d during the optimisation step is [0.0031,0.0049] instead of [0.003,0.006]. If the confidence level was 50% then the range would be [0.003,0.0055], since the lower bound of 0.0025 is smaller than the lower bound of the parameter space.

$$r_i = (p_i^U - p_i^L)(1 - c_i) \tag{8.3}$$

A set of example levels of confidence for each of the experimental parameters based on knowledge about the operation of HIVE is given in table 8.9. Due to the scale of the dimensions and orientations relating to the coil configuration, accurate measurements are difficult, meaning that these four parameters are assigned 70% confidence. The 100% confidence for $T_{coolant}$ is equivalent to fixing the value during the optimisation routine, since r_i in eq. 8.3 would be zero, as was outlined in section 8.2.1. The coolant velocity is recorded by the cooling system, however since this is located further away from the vacuum vessel, there is a little more uncertainty regarding its accuracy. Since the Rogowski coil measures the current on the outside of the vacuum vessel, a lower confidence is prescribed to account for any potential losses which may occur.

	Confidence
x_d	70%
y_d	70%
z_d	70%
ϕ	70%
$T_{coolant}$	100%
$V_{coolant}$	80%
I	80%

Table 8.9: Confidence levels for experimental parameters.

Using the optimal configuration of 3 thermocouples from section 8.2.3 (table 8.7) along with these levels of confidence results in a L_{TC} score of 1. Furthermore, even when adding Gaussian noise the placement values at the same level of 0.02 the number of admissible fields is still 1. Not only is adding this confidence to the framework useful in that it reduces the number of thermocouples required, but also allows for more variability in terms of their positioning.

A summary of the improvement in the objective function, L_{TC}, through the use of the genetic algorithm and including prior knowledge of the experiment, is highlighted in fig. 8.9.

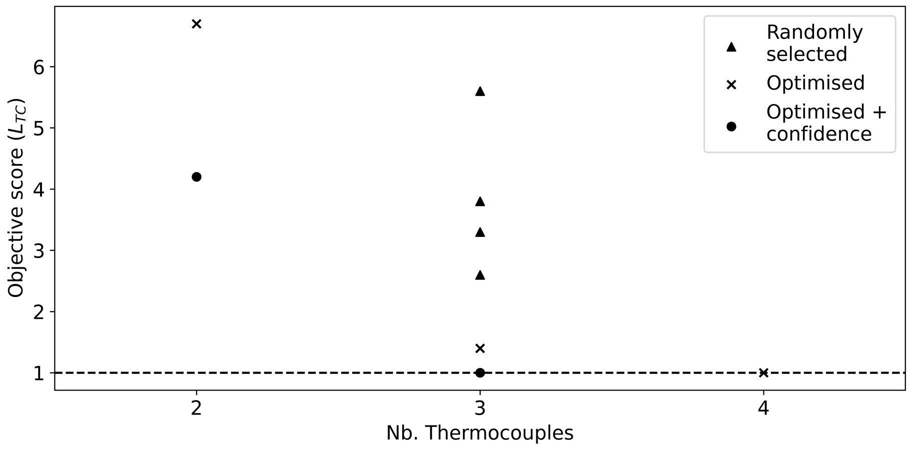
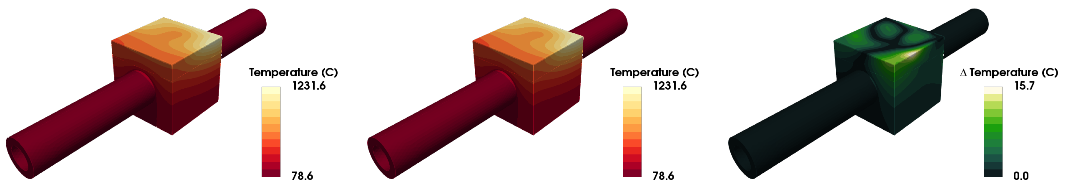


Figure 8.9: Improvement in admissible field objective function L_{TC} through the use of optimisation and inclusion of confidence.

To highlight this methodology, a FE simulation is performed where the temperature values at the three thermocouples (positions given in table 8.7) are extracted and are given in table 8.10. These are used along with the confidence about the system to calculate an inverse solution, which is passed to the surrogate models for the temperature and Von Mises stress, which are shown in fig. 8.10 and 8.11 respectively.

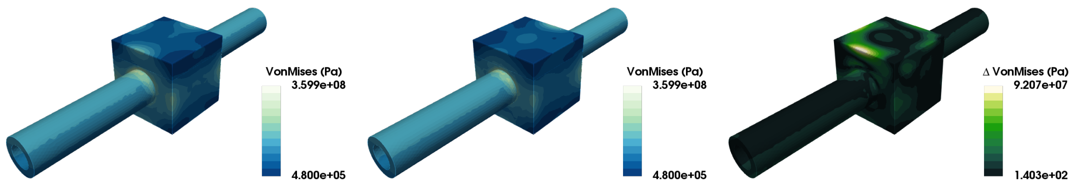
	Temperature (°C)
$T_{exp}^{TC^{(1)}}$	549.54
$T_{exp}^{TC^{(2)}}$	896.17
$T_{exp}^{TC^{(3)}}$	953.97

Table 8.10: Simulated experimental thermocouple data at locations given in table 8.7.



(a) FE model (b) Surrogate model (c) Absolute difference

Figure 8.10: Comparison between the FE model and inversely informed surrogate model for the Temperature profile using only three temperature measurements and additional prior knowledge of the experiment.



(a) FE model (b) Surrogate model (c) Absolute difference

Figure 8.11: Comparison between the FE model and inversely informed surrogate model for the Von Mises stress profile using only three temperature measurements and additional prior knowledge of the experiment.

There is a high degree of accuracy in terms of the temperature profile, with a maximum error of 2%, meaning that this can still be accurately predicted using as few as 3 thermocouples. There are slightly larger errors for the Von Mises stress field compared with that shown in section 8.2.1, meaning that to accurately predict this a larger number of thermocouples may still be required. In this example, however, the maximum Von Mises stress of the component is still accurately predicted, with 350 MPa using the inversely informed surrogate compared with the 359 MPa predicted by the simulation.

8.2.5 Optimisation with Constraints

This implementation of the thermocouple optimisation makes it extremely easy to place constraints on not only what surfaces the thermocouples can be attached to, but also where on that surface they can be placed. For example, consider a scenario where thermocouples can't be attached to the tile. This means that there are only three available surfaces where the thermocouples can be attached; Block Side 2, Block Side 4 and Block Base. Furthermore, it's preferable for the thermocouples not to be placed too close to the edges of the surface, therefore the range for the two placement parameters are limited to the range [0.2,0.8], instead of the range [0,1] used in the previous example.

Performing the optimisation routine subject to these constraints for the three thermocouple case without using results in the configuration given in table 8.11, highlighting that the constraints are satisfied with each positioning parameter within the designated range. This combination yields a fitness score of 2.4, which is, as expected, greater than the fitness score of 1.4 achieved for the unconstrained problem (section 8.2.3) . This highlights the importance of collecting data from the tile, where the larger thermal loads are located.

	Surface	**Positioning parameters**
$TC^{(1)}$	Block Side 2	(0.6179, 0.3717)
$TC^{(2)}$	Block Side 2	(0.3254, 0.4539)
$TC^{(3)}$	Block Side 4	(0.2448, 0.5308)

Table 8.11: Optimal combination of three thermocouples, subject to constraints.

For the 4 thermocouple problem, it is still possible to identify a combination of thermocouples which yield a fitness score of 1 subject to these constraints. The placement of these 4 thermocouples are summarised in table 8.12, and are also visually represented in fig. 8.12.

	Surface	**Positioning parameters**
$TC^{(1)}$	Block Side 2	(0.7510,0.2548)
$TC^{(2)}$	Block Side 2	(0.4782,0.5013)
$TC^{(3)}$	Block Side 4	(0.3450,0.2573)
$TC^{(4)}$	Block Base	(0.3882,0.2284)

Table 8.12: Optimal combination of four thermocouples, subject to constraints.

This is an extremely powerful and useful method of identifying the most efficient locations to gather data. Rather than depending on previous experience and selecting similar thermocouple locations, this methodology can identify an optimal arrangement on a case by case basis, greatly improving the efficiency of the data collected and ultimately the insight from each experiment.

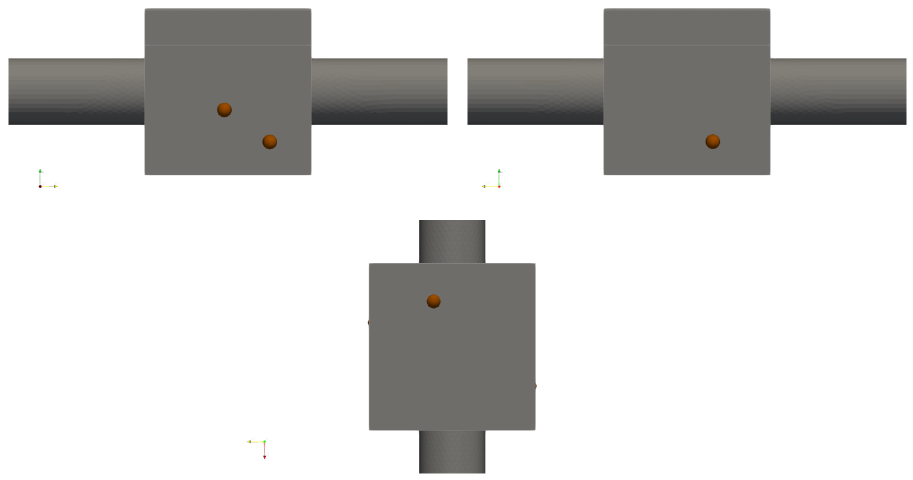


Figure 8.12: Visualisation of the 4 optimally placed thermocouples, subject to constraints.

8.3 Estimating Unknown Quantities

The focus of the work thus far has been on using surrogate models to identify experimental parameters for collecting the most valuable data. However, it can also be used to identify values for uncertain parameters. For example, when performing a simulation, a number of parameters used in the model may be assumed values taken from literature. One example of this are the material properties used in FE analysis. The properties of a material are evaluated experimentally using material science experiments, however due to subtle differences in their granular structure, the material properties calculated for each specimen will be different due to the treatment of discrete mechanics as continuous mechanics through homogenisation. To remove this variability, the material properties are calculated as the average value over many tested specimens, resulting in a statistically most likely value.

Prescribing a single value to parameters such as these can lead to inaccuracies between the simulation and the real world outcome. This section will investigate how surrogate models can be used to predict the values of uncertain parameters from sparse experimental data. The focus here will be on estimating a parameter with a high degree of uncertainty; the thermal contact conductance of a bonded surface.

8.3.1 Thermal Contact Conductance

Thermal contact conductance is the study of heat conduction between thermally variable bodies in contact with one another. When two solid bodies are in contact, heat will transfer from the hotter body to the colder one. If the bond between these two bodies is perfect, then the temperature on the interface of one body would be the exact same as that on the interface of the second. In practice, bonds between surfaces are never perfect, meaning there is a discontinuity at the interface, as shown in fig. 8.13, where q is the heat flux through the interface.

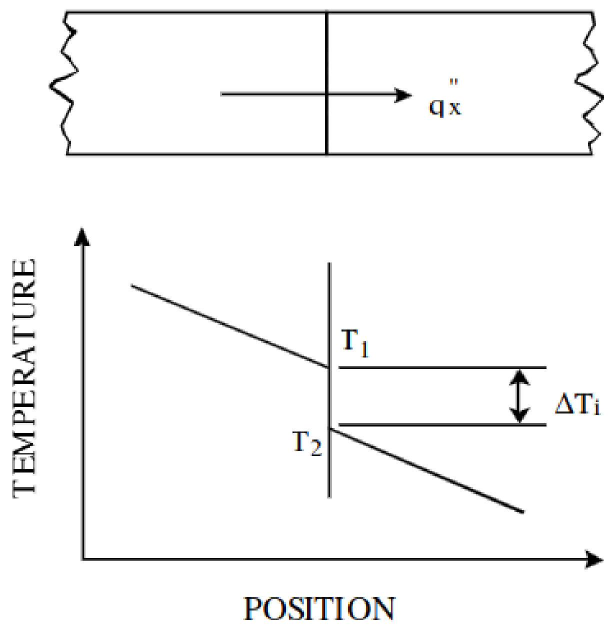


Figure 8.13: Effect of thermal contact conductance for heat in a 1D bar [217].

This imperfect heat transfer is caused by surface roughness, which is microscopic asperities and depressions that deviate from an otherwise smooth surface, see fig. 8.14. This means that the actual area of contact is much smaller than what is perceived. Along with the surface roughness, the actual contact area depends on the pressure between the contacting parts (which is the result of the method used to join the parts), the hardness of the materials and the temperature.

The thermal contact conductance coefficient (TCCC), denoted as h_c, indicates the heat conductivity which takes place between two contacting bodies and is related to the heat flux and temperature drop, see eq. 8.4. A TCCC value of zero indicates no transfer of heat across the contact, while as $h_c \to \infty$ means the contact behaves more and more like a perfect heat exchange with $\Delta T_{contact} \to 0$.

$$q = h_c A_{contact} \Delta T_{contact} \tag{8.4}$$

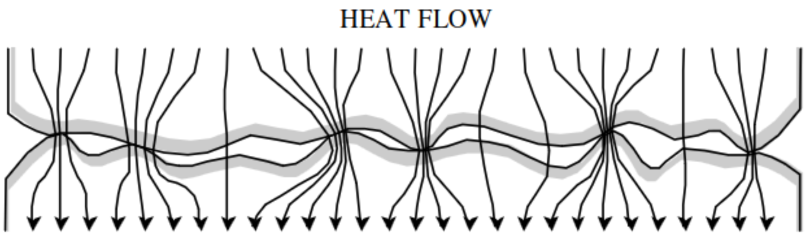


Figure 8.14: Restriction of heat flow through an interface formed by two surfaces [217].

Modelling the impact of contact surfaces is crucial when dealing with materials that have a high thermal conductivity, such as metals. Consider the 1D problem, where a bar of length L has its two ends heated and fixed to 20°C and 100°C, respectively. The bar is made up of two smaller bars, both of which have length $l = 0.5$, thermal conductivity $k = 1$ and a continuous cross-sectional area, A. Solving this as a 1D FE problem, the temperature on the two interfaces, T_2 and T_3 are given by eq. 8.5, highlighting the link between the thermal conductivity and TCCC. This also demonstrates that if $h_c = 0$, then $T_2 = T_A$ and $T_3 = T_B$, while as $h_c \to \infty$ $T_2 \to T_3 \to (T_A + T_B)/2$.

$$\begin{aligned} T_2 &= \frac{\frac{k}{h_c l} T_A + (T_A + T_B)}{\frac{k}{h_c l} + 2} \\ T_3 &= \frac{\frac{k}{h_c l} T_B + (T_A + T_B)}{\frac{k}{h_c l} + 2} \end{aligned} \tag{8.5}$$

The majority of components tested in HIVE will have contact surfaces, either between the tile and the block, the block and the pipe, or both. In addition to this, there are numerous novel conceptual designs being investigated not discussed in this work which include additional features. Given that the components tested in HIVE typically have relatively high thermal conductivities, accurately modelling the effects of the contact surfaces is essential. While TCCC values are available in literature, these are generally for popular materials under specific, well characterised conditions. For example, while the TCCC value for a bond between copper and copper can be found, it is unavailable for less commonly used materials, such as a tungsten or beryllium. Along with this, the manufacturing process for fusion relevant components may not necessarily be comparable to those in other industries.

A number of correlations to estimate the value of h_c have been defined in literature [217]. Since the TCCC depends on a variety of factors, however, these correlations are complex and composed of a number of quantities. Many of these quantities, such as the surface roughness, the micro hardness of materials and the contact pressure, are hard to quantify, especially for novel materials or bonding methods, or have a large degree of variability, making the estimating of TCCC value for a specific bond within a component extremely difficult.

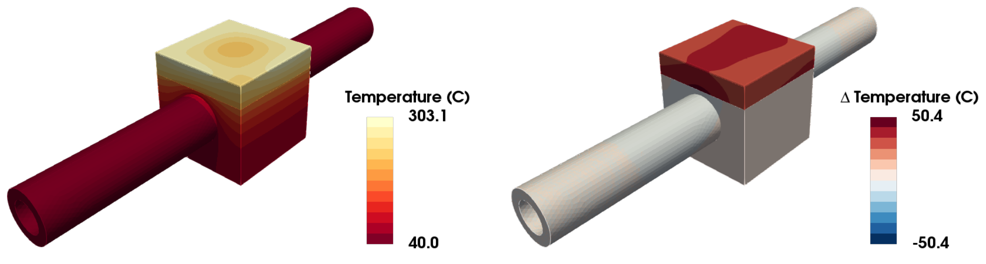
Using experimental data recorded by HIVE, the TCCC value can be estimated, much like the experimental parameters were in the previous section using surrogate models. This information can then be used to perform more accurate simulations, enabling a better understanding of the component's performance.

8.3.2 Surrogate Model

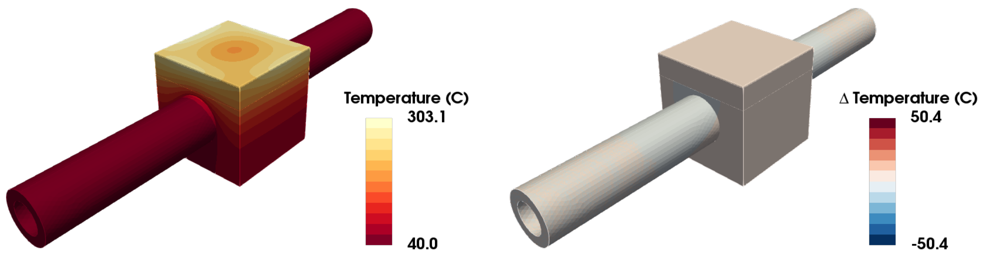
This analysis is carried out on a component with a single bonded surface between the tile and the block, where the bond is no longer treated as a perfect conductor. Externally, the component is identical to that used for the previous analysis (fig. 5.6).

The surrogate model required for this work will have 8 inputs; the 7 experimental parameters used previously and the TCCC for the contacting surface. The parameter range for the first 7 are the same as that used previously (table 7.1), however choosing an appropriate range for the TCCC can be a challenge as its value can range from 0 to infinity. Again, consider the 1D example given by eq. 8.5. If the range of TCCC was chosen as [1,10], then deviation in the interface temperature from a perfect heat exchange as a percentage would vary from 6.06% to 33.33%, which is a substantial change. On the other hand, if the range was chosen as [100,1000] its impact would be between 0.07% and 0.66%, making it very difficult to distinguish from a perfect heat exchange.

It is desirable to identify a range whereby the upper bound behaves similarly to the perfect heat exchange, and the lower bound provides a noticeable change to the component's performance. Following a sensitivity analysis, the range of values for this component has been identified as [2000,20000], with the temperature



(a) Temperature profile, $h_c = 2000$ (b) Difference from PHE, $h_c = 2000$

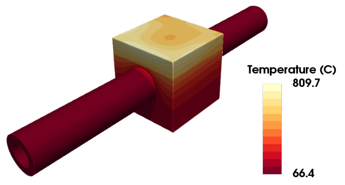


(c) Temperature profile, $h_c = 20000$ (d) Difference from PHE, $h_c = 20000$

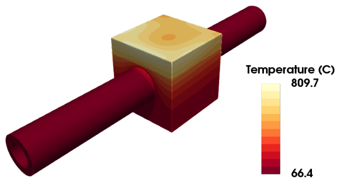
Figure 8.15: The effect of TCC for the upper and lower bound of the parameter space.

profiles and difference from the perfect heat exchange for this upper and lower bound shown in fig. 8.15. For the ‘worst’ case scenario, there is a 50 °C increase in the peak temperature compared with the perfect heat exchange case, which equates to around a 15% increase. For the ‘best’ case scenario, the change is only 6 °C, making its behaviour is nearly identical to the perfect heat exchange case. As a result, it is believed that the likely effect of the bonded surface will be within these two ranges.

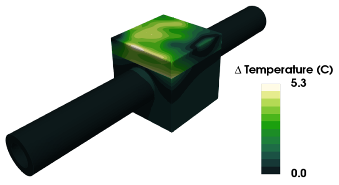
To train the surrogate model 500 observations are used, with 200 reserved for testing the model. Table 8.13 shows the nRMSE for the models generated using the three different covariance kernel functions, which highlights that the RBF model is the best performing. A comparison of a simulation and the TCCC surrogate model using the RBF kernel is given in fig. 8.16.



(a) FE model



(b) Surrogate model



(c) Absolute difference

Figure 8.16: Comparison of TCC temperature surrogate model with simulation.

	Test nRMSE	Train nRMSE
RBF	2.66E-03	1.03E-03
Matérn$_{5/2}$	2.75E-03	1.11E-04
Matérn$_{3/2}$	4.08E-03	7.96E-05

Table 8.13: Accuracy of temperature surrogate models with TCCC (coil A).

8.3.3 Inversely Informed Parameters

Similarly to the work of the previous section, the experimental parameters which result in the required temperatures at the thermocouple locations are calculated using eq. 8.1. While the previous section was interested in the temperature field resulting from the inverse solution, this time it's the accuracy of the inverse solution relating to the TCCC parameter which is of interest.

Unfortunately, the bonded component failed during the experiment due to debonding at the interface, therefore no experimental data was available for this component. As a result, T^{TC}_{exp} in eq. 8.1 will again use data extracted from simulations. Similar to the previous section, this approach has the benefit of knowing the exact TCCC value which was used in the simulation, thus providing the ability to measure the accuracy of that estimated TCCC value.

For simplicity, the 7 thermocouples placed at the centre point of the seven admissible surfaces are used, see table 8.1. A histogram of the error percentages between the inversely predicted TCCC and the value used in the simulation is shown in fig. 8.17 for each of the 200 observations which make up the test dataset.

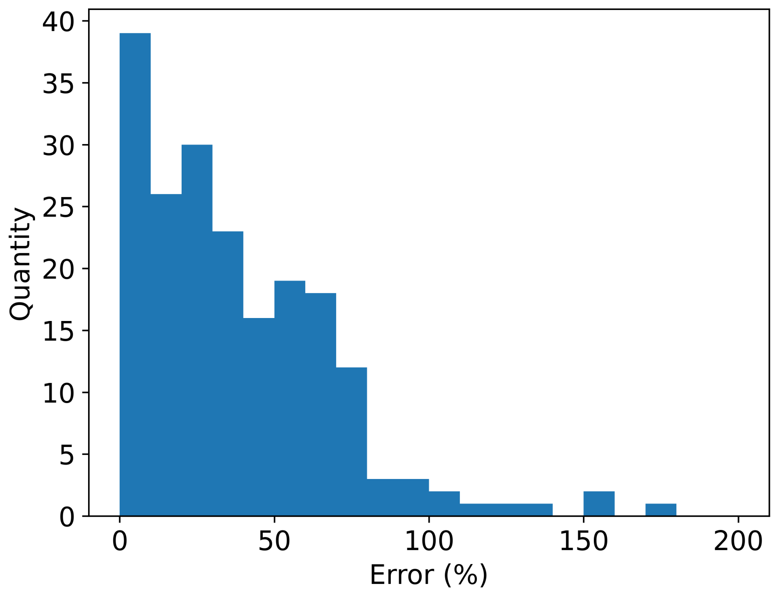
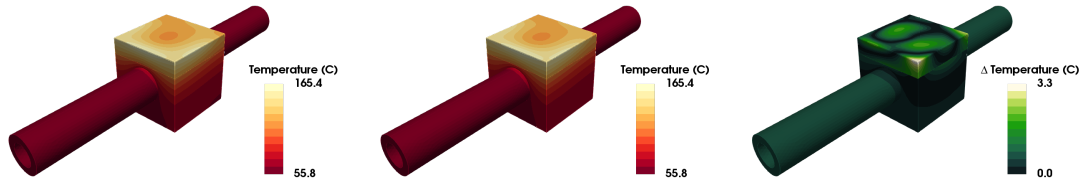


Figure 8.17: Histogram of percentage error between inverse predicted TCCC and value used in the simulation for 200 different cases.

This shows that this method is reasonably accurate in terms of predicting the TCCC, where an absolute percentage error of less than 10% is achieved for 20% of the cases and less than 30% error for 47.5% of the cases. There are, however, a high number of cases where the inverse solution is poor, with 33% of cases resulting in an absolute percentage error of more than 50%. Although not shown in this plot, the largest absolute percentage error over the 200 test cases is over 600%.

The reason for this inaccuracy is due to the non-uniqueness of the inverse solution. Consider the temperature profile shown in fig. 8.18a, which has been generated using the parameters given in the first column of table 8.14. The temperatures at the 7 thermocouple locations (given in table 8.15) have been used in eq. 8.1 to provide the inversely informed parameters, which are given in the second column of 8.14.



(a) FE model (b) Surrogate model (c) Absolute difference

Figure 8.18: Example of non-uniqueness of experimental parameters where seven thermocouples are used.

	Simulation	**Inversely informed**
x_d	4.34E-04	2.56E-03
y_d	-6.65E-03	-6.53E-03
z_d	4.27E-03	3.10E-03
ϕ	3.45E+00	-1.56E-01
$T_{coolant}$	5.58E+01	5.64E+01
$V_{coolant}$	4.41E+00	5.85E+00
I	1.14E+02	1.05E+02
h_c	5.09E+03	1.97E+04

Table 8.14: Comparison between the parameters used as inputs to the simulation and those estimated using temperature at 7 thermocouple locations.

The temperature profile generated using these inversely informed parameters matches well with the temperature field from the simulation, with only a maximum absolute temperature difference of 3 °C see fig. 8.18c. However, the inputs for both are very different, especially for the TCCC which is around 4 times larger than that used in the simulation. This change is counterbalanced by changes in the other parameters, such as ϕ and z_d. This is another example of non-uniqueness of inverse problems.

Following a similar approach to that of section 8.2.4, the parameter space in which admissible solutions is reduced based on prior knowledge on the experimental parameters. Using the same confidence levels as those given in table 8.9 results in a substantial improvement in the prediction of the TCCC, see fig. 8.19. Now, 40% of all cases have an absolute relative error of less than 10%, with only 5% of cases having an error of greater than 50%.

	Temperature (°C)
$T_{exp}^{TC^{(1)}}$	150.91
$T_{exp}^{TC^{(2)}}$	152.68
$T_{exp}^{TC^{(3)}}$	125.56
$T_{exp}^{TC^{(4)}}$	133.16
$T_{exp}^{TC^{(5)}}$	75.67
$T_{exp}^{TC^{(6)}}$	71.85
$T_{exp}^{TC^{(7)}}$	90.61

Table 8.15: Temperature at 7 thermocouple locations (table 8.1) for temperature profile shown in fig. 8.18a.

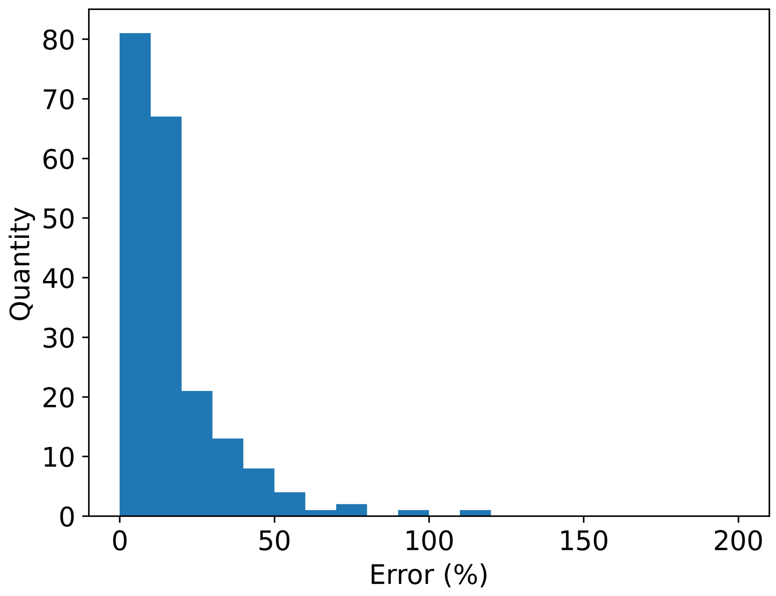


Figure 8.19: Histogram of percentage error between inverse predicted TCCC with confidence and value used in the simulation for 200 different cases.

Improvements could be made in the accuracy of these predictions by increasing the confidence regarding certain measurements. The reduction gained here is based on how HIVE currently operates, however the addition of new tools such as a 3D scanner to accurately measure the positioning of the coil in relation to the component could increase the confidence considerably, thus leading to improvements in the prediction for the TCCC.

8.4 Chapter Summary

This chapter has demonstrated how even extremely sparse experimental data for a highly complex use case can be enriched through the use of ML to provide substantially more insight. This was firstly demonstrated by using the experimental data to infer more useful measures of the component's performance while subjected to fusion like loads. Following this, it was shown that the same data can be used to infer hard to measure, highly variable properties of the component.

The key novelties of this chapter are; firstly, the use of surrogate models to solve a highly complex proxy measurement problem. It was shown that knowledge of the temperature on the component's surface at a handful of pointwise location can be used to infer the Von Mises stress throughout the component, thus providing a much better understanding of the component's strengths and weaknesses. The second novelty is the use of the genetic algorithm to optimise the sensor placement, thus resulting in the need for fewer sensors, which can speed up the time taken to perform tests.

The contents of the *VirtualLab* run files necessary to estimate the temperature and Von Mises stress field from the thermocouple data can be found in listing A.4 in the appendices. This file also contains the necessary steps to perform the thermocouple sensitivity analysis and subsequent placement optimisation.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

This thesis has presented the development and application of FF surrogate models of multi-physics, FE simulations to greatly improve the impact of the HIVE facility on the design of PFCs. As discussed in the introduction, both the goals of smarter testing and enhanced component insight could be achieved by solving inverse problems, which were reviewed in chapter 2, and is a predominant theme in chapters 6 - 8.

The performance of a variety of different supervised learning algorithms and dimensionality reduction techniques have been presented for the construction of 2D and 3D FF surrogate models. GPR coupled with PCA proved to be not only the best performing modelling strategy but also the easiest to implement due to their small number of hyperparameters. In chapter 6 it was shown that FF surrogates actually perform better than the SV surrogate does in accurately predicting the value of the single valued output.

Chapter 7 showed that FF surrogate models for the temperature and stress fields of the component can provide much more insightful DoPEs, enabling a much more varied and targeted suite of experiments to be performed. This smarter testing removes the need for previous experience and intuition of the HIVE facility and avoids performing unnecessary experiments by no longer employing a trial & error approach.

These surrogate models were also used to enrich the sparse experimental data currently generated by HIVE, where in chapter 8 they enabled the full temperature and stress field to be predicted using as little as 3 surface temperature measurements. This provides a much better understanding of the components tested in HIVE, enabling more informed decisions to be made during the next iteration of the design cycle. Given the highly nonlinear nature of the problem, this is a powerful demonstration of the potential benefits for the developed methodology. Along with this, it was also demonstrated that this sparse data can be used to estimate uncertain quantities in the problem, with the TCCC was estimated to within 30% accuracy for 78% of cases when experiment specific knowledge was included.

The *VirtualLab* package developed is not only beneficial to the HIVE operators and its clients, but also has the potential to help a number of people in the computational engineering field. Given a lack of workflow orchestration tools in the field, its fully-autonomous makeup means that it could be an extremely valuable tool. Along with this, its ability to generate vast quantities of synthetic data in a fraction of the time and use it to train surrogate models with little to no knowledge of supervised learning algorithms, is extremely powerful. Moreover, its use of containers and open-source software means that it is extremely portable, enabling anyone to benefit from it.

For the HIVE operators, the automation, and speed which *VirtualLab* performs this analysis means that components can be tested in a high throughput manner, ensuring that HIVE satisfies the demand for the facility. As the *VirtualLab* platform is open-source, this setup is easily shared with clients, enabling them to perform their own explorative work at no additional cost. This provides clients a better understanding of HIVE, enabling them to make more informed requests during an experimental campaign and gain more insight from the gathered experimental data.

Leading on from this, there is huge potential to extend this work to larger scale facilities such as CHIMERA. This facility adds complexity to HIVE through the inclusion of magnetic fields to mechanically load components, along with the use of non-conventional fluid coolants. Similarly to HIVE, the presence of numerous physical loads means that it becomes inherently more difficult to extract the desired quantity and quality of data from an experiment. This framework can similarly be used to ensure that the most valuable data from the experiment is collected. Moreover, as aspects are still yet to be decided for CHIMERA, this framework could be used to identify optimal configurations or placement of sensors a priori.

This framework is not limited to fusion experimental facilities only, but in fact any kind of experimental facility. The construction of DoPEs is commonplace for any experimental facility, while in most cases there will be certain properties which are immeasurable using the available sensors. The generic nature of coupling simulation data with ML means that much of the work carried out is easily transferable, with only domain specific knowledge required to improve the accuracy of the results.

An overview of the workflow for an experimental facility using the framework developed during this PhD is shown in fig 9.1. The different stages of the process have been referenced to sections in this thesis.

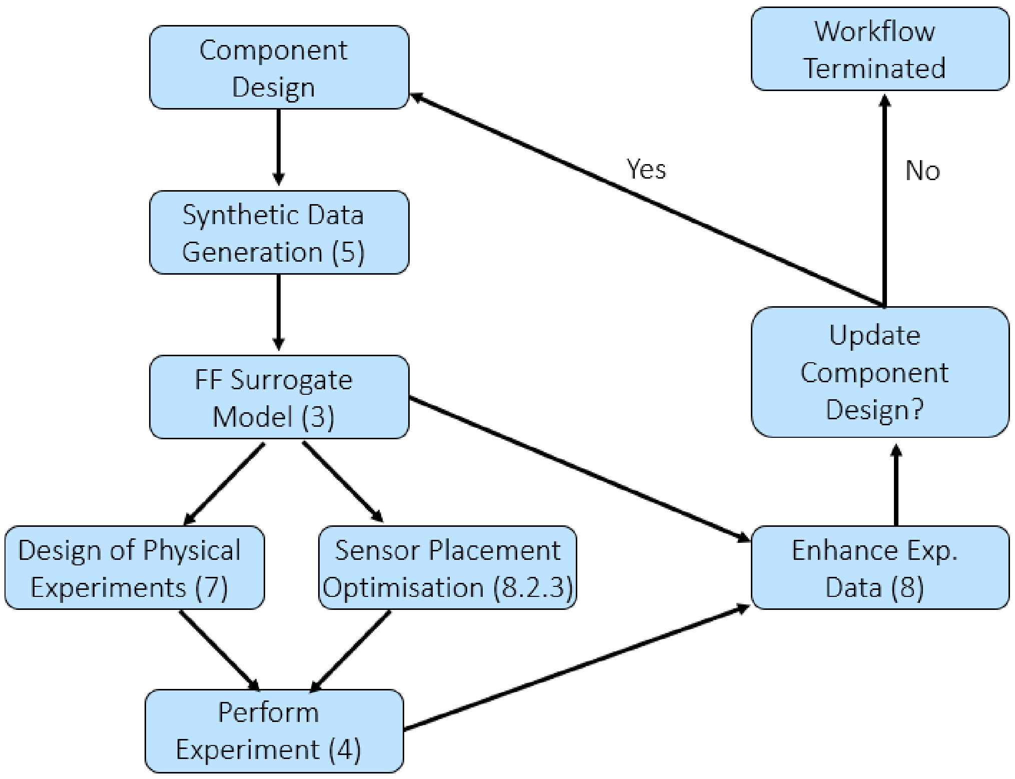


Figure 9.1: Workflow for an experimental facility using the framework developed during this project.

9.2 Developed Packages and Impact

This section provides an overview of the packages developed during this thesis and their contribution to other projects.

9.2.1 MultiSLSQP

The extension of SciPy's SLSQP algorithm to a multi-start method has enabled the inverse modelling analysis used in this work to be performed in a fraction of the time. Harnessing the use of tensor multiplication reduces the computational burden of performing tasks individually. This is especially useful in scenarios where these tasks are computationally intensive, such as matrix inversion, where *MultiSLSQP* resulted in a time reduction of 96% for a reasonably simplistic problem.

9.2.2 VirtualLab

The *VirtualLab* package has been central to the work carried out in this thesis [43]. It is an extremely powerful and adaptable package which enables automation of even the most complex workflows. Its parameterised nature means that setting vast numbers of simulations in advance is easy, while its near perfect scaling allows this data to be collected in a fraction of the time. Its in-built ML method enables the generated synthetic data to be used to train surrogate models with little to no prior knowledge of the algorithms.

The use of containerisation makes its installation simple, regardless of the operating system or architecture. This has facilitated development of workflows on local PCs with 'production runs' to generate vast volumes of data moved to supercomputers, with little more effort than syncing a few python scripts. Placing external software packages into containers not only makes it simple to upgrade to newer versions or add new ones, it also and avoids any potential conflicts that may arise between them.

The structure of *VirtualLab* makes contributing new experiments, methods, modules, and analysis extremely easy. This has been highlighted by the ease in which *VirtualLab* is used in a number of other research projects at Swansea University and other research facilities, an overview of which are given below.

Tensile Test

Tensile testing is a standard engineering and materials science experiment to investigate the mechanical properties of a material [218]. A tensile specimen with a standardized geometry is subjected to uniaxial loading along its length until failure. As shown in fig. 9.2, both ends of a specimen are clamped with the tension created as the two ends are pulled apart. This loading can be applied in two ways; through a constant force rate, where the strain in the component is measured, or a constant strain rate, where the force needed to achieve a measured strain is recorded.

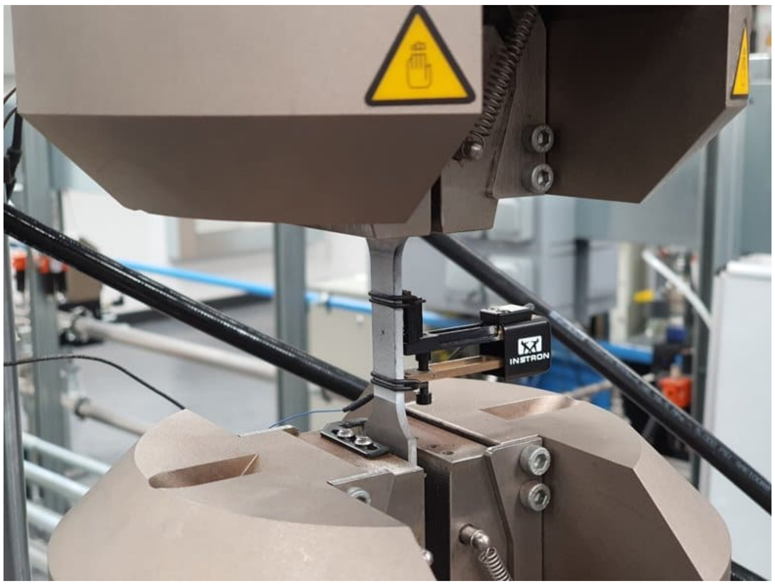


Figure 9.2: Rectangular 'dog-bone' specimen used during tensile test

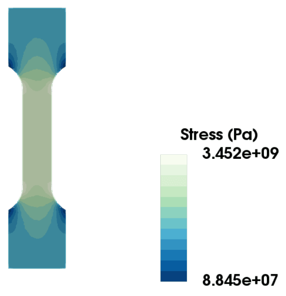
Through this test, the ultimate tensile strength is measured, along with the reduction in cross-sectional area and maximum elongation. From these measurements key material properties can be deduced, such as its elastic modulus, which dictates a material resistance to elastic deformation, its yield strength, which determines when a material deforms plastically, and its tensile strength, which is the load under which the material will break. It also provides information relating to the strain-hardening characteristics of a material.

Generally, two different styles of tensile specimens are used; one with a round cross-sectional area and another with a rectangular cross-sectional area. Fig. 9.2 shows the latter of these, which is commonly referred to as a 'dog bone' on account of its wider end and narrower midsection.

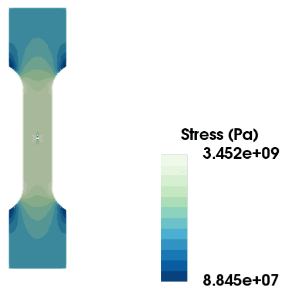
The research undertaken at Swansea University is looking at the effect that microscopic defects, such as porosities or inclusions, created during the manufacturing of the component has on the mechanical material properties. Prior to testing, the dog bone specimens undergo computerised tomography (CT) scans to detect these microscopic defects. From this data, a high fidelity mesh is generated which accounts for these defects, which can be used in computational models. This can be thought of as modelling the ‘as manufactured’ component instead of the ‘as designed’ component (CAD), which results in more accurate modelling of its behaviour [219].

The impact of these unknown, microscopic defects are compared with larger, known defects in the component. These known defects come in the form of elliptic holes included in the component design. Fig. 9.3 shows the stress field of a purely elastic simulation for 4 CAD-based components subjected to uniaxial force of 100 kN.

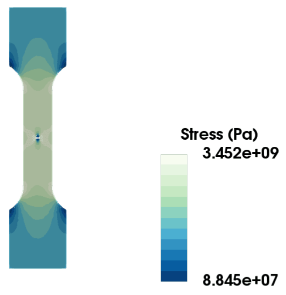
The 4 specimens are; one with no defect (fig. 9.3a), one with a circular hole (fig. 9.3b), one with an elliptic hole (fig. 9.3c) and finally one with a circular hole off-centre (fig. 9.3d). Unsurprisingly, it’s the specimen with an elliptic hole which is subjected to the most severe stresses.



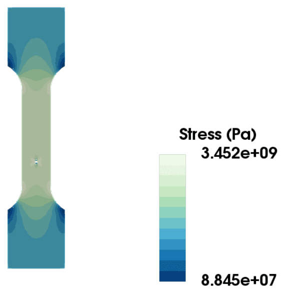
(a) Specimen 0



(b) Specimen 1



(c) Specimen 2



(d) Specimen 3

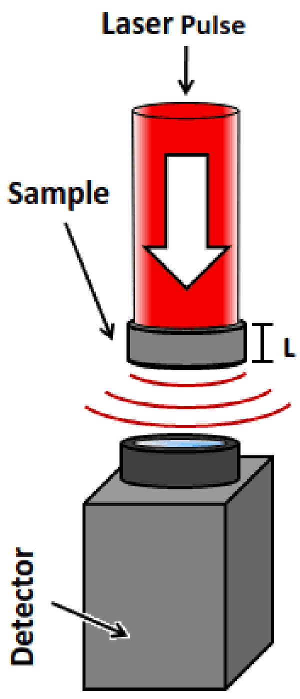
Figure 9.3: Stress field for purely elastic simulation

Laser Flash Analysis

Laser flash analysis is an experiment used to measurement the thermal properties of a material. In this experiment, a short laser pulse heats the 'front' surface of a disc-shaped component, with the resulting temperature rise on the opposing 'bottom' face measured as a function of time using a detector. An illustration of this is shown in fig. 9.4a alongside an image of the machine which conducts the experiment 9.4b.

The data recorded by the detector is used to measure the thermal diffusivity of a material. The thermal diffusivity, α, is calculated using the empirical correlation given by eq. 9.1 where L is the thickness of the sample (highlighted in fig. 9.4a) and $t_{0.5}$ is the half rise time, which is half of the time needed to reach the peak temperature on the bottom surface.

$$\alpha = \frac{0.1388L^2}{t_{0.5}} \tag{9.1}$$



(a) Experiment illustration

(b) LFA machine

Figure 9.4: LFA experiment

A material's thermal diffusivity measures the transfer rate of heat between two points. The faster the temperature rise reaches the bottom face, the higher the thermal diffusivity of the material. From the thermal diffusivity the thermal conductivity, k, of a material can be deduced using eq. 9.2, where C_p and ρ are the specific heat and density of the material, respectively.

$$k = \alpha C_p \rho \tag{9.2}$$

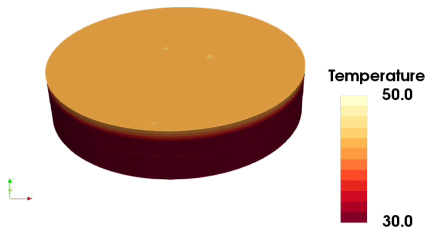
The work carried out at Swansea University using the LFA experiment is similar to that of the tensile test, but instead investigates the effect of microscopic defects on the material's thermal properties. Again *VirtualLab* has been used to construct both CAD and image based simulations.

As the component is briefly subjected to a laser pulse, the LFA experiment is a transient problem and must be modelled as such. The time discretisation for the simulation is defined in the parameters file, shown in listing 9.1, where the initial temperature, InitTemp, time step sizes, dt, and the parameter which decides whether the time discretisation is implicit or explicit Theta are shown. The makeup of dt specifies that two different sizes of time steps will be used; the first 50 will calculate the solution every 0.00002 seconds, once they are completed the next 100 will calculate the solution every 0.0005 seconds. The other attributes of Sim shown in this listing are used to define material properties and BCs of the problem.

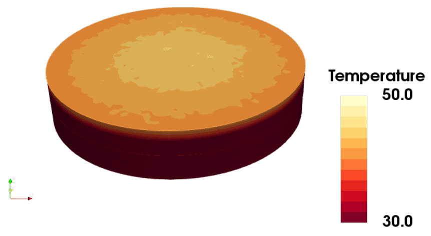
```
Sim = Namespace()
Sim.AsterFile = 'Disc_Lin'
Sim.Mesh = 'Mesh1'

Sim.Energy = 5.32468714
Sim.LaserT= 'Trim'
Sim.LaserS = 'Gauss'

Sim.Materials = 'Copper'

Sim.HTC = 0
Sim.ExtTemp = 20

Sim.InitTemp = 20
Sim.dt = [(0.00002,50,1), (0.0005,100,2)]
Sim.Theta = 0.5
```

Listing 9.1: Content of Parameter master file for LFA experiment

Ideally the profile of the last pulse would be uniform, however in practice its magnitude is greater at the centre. As a result, it is modelled as a 2D Gaussian profile over the top surface. Fig. 9.5 shows the temperature field calculated by the CAD-based simulation after 0.001 s, with a uniform laser pulse used for the simulation shown in fig. 9.5a and a Gaussian profile used for fig. 9.5b



(a) Uniform laser pulse



(b) Gaussian laser pulse

Figure 9.5: Temperature profile of CAD based LFA simulation after 0.001s

Irradiation Damage

As highlighted in section 4.2.1, during a fusion reaction, a neutron is released to the plasma, which interacts with the PFCs on the inner wall of the fusion device. Over time, exposure to neutrons degrades a material, worsening its material properties in a process known as irradiation damage. This is likely to have a large effect on the lifespan of many components in a fusion device.

Modelling the effects of irradiation damage is extremely complex since neutrons interact with precipitates and dislocation loops in the material makeup, see fig. 9.6. *VirtualLab* is used as part of a EUROfusion fellowship to predict the long term material properties of component's subjected to neutron damage. This requires using a number of external modules, including OpenMC, MoDElib and Paramak alongside *code_aster*. The workflow for modelling the effects of neutron damage is shown in fig. 9.7, highlighting that *VirtualLab* is capable of dealing with highly complex and specialised cases.

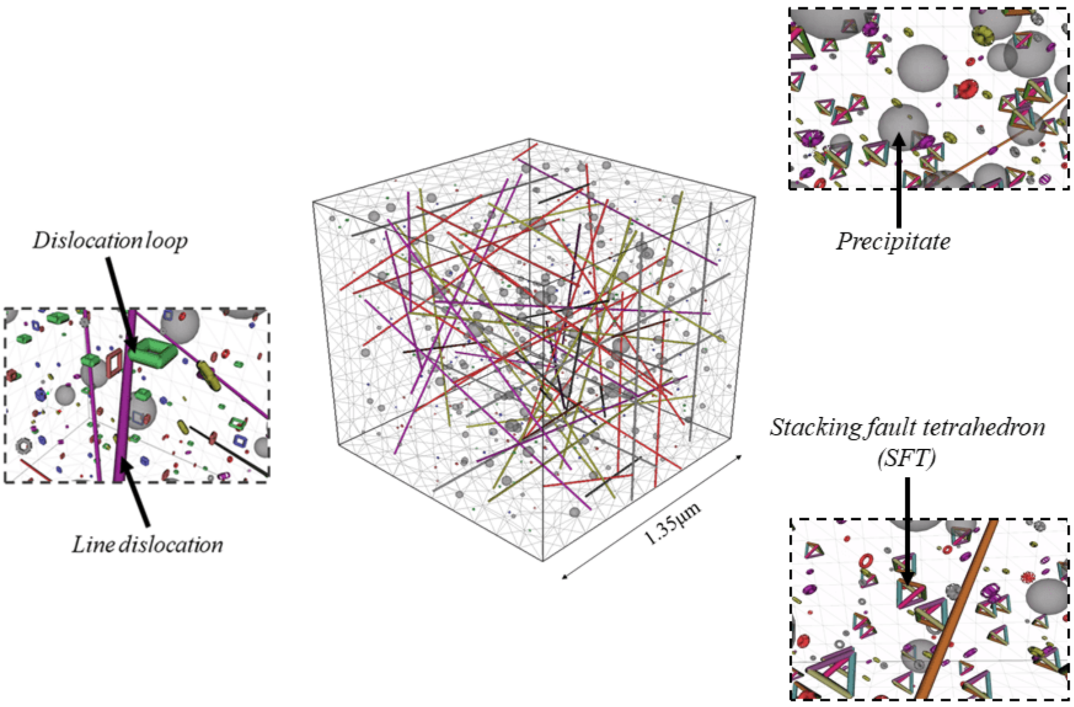


Figure 9.6: Microscopic features which neutrons interact with.

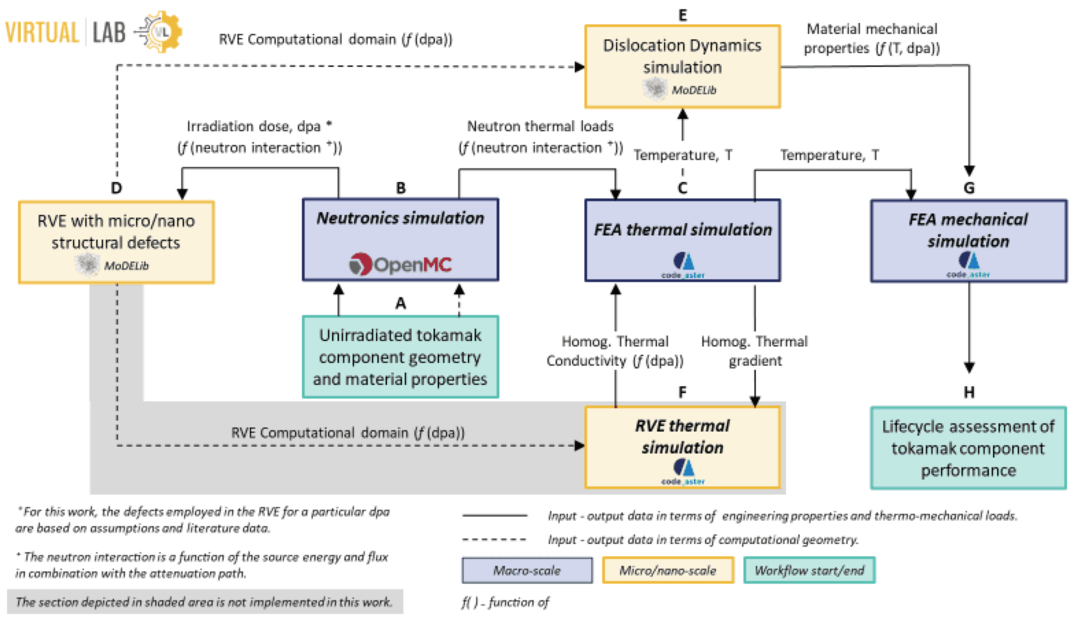

VIRTUAL LAB
RVE Computational domain (*f* (dpa))
E
Dislocation Dynamics simulation
MoDELib
Material mechanical properties (*f* (T, dpa))
Irradiation dose, dpa *
(*f* (neutron interaction *))
Neutron thermal loads
(*f* (neutron interaction *))
Temperature, T
Temperature, T
D
B
C
G
RVE with micro/nano structural defects
MoDELib
Neutronics simulation
OpenMC
FEA thermal simulation
code_aster
FEA mechanical simulation
code_aster
A
Unirradiated tokamak component geometry and material properties
Homog. Thermal Conductivity (*f* (dpa))
Homog. Thermal gradient
F
H
RVE Computational domain (*f* (dpa))
RVE thermal simulation
code_aster
Lifecycle assessment of tokamak component performance
* For this work, the defects employed in the RVE for a particular dpa are based on assumptions and literature data.
* The neutron interaction is a function of the source energy and flux in combination with the attenuation path.
The section depicted in shaded area is not implemented in this work.
Input - output data in terms of engineering properties and thermo-mechanical loads.
Input - output data in terms of computational geometry.
Macro-scale
Micro/nano-scale
Workflow start/end
f( ) - function of

Figure 9.7: Workflow used for irradiation damage modelling

9.3 Future Work

9.3.1 Experimental Data

The work presented in chapter 8 required experimentally collected thermocouple data, which was unfortunately unavailable. While the simulated experimental data used provided certain benefits in terms of the ability to quantify its accuracy, this would have been a more powerful demonstration using actual experimental data. Once suitable experimental data is available, this method could be applied to it.

9.3.2 Design Optimisation

The framework shown in fig. 9.1 shows the potential of optimising the design of the component in an automated way. The information gathered from the surrogate model, such as maximum stresses in the component during various loading cases, can be fed back to the parameterised design iteratively until an optimally designed component is produced.

9.3.3 Time-series Forecasting

This work has focused on the steady-state behaviour of components due to a focus on their 'in-service' condition. However, additional knowledge can be gleaned from the transient data recorded prior to achieving steady state, including developing capabilities for improved life-cycle assessment of real time feedback control.

Extension of this work to the transient regime would likely require generating surrogate models with an alternative ML algorithm which is able to cope with large quantities of sequential data. RNNs shown huge promise in this regard, having already demonstrated strong results for a variety of different engineering problems [65]–[68].

9.3.4 Proxy Measurements

As discussed in chapter 1, the use of proxy measurements are common in a wide variety of experiments, where the property of interest is difficult to measure. The ability shown in this work in identifying properties of interest using extremely sparse data for a highly complex, multi-physics problem paves the way for a large amount of further research.

This framework could be used for alternative component geometries to those currently specified for standardised testing of coupons, such as in a tensile test and LFA. Doing so would provide a baseline for the framework against a standardised facility. This, however, would be invaluable for investigate advanced manufacturing methods such as additive manufacturing or composite materials, where the inhomogeneous meso-structure is an integral part of the behaviour. Furthermore, it enables estimating extremely localised and internal behaviours in component scale testing, where otherwise only broad observations may be possible due to experimental limitations.

9.3.5 VirtualLab

The *VirtualLab* platform is open source and freely available for others to use and, ideally, contribute to. There is currently a growing body of work from alternative research projects which will need to be integrated for others to benefit from. The flexibility which *VirtualLab* offers coupled with its ease of installation and contribution means that there is huge potential for this platform to be used for almost any engineering workflow. This has already been highlighted by the diverse projects and workflows which *VirtualLab* is already involved with. Extension of this package for use with efficiently parallelised simulation codes, such as *code_aster* with MPI, ParaFEM or MOOSE, would enable its use in projects where the focus is on problems with large numbers of DoFs.

References

[1] H. Ritchie, M. Roser, and P. Rosado, "Energy," *Our World in Data*, Oct. 27, 2022. [Online]. Available: `https://ourworldindata.org/energy-production-consumption` (visited on 01/03/2023).

[2] R. Lindsey. "Climate Change: Atmospheric Carbon Dioxide — NOAA Climate.gov." (Jun. 23, 2022), [Online]. Available: `http://www.climate.gov/news-features/understanding-climate` (visited on 01/03/2023).

[3] X. Zheng, D. Streimikiene, T. Balezentis, A. Mardani, F. Cavallaro, and H. Liao, "A review of greenhouse gas emission profiles, dynamics, and climate change mitigation efforts across the key climate change players," *Journal of Cleaner Production*, vol. 234, pp. 1113–1133, Oct. 10, 2019, ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2019.06.140`.

[4] O. Wallach. "Race to Net Zero: Carbon Neutral Goals by Country," Visual Capitalist. (Jun. 8, 2021), [Online]. Available: `https://www.visualcapitalist.com/sp/race-to-net-zero-carbon-neutral-goals-by-country/` (visited on 01/03/2023).

[5] "Global Emissions," Center for Climate and Energy Solutions. (), [Online]. Available: `https://www.c2es.org/content/international-emissions/` (visited on 03/23/2023).

[6] M. Lewis. "A Chinese company is building a colossal 16 MW offshore wind turbine [Update]," Electrek. (Feb. 22, 2022), [Online]. Available: `https://electrek.co/2022/02/22/a-chinese-company-is-building-a-colossal-16-mw-offshore-wind-turbine/` (visited on 01/03/2023).

[7] "Nuclear Fission and Fusion - Difference and Comparison." (), [Online]. Available: `https://www.diffen.com/difference/Nuclear_Fission_vs_Nuclear_Fusion` (visited on 10/28/2022).

[8] “JET,” EUROfusion. (), [Online]. Available: https://euro-fusion.org/devices/jet/ (visited on 03/25/2023).

[9] I. T. Chapman and A. W. Morris, “UKAEA capabilities to address the challenges on the path to delivering fusion power,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 377, no. 2141, p. 20 170 436, Mar. 25, 2019. DOI: 10.1098/rsta.2017.0436.

[10] D. Hancock and D. Homfray, *Testing Advanced Divertor Concepts for Fusion Power PlantsUsing a Small High Heat Flux Facil- ity,* Jan. 1, 2020. [Online]. Available: https://scientific-publications.ukaea.uk/wp-content/uploads/Preprints/UKAEA-CCFE-PR1833.pdf.

[11] M. Tanaka and G. Dulikravich, *Inverse Problems in Engineering Mechanics.* Elsevier Science, 1998, ISBN: 978-0-08-053516-6.

[12] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd edition. Upper Saddle River, N.J: Pearson, Jul. 27, 2004, 408 pp., ISBN: 978-0-201-54361-2.

[13] S. Jacquemoud, C. Bacour, H. Poilvé, and J. .-. Frangi, “Comparison of Four Radiative Transfer Models to Simulate Plant Canopies Reflectance: Direct and Inverse Mode,” *Remote Sensing of Environment*, vol. 74, no. 3, pp. 471–481, Dec. 1, 2000, ISSN: 0034-4257. DOI: 10.1016/S0034-4257(00)00139-5.

[14] F. G. Boudinot and J. Wilson, “Does a proxy measure up? A framework to assess and convey proxy reliability,” *Climate of the Past*, vol. 16, no. 5, pp. 1807–1820, Sep. 28, 2020, ISSN: 1814-9324. DOI: 10.5194/cp-16-1807-2020. [Online]. Available: https://cp.copernicus.org/articles/16/1807/2020/.

[15] M. R. Montgomery, M. Gragnolati, K. A. Burke, and E. Paredes, “Measuring Living Standards with Proxy Variables,” *Demography*, vol. 37, no. 2, pp. 155–174, 2000, ISSN: 0070-3370. DOI: 10.2307/2648118.

[16] C. Turlure, J. Choutt, H. Van Dyck, M. Baguette, and N. Schtickzelle, “Functional habitat area as a reliable proxy for population size: Case study using two butterfly species of conservation concern,” *Journal of Insect Conservation*, vol. 14, no. 4, pp. 379–388, Aug. 1, 2010, ISSN: 1572-9753. DOI: 10.1007/s10841-010-9269-3.

[17] J. De Valck and J. Rolfe, “Reviewing the use of proxies to value coastal and marine biodiversity protection: The Great Barrier Reef in Australia,” *Marine Policy*, vol. 136, p. 104 890, Feb. 1, 2022, ISSN: 0308-597X. DOI: 10.1016/j.marpol.2021.104890.

[18] M. C. Chen, J. R. Anderson, and M. H. Sohn, “What can a mouse cursor tell us more? correlation of eye/mouse movements on web browsing,” in *CHI ’01 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA ’01, New York, NY, USA: Association for Computing Machinery, Mar. 31, 2001, pp. 281–282, ISBN: 978-1-58113-340-0. DOI: 10.1145/634067.634234.

[19] W. J. Parker, R. J. Jenkins, C. P. Butler, and G. L. Abbott, “Flash Method of Determining Thermal Diffusivity, Heat Capacity, and Thermal Conductivity,” *Journal of Applied Physics*, vol. 32, no. 9, pp. 1679–1684, Sep. 1961, ISSN: 0021-8979. DOI: 10.1063/1.1728417.

[20] R. D. Cowan, “Pulse Method of Measuring Thermal Diffusivity at High Temperatures,” *Journal of Applied Physics*, vol. 34, no. 4, pp. 926–927, Apr. 1963, ISSN: 0021-8979. DOI: 10.1063/1.1729564.

[21] N. Saba, M. Jawaid, and M. T. H. Sultan, “1 - An overview of mechanical and physical testing of composite materials,” in *Mechanical and Physical Testing of Biocomposites, Fibre-Reinforced Composites and Hybrid Composites*, ser. Woodhead Publishing Series in Composites Science and Engineering, M. Jawaid, M. Thariq, and N. Saba, Eds., Woodhead Publishing, Jan. 1, 2019, pp. 1–12, ISBN: 978-0-08-102292-4. DOI: 10.1016/B978-0-08-102292-4.00001-1.

[22] O. M. Alifanov, *Inverse Heat Transfer Problems.* Springer Science & Business Media, Dec. 6, 2012, 360 pp., ISBN: 978-3-642-76436-3. Google Books: olzmCAAAQBAJ.

[23] H. R. Tamaddon-Jahromi, N. K. Chakshu, I. Sazonov, L. M. Evans, H. Thomas, and P. Nithiarasu, “Data-driven inverse modelling through neural network (deep learning) and computational heat transfer,” *Computer Methods in Applied Mechanics and Engineering*, vol. 369, p. 113 217, Sep. 1, 2020, ISSN: 0045-7825. DOI: 10.1016/j.cma.2020.113217.

[24] H. Vardhan, P. Volgyesi, and J. Sztipanovits, *Fusion of ML with Numerical Simulation for Optimized Propeller Design*. Feb. 28, 2023. DOI: 10.48550/arXiv.2302.14740.

[25] J. S. Toquica, P. S. Oliveira, W. S. R. Souza, J. M. S. T. Motta, and D. L. Borges, "An analytical and a Deep Learning model for solving the inverse kinematic problem of an industrial parallel robot," *Computers & Industrial Engineering*, vol. 151, p. 106 682, Jan. 1, 2021, ISSN: 0360-8352. DOI: 10.1016/j.cie.2020.106682.

[26] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006, ISBN: 0-387-31073-8.

[27] T. Hastie and R. Tibshirani, *The Elements of Statistical Learning*. New York: Springer Series in Statistics, 2009.

[28] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1, 1943, ISSN: 1522-9602. DOI: 10.1007/BF02478259.

[29] D. E. Smith, *A Source Book in Mathematics*, 1st ed. New York [etc.]: McGraw-Hill Book Company, inc., 1929, xvii, 701.

[30] V. Thomée, "From finite differences to finite elements: A short history of numerical analysis of partial differential equations," *Journal of Computational and Applied Mathematics*, Numerical Analysis 2000. Vol. VII: Partial Differential Equations, vol. 128, no. 1, pp. 1–54, Mar. 1, 2001, ISSN: 0377-0427. DOI: 10.1016/S0377-0427(00)00507-0.

[31] M. F. Kasim, D. Watson-Parris, L. Deaconu, *et al.*, "Building high accuracy emulators for scientific simulations with deep neural architecture search," *Machine Learning: Science and Technology*, vol. 3, no. 1, p. 015 013, Dec. 2021, ISSN: 2632-2153. DOI: 10.1088/2632-2153/ac3ffa.

[32] S. Bagheri, U. Reinicke, D. Anders, and W. Konen, "Surrogate-assisted optimization for augmentation of finite element techniques," *Journal of Computational Science*, vol. 54, p. 101 427, Sep. 1, 2021, ISSN: 1877-7503. DOI: 10.1016/j.jocs.2021.101427.

[33] O. Sallam, *Background and Literature Review of Inverse Problem History, Inverse Modeling Methods and Geophysics for Groundwater Studies.* Dec. 15, 2019. DOI: 10.13140/RG.2.2.13201.79201.

[34] M. M. Billah, A. I. Khan, J. Liu, and P. Dutta, "Physics-informed deep neural network for inverse heat transfer problems in materials," *Materials Today Communications*, vol. 35, p. 106 336, Jun. 1, 2023, ISSN: 2352-4928. DOI: 10.1016/j.mtcomm.2023.106336.

[35] Z. He, F. Ni, W. Wang, and J. Zhang, "A physics-informed deep learning method for solving direct and inverse heat conduction problems of materials," *Materials Today Communications*, vol. 28, p. 102 719, Sep. 1, 2021, ISSN: 2352-4928. DOI: 10.1016/j.mtcomm.2021.102719.

[36] R. Jin, X. Du, and W. Chen, "The use of metamodeling techniques for optimization under uncertainty," *Structural and Multidisciplinary Optimization*, vol. 25, no. 2, pp. 99–116, Jul. 1, 2003, ISSN: 1615-1488. DOI: 10.1007/s00158-002-0277-0.

[37] C. Thelin, J. Salmon, S. Gorrell, *et al.*, "Using surrogate models to predict nodal results for fatigue risk analysis," *International Journal of Fatigue*, vol. 146, p. 106 039, May 1, 2021, ISSN: 0142-1123. DOI: 10.1016/j.ijfatigue.2020.106039.

[38] A. Singh, M. Wolf, G. Jacobs, and F. König, "Machine learning based surrogate modelling for the prediction of maximum contact temperature in EHL line contacts," *Tribology International*, vol. 179, p. 108 166, Jan. 1, 2023, ISSN: 0301-679X. DOI: 10.1016/j.triboint.2022.108166.

[39] C. Schemmann, "Optimization of the Operation Characteristic of a Highly Stressed Centrifugal Compressor Impeller Using Automated Optimization and Metamodeling Methods," Jun. 29, 2017. DOI: 10.1115/GT2017-63262.

[40] S. Bunnell, C. Thelin, S. Gorrell, J. Salmon, C. Ruoti, and A. Hepworth, "Rapid Visualization of Compressor Blade Finite Element Models Using Surrogate Modeling," Jun. 11, 2018, V07AT30A011. DOI: 10.1115/GT2018-77188.

[41] R. Heap, A. Hepworth, and C. Jensen, "Real-Time Visualization of Finite Element Models Using Surrogate Modeling Methods," *Journal of Computing and Information Science in Engineering*, vol. 15, p. 011 007, Mar. 1, 2015. DOI: 10.1115/1.4029217.

[42] M. R. Kianifar and F. Campean, “Performance evaluation of metamodelling methods for engineering problems: Towards a practitioner guide,” *Structural and Multidisciplinary Optimization*, vol. 61, no. 1, pp. 159–186, Jan. 1, 2020, ISSN: 1615-1488. DOI: `10.1007/s00158-019-02352-1`.

[43] R. Lewis, L. M. Evans, and B. Thorpe, *VirtualLab*, version v0.1, 2023. DOI: `10.5281/zenodo.10137606`.

[44] R. Lewis, Ll. M. Evans, and B. Thorpe, *VirtualLab documentation*, 2023. [Online]. Available: `https://virtuallab.readthedocs.io/en/latest/index.html`.

[45] R. Lewis, *Simulation data for HIVE experiment generated using VirtualLab*, 2023. DOI: `10.5281/zenodo.8300663`.

[46] R. Lewis, “MultiSLSQP: An efficient multi-start SLSQP algorithm,” 2023. DOI: `10.5281/zenodo.10058006`.

[47] J. Hadamard, “Sur les problèmes aux dérivées partielles et leur signification physique,” *Princeton University Bulletin*, vol. 13, pp. 49–52, 1902.

[48] J. Virieux, R. Brossier, L. Métivier, S. Operto, and A. Ribodetti, “Direct and indirect inversions,” *Journal of Seismology*, vol. 20, no. 4, pp. 1107–1121, Oct. 1, 2016, ISSN: 1573-157X. DOI: `10.1007/s10950-016-9587-3`.

[49] G. H. Golub, P. C. Hansen, and D. P. O’Leary, “Tikhonov Regularization and Total Least Squares,” *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 1, pp. 185–194, Jan. 1999, ISSN: 0895-4798. DOI: `10.1137/S0895479897326432`.

[50] Y. Jaluria, “Solution of Inverse Problems in Thermal Systems,” *Journal of Thermal Science and Engineering Applications*, vol. 12, no. 1, Sep. 19, 2019, ISSN: 1948-5085. DOI: `10.1115/1.4042353`. [Online]. Available: `https://doi.org/10.1115/1.4042353`.

[51] C. Wanigasekara, E. Oromiehie, A. Swain, B. G. Prusty, and S. K. Nguang, “Machine learning-based inverse predictive model for AFP based thermoplastic composites,” *Journal of Industrial Information Integration*, vol. 22, p. 100 197, Jun. 1, 2021, ISSN: 2452-414X. DOI: `10.1016/j.jii.2020.100197`.

[52] J. Wilson, “Collaborative Intelligence: Humans and AI Are Joining Forces,” *Harvard Business Review*, Jul. 1, 2018, ISSN: 0017-8012.

[53] A. Dabrowski and L. Dabrowski, "Inverse heat transfer problem solution of sounding rocket using moving window optimization," *PLoS ONE*, vol. 14, no. 6, e0218600, Jun. 24, 2019, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0218600. pmid: 31233533.

[54] P. Zhu, Y. Zhang, and G.-L. Chen, "Metamodel-based lightweight design of an automotive front-body structure using robust optimization," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 223, no. 9, pp. 1133–1147, Sep. 1, 2009, ISSN: 0954-4070. DOI: 10.1243/09544070JAUTO1045.

[55] V. T. Dang, C. Labergere, and P. Lafon, "POD surrogate models using adaptive sampling space parameters for springback optimization in sheet metal forming," *Procedia Engineering*, International Conference on the Technology of Plasticity, ICTP 2017, 17-22 September 2017, Cambridge, United Kingdom, vol. 207, pp. 1588–1593, Jan. 1, 2017, ISSN: 1877-7058. DOI: 10.1016/j.proeng.2017.10.1053.

[56] K. Khaledi, S. Miro, M. König, and T. Schanz, "Robust and reliable metamodels for mechanized tunnel simulations," *Computers and Geotechnics*, vol. 61, pp. 1–12, Sep. 1, 2014, ISSN: 0266-352X. DOI: 10.1016/j.compgeo.2014.04.005.

[57] J. Havinga, P. K. Mandal, and T. van den Boogaard, "Exploiting data in smart factories: Real-time state estimation and model improvement in metal forming mass production," *International Journal of Material Forming*, vol. 13, no. 5, pp. 663–673, Sep. 1, 2020, ISSN: 1960-6214. DOI: 10.1007/s12289-019-01495-2.

[58] H. Kato and K.-i. Funazaki, "POD-Driven Adaptive Sampling for Efficient Surrogate Modeling and its Application to Supersonic Turbine Optimization," in *Volume 2B: Turbomachinery*, Düsseldorf, Germany: American Society of Mechanical Engineers, Jun. 16, 2014, V02BT45A023, ISBN: 978-0-7918-4561-5. DOI: 10.1115/GT2014-27229.

[59] R. Dupuis, J.-C. Jouhaud, and P. Sagaut, "Aerodynamic Data Predictions for Transonic Flows via a Machine-Learning-based Surrogate Model," Jan. 8, 2018. DOI: 10.2514/6.2018-1905.

[60] X. Li and N. Polydorides, "Time-efficient surrogate models of thermal modeling in laser powder bed fusion," *Additive Manufacturing*, vol. 59, p. 103 122, Nov. 1, 2022, ISSN: 2214-8604. DOI: 10.1016/j.addma.2022.103122.

[61] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, Feb. 1991, ISSN: 0001-1541, 1547-5905. DOI: 10.1002/aic.690370209.

[62] A. Shokraei Fard, D. C. Reutens, and V. Vegh, "From CNNs to GANs for cross-modality medical image estimation," *Computers in Biology and Medicine*, vol. 146, p. 105 556, Jul. 1, 2022, ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2022.105556.

[63] F. Strozzi and R. Pozzi, "Improving time series features identification by means of Convolutional Neural Networks and Recurrence Plot," *IFAC-PapersOnLine*, 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022, vol. 55, no. 10, pp. 601–606, Jan. 1, 2022, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2022.09.464.

[64] O. J. Bartlett, D. M. Benoit, K. A. Pimbblet, B. Simmons, and L. Hunt, "Noise reduction on single-shot images using an autoencoder," *Monthly Notices of the Royal Astronomical Society*, vol. 521, no. 4, pp. 6318–6329, Mar. 31, 2023, ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stad665. arXiv: 2303.00656 [astro-ph].

[65] Z. C. Lipton, J. Berkowitz, and C. Elkan. "A Critical Review of Recurrent Neural Networks for Sequence Learning." arXiv: 1506.00019 [cs]. (Oct. 17, 2015), preprint.

[66] M. B. Gorji, M. Mozaffar, J. N. Heidenreich, J. Cao, and D. Mohr, "On the potential of recurrent neural networks for modeling path dependent plasticity," *Journal of the Mechanics and Physics of Solids*, vol. 143, p. 103 972, Oct. 1, 2020, ISSN: 0022-5096. DOI: 10.1016/j.jmps.2020.103972.

[67] B. P. van de Weg, L. Greve, M. Andres, T. K. Eller, and B. Rosic, "Neural network-based surrogate model for a bifurcating structural fracture response," *Engineering Fracture Mechanics*, vol. 241, p. 107 424, Jan. 1, 2021, ISSN: 0013-7944. DOI: 10.1016/j.engfracmech.2020.107424.

[68] T. Tancogne-Dejean, M. B. Gorji, J. Zhu, and D. Mohr, "Recurrent neural network modeling of the large deformation of lithium-ion battery cells," *International Journal of Plasticity*, vol. 146, p. 103 072, Nov. 1, 2021, ISSN: 0749-6419. DOI: 10.1016/j.ijplas.2021.103072.

[69] L. Liang, M. Liu, C. Martin, and W. Sun, "A deep learning approach to estimate stress distribution: A fast and accurate surrogate of finite-element analysis," *Journal of The Royal Society Interface*, vol. 15, no. 138, p. 20 170 844, Jan. 24, 2018. DOI: 10.1098/rsif.2017.0844.

[70] P. Sharma, L. Evans, M. Tindall, and P. Nithiarasu, "Stiff-PDEs and Physics-Informed Neural Networks," *Archives of Computational Methods in Engineering*, Feb. 7, 2023, ISSN: 1886-1784. DOI: 10.1007/s11831-023-09890-4.

[71] S. Manavi, T. Becker, and E. Fattahi, "Enhanced surrogate modelling of heat conduction problems using physics-informed neural network framework," *International Communications in Heat and Mass Transfer*, vol. 142, p. 106 662, Mar. 1, 2023, ISSN: 0735-1933. DOI: 10.1016/j.icheatmasstransfer.2023.106662.

[72] L. Van Gelder, P. Das, H. Janssen, and S. Roels, "Comparative study of metamodelling techniques in building energy simulation: Guidelines for practitioners," *Simulation Modelling Practice and Theory*, vol. 49, pp. 245–257, Dec. 1, 2014, ISSN: 1569-190X. DOI: 10.1016/j.simpat.2014.10.004.

[73] B. Bhattacharyya, "Uncertainty quantification of dynamical systems by a POD–Kriging surrogate model," *Journal of Computational Science*, vol. 60, p. 101 602, Apr. 1, 2022, ISSN: 1877-7503. DOI: 10.1016/j.jocs.2022.101602.

[74] T.-H. Yi, H.-N. Li, and M. Gu, "A new method for optimal selection of sensor location on a high-rise building using simplified finite element model," *Structural Engineering and Mechanics*, vol. 37, Mar. 25, 2011. DOI: 10.12989/sem.2011.37.6.671.

[75] F. G. H. Yap and H.-H. Yen, "A survey on sensor coverage and visual data capturing/processing/transmission in wireless visual sensor networks," *Sensors (Basel, Switzerland)*, vol. 14, no. 2, pp. 3506–3527, Feb. 20, 2014, ISSN: 1424-8220. DOI: 10.3390/s140203506. pmid: 24561401.

[76] M. Marks, “A Survey of Multi-Objective Deployment in Wireless Sensor Networks,” *Journal of Telecommunications and Information Technology*, no. 3, pp. 36–41, 3 2010, ISSN: 1899-8852.

[77] V. Akbarzadeh, A. H.-R. Ko, C. Gagné, and M. Parizeau, “Topography-Aware Sensor Deployment Optimization with CMA-ES,” in *Parallel Problem Solving from Nature, PPSN XI*, ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2010, pp. 141–150, ISBN: 978-3-642-15871-1. DOI: `10.1007/978-3-642-15871-1_15`.

[78] D. Jourdan and O. de Weck, “Layout optimization for a wireless sensor network using a Multi-Objective Genetic Algorithm,” presented at the IEEE Vehicular Technology Conference, vol. 5, Feb. 1, 2004, 2466–2470 Vol.5, ISBN: 978-0-7803-8255-8. DOI: `10.1109/VETECS.2004.1391366`.

[79] P. Chiu and F. Lin, “A simulated annealing algorithm to support the sensor placement for target location,” in *Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No.04CH37513)*, vol. 2, May 2004, 867–870 Vol.2. DOI: `10.1109/CCECE.2004.1345252`.

[80] Z. Xu, Y. Guo, and J. Homer Saleh, “Multi-objective optimization for sensor placement: An integrated combinatorial approach with reduced order model and Gaussian process,” *Measurement*, vol. 187, p. 110 370, Jan. 1, 2022, ISSN: 0263-2241. DOI: `10.1016/j.measurement.2021.110370`.

[81] A. Keane and J. Scanlan, “Design search and optimization in aerospace engineering,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1859, pp. 2501–2529, May 22, 2007. DOI: `10.1098/rsta.2007.2019`.

[82] P. Ruane, P. Walsh, and J. Cosgrove, “Using Simulation Optimization to Improve the Performance of an Automated Manufacturing Line,” *Procedia Computer Science*, 4th International Conference on Industry 4.0 and Smart Manufacturing, vol. 217, pp. 630–639, Jan. 1, 2023, ISSN: 1877-0509. DOI: `10.1016/j.procs.2022.12.259`.

[83] J. Jiang, X. Xu, and J. Stringer, “Optimization of process planning for reducing material waste in extrusion based additive manufacturing,” *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 317–325, Oct. 1, 2019, ISSN: 0736-5845. DOI: `10.1016/j.rcim.2019.05.007`.

[84] D. Liu, X. Hu, and Q. Jiang, "Design and optimization of logistics distribution route based on improved ant colony algorithm," *Optik*, vol. 273, p. 170 405, Feb. 1, 2023, ISSN: 0030-4026. DOI: 10.1016/j.ijleo.2022.170405.

[85] "The Traveling Salesman Problem," in *Combinatorial Optimization: Theory and Algorithms*, ser. Algorithms and Combinatorics, B. Korte and J. Vygen, Eds., Berlin, Heidelberg: Springer, 2008, pp. 527–562, ISBN: 978-3-540-71844-4. DOI: 10.1007/978-3-540-71844-4_21.

[86] K. Tadakuma, Y. Tani, and S. Aso, "Effect of Fuselage Cross Section on Aerodynamic Characteristics of Reusable Launch Vehicles," *Open Journal of Fluid Dynamics*, vol. 06, pp. 222–233, Jan. 1, 2016. DOI: 10.4236/ojfd.2016.63017.

[87] S. Salehizadeh, P. Yadmellat, and M. Menhaj, "Local Optima Avoidable Particle Swarm Optimization," in *2009 IEEE Swarm Intelligence Symposium*, Mar. 2009, pp. 16–21. DOI: 10.1109/SIS.2009.4937839.

[88] M. Mitchell, *An Introduction to Genetic Algorithms.* Cambridge, MA: MIT Press, 1996.

[89] A. F. Gad. "PyGAD: An Intuitive Genetic Algorithm Python Library." arXiv: 2106.06158 [cs, math]. (Jun. 11, 2021), preprint.

[90] J. Li and M. Zhang, "Data-based approach for wing shape design optimization," *Aerospace Science and Technology*, vol. 112, p. 106 639, Mar. 1, 2021. DOI: 10.1016/j.ast.2021.106639.

[91] R.-E. Plessix, "A review of the adjoint-state method for computing the gradient of a functional with geophysical applications," *Geophysical Journal International*, vol. 167, pp. 495–503, Nov. 1, 2006. DOI: 10.1111/j.1365-246X.2006.02978.x.

[92] S. Ruder. "An overview of gradient descent optimization algorithms." arXiv: 1609.04747 [cs]. (Jun. 15, 2017), preprint.

[93] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, Jan. 1, 1999, ISSN: 0893-6080. DOI: 10.1016/S0893-6080(98)00116-6.

[94] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011, ISSN: 1533-7928.

[95] M. D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method." arXiv: 1212.5701 [cs]. (Dec. 22, 2012), preprint.

[96] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." arXiv: 1412.6980 [cs]. (Jan. 29, 2017), preprint.

[97] T. Dozat, "Incorporating Nesterov Momentum into Adam," in *Proceedings of the 4th International Conference on Learning Representations*, Feb. 18, 2016.

[98] J. Raphson, *Analysis Æequationum Universalis.* Thomas Bradyll, 1697.

[99] H. H. Tan and K. H. Lim, "Review of second-order optimization techniques in artificial neural networks backpropagation," *IOP Conference Series: Materials Science and Engineering*, vol. 495, no. 1, p. 012 003, Apr. 2019, ISSN: 1757-899X. DOI: 10.1088/1757-899X/495/1/012003.

[100] R. Fletcher, *Practical Methods of Optimization.* New York: John Wiley & Sons, Ltd, 1987.

[101] D. Kraft, *A Software Package for Sequential Quadratic Programming.* Wiss. Berichtswesen d. DFVLR, 1988, 33 pp. Google Books: 4rKaGwAACAAJ.

[102] A. Lam. "BFGS in a Nutshell: An Introduction to Quasi-Newton Methods," Medium. (Feb. 22, 2022).

[103] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 3 Mar. 2020, ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2. [Online]. Available: https://www.nature.com/articles/s41592-019-0686-2 (visited on 11/15/2023).

[104] D. McLean, "Continuum Fluid Mechanics and the Navier-Stokes Equations," in Nov. 1, 2012, pp. 13–77, ISBN: 978-1-119-96751-4. DOI: 10.1002/9781118454190.ch3.

[105] J. R. Cannon, *The One-Dimensional Heat Equation*, ser. Encyclopedia of Mathematics and Its Applications. Cambridge: Cambridge University Press, 1984, ISBN: 978-0-521-30243-2. DOI: 10.1017/CBO9781139086967.

[106] D. P. Hampshire, "A derivation of Maxwell's equations using the Heaviside notation," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 376, no. 2134, Oct. 29, 2018. DOI: 10.1098/rsta.2017.0447.

[107] R. Courant, K. Friedrichs, and H. Lewy, "Über die partiellen Differenzengleichungen der mathematischen Physik," *Mathematische Annalen*, pp. 32–74, Dec. 1, 1928, ISSN: 1432-1807. DOI: 10.1007/BF01448839.

[108] O. C. Zienkiewicz, *The Finite Element Method in Engineering Science*. McGraw-Hill, 1971, 548 pp., ISBN: 978-0-07-094138-0.

[109] C. Hirsch, "- Introduction: An Initial Guide to CFD and to this Volume," in *Numerical Computation of Internal and External Flows (Second Edition)*, C. Hirsch, Ed., Oxford: Butterworth-Heinemann, Jan. 1, 2007, pp. 1–V, ISBN: 978-0-7506-6594-0. DOI: 10.1016/B978-075066594-0/50039-4.

[110] J. Carson, "Model verification and validation," presented at the Winter Simulation Conference Proceedings, vol. 1, Jan. 1, 2002, pp. 52–58. DOI: 10.1109/WSC.2002.1172868.

[111] R. G. Sargent, "Verification and validation of simulation models," in *Proceedings of the 2010 Winter Simulation Conference*, Dec. 2010, pp. 166–183. DOI: 10.1109/WSC.2010.5679166.

[112] T. Maric, D. Gläser, J.-P. Lehr, *et al.* "A Research Software Engineering Workflow for Computational Science and Engineering." arXiv: 2208.07460 [cs]. (Aug. 15, 2022), preprint.

[113] J. Cantu. "What's the Business Value of a One-Platform Approach?" (Sep. 1, 2021), [Online]. Available: https://www.placetechnology.com/blog/whats-the-business-value-of-a-one-platform-approach (visited on 03/08/2023).

[114] "Ansys Workbench — Simulation Integration Platform." (), [Online]. Available: https://www.ansys.com/en-gb/products/ansys-workbench (visited on 03/08/2023).

[115] I. Pan, L. R. Mason, and O. K. Matar, "Data-centric Engineering: Integrating simulation, machine learning and statistics. Challenges and opportunities," *Chemical Engineering Science*, vol. 249, p. 117 271, Feb. 15, 2022, ISSN: 0009-2509. DOI: 10.1016/j.ces.2021.117271.

[116] M. D. Wilkinson, M. Dumontier, IJ. J. Aalbersberg, *et al.*, "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific Data*, vol. 3, no. 1, p. 160 018, 1 Mar. 15, 2016, ISSN: 2052-4463. DOI: 10.1038/sdata.2016.18.

[117] J. Rathod. "Underfitting, Overfitting, and Regularization," Jash Rathod. (Sep. 30, 2021), [Online]. Available: `https://jashrathod.github.io/2021-09-30-underfitting-overfitting-and-regularization/` (visited on 03/26/2023).

[118] A. Paszke, S. Gross, F. Massa, *et al.* "PyTorch: An Imperative Style, High-Performance Deep Learning Library." arXiv: `1912.01703 [cs, stat]`. (Dec. 3, 2019), preprint.

[119] R. O'Conner. "PyTorch vs TensorFlow in 2023," News, Tutorials, AI Research. (Dec. 14, 2021), [Online]. Available: `https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/` (visited on 03/06/2023).

[120] I. Goodfellow, *Deep Learning.* MIT Press, 2016, 200 - 201.

[121] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima." arXiv: `1609.04836 [cs, math]`. (Feb. 9, 2017), preprint.

[122] Y. Dimopoulos, P. Bourret, and S. Lek, "Use of some sensitivity criteria for choosing networks with good generalization ability," *Neural Processing Letters*, vol. 2, no. 6, pp. 1–4, Dec. 1, 1995, ISSN: 1573-773X. DOI: `10.1007/BF02309007`.

[123] Neale, *Bayesian Learning for Neural Networks.* New York: Springer- Verlag, 1996.

[124] C. E. Rasmussen, *Gaussian Processes for Machine Learning.* 2006.

[125] M. L. Stein, *Interpolation of Spatial Data.* New York: Springer- Verlag, 1999.

[126] E. Bonilla, K. Chai, and C. Williams, "Multi-task Gaussian Process Prediction.," Jan. 1, 2007.

[127] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration." arXiv: `1809.11165 [cs, stat]`. (Jun. 29, 2021), preprint.

[128] R. E. Bellman, *Dynamic Programming.* Princton University Press, 1957, ISBN: 978-0-691-07951-6.

[129] G. Lebanon, *Probability: The Analysis of Data, Volume 1.* CreateSpace Independent Publishing Platform: CreateSpace Independent Publishing Platform, Oct. 9, 2012, 346 pp., ISBN: 978-1-4793-4476-5.

[130] M. Pharr, *Physically Based Rendering: From Theory to Implementation*, 2nd edition. Burlington, MA: Morgan Kaufmann, Aug. 26, 2010, 1240 pp., ISBN: 978-0-12-375079-2.

[131] S. E. Davis, S. Cremaschi, and M. R. Eden, "Efficient Surrogate Model Development: Impact of Sample Size and Underlying Model Dimensions," in *Computer Aided Chemical Engineering*, vol. 44, Elsevier, 2018, pp. 979–984, ISBN: 978-0-444-64241-7. DOI: 10.1016/B978-0-444-64241-7.50158-0.

[132] B. C. Williams, "Novel Tool for Selecting Surrogate Modeling Techniques for Surface Approximation," *Computer aided chemical engineering*, vol. 50, M. A. Turkay, Ed., Jan. 2021. DOI: 10.1016/B978-0-323-88506-5.50071-1.

[133] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and Analysis of Computer Experiments," *Statistical Science*, vol. 4, no. 4, pp. 409–423, Nov. 1989, ISSN: 0883-4237, 2168-8745. DOI: 10.1214/ss/1177012413.

[134] J. N. Fuhg, A. Fau, and U. Nackenhorst, "State-of-the-Art and Comparative Review of Adaptive Sampling Methods for Kriging," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 2689–2747, Jun. 1, 2021, ISSN: 1886-1784. DOI: 10.1007/s11831-020-09474-6.

[135] K. Crombecq, D. Gorissen, D. Deschrijver, and T. Dhaene, "A Novel Hybrid Sequential Design Strategy for Global Surrogate Modeling of Computer Experiments," *SIAM J. Scientific Computing*, vol. 33, pp. 1948–1974, Jan. 1, 2011. DOI: 10.1137/090761811.

[136] C. Lam, "Sequential adaptive designs in computer experi- ments for response surface model fi," The Ohio State University, 2008.

[137] J. Eason and S. Cremaschi, "Adaptive sequential sampling for surrogate model generation with artificial neural networks," *Computers & Chemical Engineering*, vol. 68, pp. 220–232, Sep. 4, 2014, ISSN: 0098-1354. DOI: 10.1016/j.compchemeng.2014.05.021.

[138] “External heating,” ITER. (), [Online]. Available: `http://www.iter.org/sci/plasmaheating` (visited on 03/25/2023).

[139] S. Wurzel. “Measuring Progress in Fusion Energy: The Triple Product.” (2019), [Online]. Available: `https://www.fusionenergybase.com/article/measuring-progress-in-fusion-energy-the-triple-products` (visited on 01/03/2023).

[140] “Fusion world — JET makes history, again,” ITER. (), [Online]. Available: `http://www.iter.org/newsline/-/3722` (visited on 10/28/2022).

[141] A. F. “ITER - world’s most complex machine takes shape,” WordlessTech. (Dec. 8, 2017), [Online]. Available: `https://wordlesstech.com/iter-worlds-most-complex-machine-takes-shape/` (visited on 03/06/2023).

[142] “Tokamak,” ITER. (), [Online]. Available: `http://www.iter.org/mach/tokamak` (visited on 03/25/2023).

[143] “ITER - the way to new energy,” ITER. (), [Online]. Available: `http://www.iter.org` (visited on 03/06/2023).

[144] “The divertor,” ITER. (), [Online]. Available: `http://www.iter.org/mach/divertor` (visited on 01/04/2023).

[145] E. Pajuste, G. Kizane, L. Avotina, A. Vitins, and A. S. Teimane, “Tritium retention in plasma facing materials of JET ITER-Like-Wall retrieved from the vacuum vessel in 2012 (ILW1), 2014 (ILW2) and 2016 (ILW3),” *Nuclear Materials and Energy*, vol. 27, p. 101 001, Jun. 1, 2021, ISSN: 2352-1791. DOI: `10.1016/j.nme.2021.101001`.

[146] A. Widdowson, E. Alves, A. Baron-Wiechec, *et al.*, “Overview of the JET ITER-like wall divertor,” *Nuclear Materials and Energy*, Proceedings of the 22nd International Conference on Plasma Surface Interactions 2016, 22nd PSI, vol. 12, pp. 499–505, Aug. 1, 2017, ISSN: 2352-1791. DOI: `10.1016/j.nme.2016.12.008`.

[147] A. J. H. Donné, “The European roadmap towards fusion electricity,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 377, no. 2141, p. 20 170 432, Mar. 25, 2019. DOI: `10.1098/rsta.2017.0432`.

[148] Jacquireid. “Why our spherical tokamak design holds the key to commercial fusion energy,” Tokamak Energy. (Feb. 19, 2021), [Online]. Available: `https://www.tokamakenergy.co.uk/2021/02/19/why-our-spherical-tokamak-design-holds-the-key-to-commercial-fusion-energy/` (visited on 01/04/2023).

[149] “STEP - Spherical Tokamak for Energy Production,” STEP. (), [Online]. Available: `https://step.ukaea.uk/` (visited on 03/06/2023).

[150] “MAST Upgrade,” Culham Centre for Fusion Energy. (), [Online]. Available: `https://ccfe.ukaea.uk/research/mast-upgrade/` (visited on 03/06/2023).

[151] A. Litnovsky, V. Philipps, P. Wienhold, *et al.*, “Experimental investigations of castellated monoblock structures in TEXTOR,” *Journal of Nuclear Materials*, PSI-16, vol. 337–339, pp. 917–921, Mar. 1, 2005, ISSN: 0022-3115. DOI: `10.1016/j.jnucmat.2004.09.064`.

[152] M. Guida, F. Marulo, M. Meo, and S. Russo, “Certification by birdstrike analysis on C27J fullscale ribless composite leading edge,” *International Journal of Impact Engineering*, vol. 54, pp. 105–113, Apr. 1, 2013, ISSN: 0734-743X. DOI: `10.1016/j.ijimpeng.2012.10.002`.

[153] G. De Temmerman, M. A. van den Berg, J. Scholten, *et al.*, “High heat flux capabilities of the Magnum-PSI linear plasma device,” *Fusion Engineering and Design*, Proceedings of the 27th Symposium On Fusion Technology (SOFT-27); Liège, Belgium, September 24-28, 2012, vol. 88, no. 6, pp. 483–487, Oct. 1, 2013, ISSN: 0920-3796. DOI: `10.1016/j.fusengdes.2013.05.047`.

[154] H. Greuner, H. Bolt, B. Böswirth, *et al.*, “Design, performance and construction of a 2MW ion beam test facility for plasma facing components,” *Fusion Engineering and Design*, Proceedings of the 23rd Symposium of Fusion Technology, vol. 75–79, pp. 345–350, Nov. 1, 2005, ISSN: 0920-3796. DOI: `10.1016/j.fusengdes.2005.06.021`.

[155] “Prototype — The hottest stuff in ITER,” ITER. (), [Online]. Available: `http://www.iter.org/newsline/-/3027` (visited on 03/24/2023).

[156] V. Kuznetsov, A. Gorbenko, V. Davydov, *et al.*, "Status of the IDTF high-heat-flux test facility," *Fusion Engineering and Design*, Proceedings of the 11th International Symposium on Fusion Nuclear Technology-11 (ISFNT-11) Barcelona, Spain, 15-20 September, 2013, vol. 89, no. 7, pp. 955–959, Oct. 1, 2014, ISSN: 0920-3796. DOI: `10.1016/j.fusengdes.2014.04.064`.

[157] P. Majerus, R. Duwe, T. Hirai, W. Kühnlein, J. Linke, and M. Rödig, "The new electron beam test facility JUDITH II for high heat flux experiments on plasma facing components," *Fusion Engineering and Design*, Proceedings of the 23rd Symposium of Fusion Technology, vol. 75–79, pp. 365–369, Nov. 1, 2005, ISSN: 0920-3796. DOI: `10.1016/j.fusengdes.2005.06.058`.

[158] "CHIMERA," Culham Centre for Fusion Energy. (), [Online]. Available: `https://ccfe.ukaea.uk/divisions/fusion-technology/chimera/` (visited on 01/04/2023).

[159] D. Hancock, "Employing Additive Manufacturing for Fusion High Heat Flux Structures," Ph.D. dissertation, University of Sheffield, Jun. 26, 2018. [Online]. Available: `https://etheses.whiterose.ac.uk/23009/` (visited on 03/06/2023).

[160] A. vander Vorst, A. Rosen, and Y. Kotsuka, *RF/Microwave Interaction with Biological Tissues*, ser. Wiley Series in Microwave and Optical Engineering. Hoboken, N.J: John Wiley & Sons : IEEE, 2006, 330 pp., ISBN: 978-0-471-73277-8.

[161] H. Lamb and J. W. L. Glaisher, "XIII. On electrical motions in a spherical conductor," *Philosophical Transactions of the Royal Society of London*, vol. 174, pp. 519–549, 1883. DOI: `10.1098/rstl.1883.0013`.

[162] S. L. Semiatin, *Elements of Induction Heating: Design, Control, and Applications*. ASM International, Jan. 1, 1988, 342 pp., ISBN: 978-1-61503-198-6. Google Books: `zXmTLYwO3McC`.

[163] F. Vidal, M. Garnier, N. Freud, J. M. Létang, and N. W. John, *Simulation of X-ray Attenuation on the GPU*. The Eurographics Association, 2009, ISBN: 978-3-905673-71-5. DOI: `10.2312/LocalChapterEvents/TPCG/TPCG09/025-032`.

[164] B. Thorpe, *Cad2Vox*, 2022. [Online]. Available: `https://github.com/bjthorpe/Cad2vox`.

[165] *SALOME: The open-source platform for numerical simulation*, version 6.7.0. [Online]. Available: `https://www.salome-platform.org/`.

[166] *Code_aster: Structures andthermomechanics analysis for studies and research*, version 14.4. [Online]. Available: `https://code-aster.org/V2/spip.php?rubrique2`.

[167] R. Otin, "ERMES: A nodal-based finite element code for electromagnetic simulations in frequency domain," *Computer Physics Communications*, vol. 184, no. 11, pp. 2588–2595, Nov. 1, 2013, ISSN: 0010-4655. DOI: `10.1016/j.cpc.2013.06.010`.

[168] *OpenMP*. [Online]. Available: `https://www.openmp.org/`.

[169] *GiD*. [Online]. Available: `https://www.gidsimulation.com/`.

[170] *H5py: Pythonic interface to the HDF5 binary data format.* [Online]. Available: `https://www.h5py.org/`.

[171] namespace, *Python namespaces.* [Online]. Available: `https://docs.python.org/3/library/types.html#types.SimpleNamespace`.

[172] *Socket: Low-level networking interface.* [Online]. Available: `https://docs.python.org/3/library/socket.html`.

[173] G. S. Almasi and A. Gottlieb, *Highly Parallel Computing.* Benjamin/Cummings, 1989, 552 pp., ISBN: 978-0-8053-0177-9.

[174] L. M. Evans, T. Minniti, T. Barrett, A. von Müller, and L. Margetts, "Virtual qualification of novel heat exchanger components with the image-based finite element method," in *Proceedings of 9th Conference on Industrial Computed Tomography (iCT), Padova, Italy*, 2019.

[175] L. Bieker-Walz, D. Krajzewicz, A. Morra, C. Michelacci, and F. Cartolano, "Traffic Simulation for All: A Real World Traffic Scenario from the City of Bologna," *Lecture Notes in Control and Information Sciences*, vol. 13, pp. 47–60, Mar. 12, 2015, ISSN: 978-3-319-15023-9. DOI: `10.1007/978-3-319-15024-6_4`.

[176] R. Van Loon and F. N. Van de Vosse, *Fluid–structure interaction in biomedical applications*, Wiley Online Library, 2010.

[177] M. M. McKerns and M. A. G. Aivazis, *Pathos: A framework for heterogeneous computing*, version 0.2.7, 2010. [Online]. Available: `https://uqfoundation.github.io/project/pathos`.

[178] M. M. McKerns, L. Strand, T. Sullivan, A. Fang, and M. A. G. Aivazis. "Building a Framework for Predictive Science." arXiv: `1202.1056 [cs]`. (Feb. 6, 2012), preprint.

[179] *Threading: Thread-based parallelism.* [Online]. Available: `https://docs.python.org/3/library/threading.html`.

[180] *Multiprocessing: Process-based parallelism.* [Online]. Available: `https://docs.python.org/3/library/multiprocessing.html`.

[181] *Parallel python*, version 1.6.6.3. [Online]. Available: `https://www.parallelpython.com/`.

[182] "Message Passing Interface :: High Performance Computing." (), [Online]. Available: `https://hpc.nmsu.edu/discovery/mpi/introduction/` (visited on 03/08/2023).

[183] *Slurm: Workload manager*, version 23.02. [Online]. Available: `https://slurm.schedmd.com/overview.html`.

[184] "About Sunbird – Supercomputing Wales Portal." (), [Online]. Available: `https://portal.supercomputing.wales/index.php/about-sunbird/` (visited on 03/26/2023).

[185] F. T. Ulaby, *Fundamentals of Applied Electromagnetics*, 5th edition. Upper Saddle River, NJ: Pearson College Div, Jan. 1, 2006, 448 pp., ISBN: 978-0-13-241326-8.

[186] J. C. Maxwell, *A Treatise on Electricity and Magnetism, Vol. II.* Oxford University Press, 1904.

[187] "CWT Rogowski coil — PEM." (), [Online]. Available: `http://www.pemuk.com/products/cwt-current-probe/cwt.aspx` (visited on 03/09/2023).

[188] "RS PRO 2205A PC Based Oscilloscope." (), [Online]. Available: `https://uk.rs-online.com/web/p/oscilloscopes/1796185` (visited on 03/09/2023).

[189] H. Sönnerlind. "Singularities in Finite Element Models: Dealing with Red Spots," COMSOL. (2015).

[190] N. Stevens. “Practical Tips for Dealing with Stress Singularities — Nick J Stevens,” Medium. (Sep. 12, 2020), [Online]. Available: https://medium.com/@nickjstevens/practical-tips-for-dealing-with-stress-singularities-nick-j-stevens-1fe6a57d55a9.

[191] A. Rizzi and J. M. Luckring, “Historical development and use of CFD for separated flow simulations relevant to military aircraft,” *Aerospace Science and Technology*, vol. 117, p. 106 940, Oct. 1, 2021, ISSN: 1270-9638. DOI: 10.1016/j.ast.2021.106940.

[192] M. Aultman, Z. Wang, R. Auza-Gutierrez, and L. Duan, “Evaluation of CFD methodologies for prediction of flows around simplified and complex automotive models,” *Computers & Fluids*, vol. 236, p. 105 297, Mar. 30, 2022, ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2021.105297.

[193] V. Viitanen, T. Sipilä, A. Sánchez-Caja, and T. Siikonen, “CFD predictions of unsteady cavitation for a marine propeller in oblique inflow,” *Ocean Engineering*, vol. 266, p. 112 596, Dec. 15, 2022, ISSN: 0029-8018. DOI: 10.1016/j.oceaneng.2022.112596.

[194] M. Gracka, R. Lima, J. M. Miranda, S. Student, B. Melka, and Z. Ostrowski, “Red blood cells tracking and cell-free layer formation in a microchannel with hyperbolic contraction: A CFD model validation,” *Computer Methods and Programs in Biomedicine*, vol. 226, p. 107 117, Nov. 1, 2022, ISSN: 0169-2607. DOI: 10.1016/j.cmpb.2022.107117.

[195] “Water Boiling Graph Curve at 1 Atmosphere.” (), [Online]. Available: https://www.engineersedge.com/heat_transfer/water_boiling_graph_curve_13825.htm (visited on 03/10/2023).

[196] “International Association for the Properties of Water and Steam.” (), [Online]. Available: http://www.iapws.org/ (visited on 03/10/2023).

[197] W. Wagner, J. R. Cooper, A. Dittmann, *et al.*, “The IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam,” *Journal of Engineering for Gas Turbines and Power*, vol. 122, no. 1, pp. 150–184, Jan. 1, 2000, ISSN: 0742-4795, 1528-8919. DOI: 10.1115/1.483186. [Online]. Available: https://asmedigitalcollection.asme.org/gasturbinespower/article/122/1/150/461340/The-IAPWS-Industrial-Formulation-1997-for-the (visited on 03/10/2023).

[198] R. H. S. Winterton, “Where did the Dittus and Boelter equation come from?” *International Journal of Heat and Mass Transfer*, vol. 41, no. 4, pp. 809–810, Feb. 1, 1998, ISSN: 0017-9310. DOI: 10.1016/S0017-9310(97)00177-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0017931097001774 (visited on 03/10/2023).

[199] K. Robinson, J. G. Hawley, G. P. Hammond, and N. J. Owen, “Convective coolant heat transfer in internal combustion engines,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 217, no. 2, pp. 133–146, Feb. 1, 2003, ISSN: 0954-4070. DOI: 10.1177/095440700321700207. [Online]. Available: https://doi.org/10.1177/095440700321700207 (visited on 07/20/2022).

[200] E. N. Sieder and G. E. Tate, “Heat Transfer and Pressure Drop of Liquids in Tubes,” *Industrial & Engineering Chemistry*, vol. 28, no. 12, pp. 1429–1435, Dec. 1936, ISSN: 0019-7866, 1541-5724. DOI: 10.1021/ie50324a027. [Online]. Available: https://pubs.acs.org/doi/abs/10.1021/ie50324a027 (visited on 03/10/2023).

[201] A. E. Bergles and W. M. Rohsenow, “The Determination of Forced-Convection Surface-Boiling Heat Transfer,” *Journal of Heat Transfer*, vol. 86, no. 3, pp. 365–372, Aug. 1, 1964, ISSN: 0022-1481. DOI: 10.1115/1.3688697. [Online]. Available: https://doi.org/10.1115/1.3688697 (visited on 03/10/2023).

[202] X. Fang, Y. Yuan, A. Xu, L. Tian, and Q. Wu, “Review of correlations for subcooled flow boiling heat transfer and assessment of their applicability to water,” *Fusion Engineering and Design*, vol. 122, pp. 52–63, Nov. 1, 2017, ISSN: 0920-3796. DOI: 10.1016/j.fusengdes.2017.09.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920379617308086 (visited on 07/18/2022).

[203] M. Araki, M. Ogawa, T. Kunugi, K. Satoh, and S. Suzuki, “Experiments on heat transfer of smooth and swirl tubes under one-sided heating conditions,” *International Journal of Heat and Mass Transfer*, vol. 39, no. 14, pp. 3045–3055, Sep. 1, 1996, ISSN: 0017-9310. DOI: 10.1016/0017-9310(95)00344-4.

[204] "Analysis of the JAERI critical heta flux database for fusion application." (), [Online]. Available: https://www.osti.gov/etdeweb/servlets/purl/537993 (visited on 12/06/2021).

[205] F. Inasaka and H. Nariai, "Critical Heat Flux and Flow Characteristics of Subcooled Flow Boiling in Narrow Tubes," *JSME international journal*, vol. 30, no. 268, pp. 1595–1600, 1987. DOI: 10.1299/jsme1987.30.1595.

[206] L. S. Tong, "An Evaluation of the Departure from Nucleate Boiling in Bundles of Reactor Fuel Rods," *Nuclear Science and Engineering*, vol. 33, no. 1, pp. 7–15, Jul. 1, 1968, ISSN: 0029-5639. DOI: 10.13182/NSE68-A20912. [Online]. Available: https://doi.org/10.13182/NSE68-A20912 (visited on 03/13/2023).

[207] D. Knuth, *Art of Computer Programming, The: Sorting and Searching, Volume 3: 03*, 2nd edition. Boston: Addison-Wesley Professional, Jun. 30, 1998, 800 pp., ISBN: 978-0-201-89685-5.

[208] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A Comparative Study of Efficient Initialization Methods for the K-Means Clustering Algorithm," *Expert Systems with Applications*, vol. 40, no. 1, pp. 200–210, Jan. 2013, ISSN: 09574174. DOI: 10.1016/j.eswa.2012.07.021. arXiv: 1209.1960 [cs]. [Online]. Available: http://arxiv.org/abs/1209.1960 (visited on 03/13/2023).

[209] G. Jenks, "The Data Model Concept in Statistical Mapping," 1967.

[210] *SciKit: K-means.* [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

[211] L. Liu, J. Sun, W. Chen, and P. Sun, "Study on the machining distortion of aluminum alloy parts induced by forging residual stresses," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 231, no. 4, pp. 618–627, Mar. 2017, ISSN: 0954-4054, 2041-2975. DOI: 10.1177/0954405415583805. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0954405415583805 (visited on 03/26/2023).

[212] D. Stathakis, "How many hidden layers and nodes?" *International Journal of Remote Sensing*, vol. 30, no. 8, pp. 2133–2147, Apr. 20, 2009, ISSN: 0143-1161. DOI: 10.1080/01431160802549278. [Online]. Available: https://doi.org/10.1080/01431160802549278 (visited on 03/14/2023).

[213] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[214] S. SHARMA. "Activation Functions in Neural Networks," Medium. (Nov. 20, 2022), [Online]. Available: `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6` (visited on 03/29/2023).

[215] "AMAZE: Building confidence in additive manufacturing.," AMAZE. (), [Online]. Available: `https://www.the-mtc.org/what-we-do/projects/amaze/` (visited on 03/29/2023).

[216] J. J. Grefenstette, "Optimization of Control Parameters for Genetic Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 122–128, Jan. 1986, ISSN: 2168-2909. DOI: `10.1109/TSMC.1986.289288`.

[217] G. Ayers, "Cylindrical thermal contact conductance," Texas A&M, 2003. [Online]. Available: `https://core.ac.uk/download/pdf/4267977.pdf`.

[218] "Tensile Testing Machines — An Introduction." (), [Online]. Available: `https://www.instron.com/en-gb/resources/test-types/tensile-test` (visited on 03/06/2023).

[219] Ll. M. Evans, L. Margetts, V. Casalegno, *et al.*, "Transient thermal finite element analysis of CFC–Cu ITER monoblock using X-ray tomography data," *Fusion Engineering and Design*, vol. 100, pp. 100–111, Nov. 1, 2015, ISSN: 0920-3796. DOI: `10.1016/j.fusengdes.2015.04.048`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S092037961500277X` (visited on 03/08/2023).

Appendix A

VirtualLab run files

This appendix includes the *VirtualLab* run files which are used to perform the analysis in chapter 6, 7 and 8. These are also included in the *VirtualLab* documentation and make up part of the tutorials [43], [44].

```
#!/usr/bin/env python3

import requests
import os
from types import SimpleNamespace as Namespace
from Scripts.Common.VirtualLab import VLSetup


CoilType = 'Pancake'
ModelType = 'GPR' # this can be GPR or MLP
CreateModel = True
PVAnalysis = True

# ====================================================================
# Setup VirtualLab
VirtualLab=VLSetup('HIVE','ML_analysis')

VirtualLab.Settings(Launcher='sequential',NbJobs=1,Mode='t')

# ====================================================================
# check data has been created, if not download it
DataFile = '{}_coil/PowerVariation.hdf'.format(CoilType)
if not VirtualLab.InProject(DataFile):
    DataFileFull = "{}/{}".format(VirtualLab.GetProjectDir(),DataFile)
    print("Data doesn't exist, so downloading.")
    r = requests.get('https://zenodo.org/record/8300663/files/PowerVariation.hdf')
    os.makedirs(os.path.dirname(DataFileFull),exist_ok=True)
    with open(DataFileFull,'wb') as f:
        f.write(r.content)

# ====================================================================
# Create ML model
if ModelType=='MLP' and CreateModel:
    # Create three MLP models with different architectures and compare their performance
    main_parameters = Namespace()
    var_parameters = Namespace()

    ML = Namespace()
    ML.File = ('NN_Models','MLP_hdf5')
    ML.TrainingParameters = {'Epochs':1000,'lr':0.05}
    ML.TrainData = [DataFile, 'Features', [['Power'],['Variation']],{'group':'Train'}]
    ML.ValidationData = [DataFile, 'Features', [['Power'],['Variation']],{'group':'Test'}] #
        data used to monitor for overfitting
    ML.Seed = 100 # initial weights of MLP are randomised so this ensures reproducability
```

```python
    main_parameters.ML = ML

    Architectures = [[32,32],[16,32,16],[8,16,8,4]] # the hidden layers of the MLP
    ML = Namespace(Name = [],ModelParameters=[])
    for architecture in Architectures:
        ML.ModelParameters.append({'Architecture':architecture})
        arch_str = '_'.join(map(str,architecture)) # convert architecture to string and save
            under that name
        ML.Name.append("PV/{}/MLP/{}".format(CoilType,arch_str))
    var_parameters.ML = ML

    DA = Namespace()
    DA.Name = "Analysis/{}/PowerVariation/MLP_Compare".format(CoilType) # results will be
        saved to same directory as before
    DA.File = ['PowerVariation','MLP_compare']
    DA.MLModels = var_parameters.ML.Name # use the models defined earlier
    DA.TestData = [DataFile, 'Features', [['Power'],['Variation']],{'group':'Test'}] # unseen
         data to analyse performance
    main_parameters.DA = DA

    VirtualLab.Parameters(main_parameters,var_parameters)

    # generate MLP models
    VirtualLab.ML()
    # analyse performance of MLP model
    VirtualLab.DA()

elif ModelType=='GPR' and CreateModel:
    # Create three GPR models each with different kernels and compare their performance
    main_parameters = Namespace()
    var_parameters = Namespace()

    # parameters used to generate model
    ML = Namespace()
    ML.File = ('GPR_Models','GPR_hdf5')
    ML.TrainingParameters = {'Epochs':1000,'lr':0.05}
    ML.TrainData = [DataFile, 'Features', [['Power'],['Variation']],{'group':'Train'}]
    main_parameters.ML = ML

    GPR_kernels = ['RBF','Matern_1.5','Matern_2.5']
    ML = Namespace(Name = [],ModelParameters=[])
    for kernel in GPR_kernels:
        ML.ModelParameters.append({'kernel':kernel})
        ML.Name.append("PV/{}/GPR/{}".format(CoilType,kernel))
    var_parameters.ML = ML

    # parameters used to compare models
    DA = Namespace()
    DA.Name = "Analysis/{}/PowerVariation/GPR_Compare".format(CoilType)
    DA.File = ['PowerVariation','GPR_compare']
    DA.MLModels = var_parameters.ML.Name # use the models defined above
    DA.TestData = [DataFile, 'Features', [['Power'],['Variation']],{'group':'Test'}] # unseen
         data to analyse performance
    main_parameters.DA = DA

    VirtualLab.Parameters(main_parameters,var_parameters)

    # generate GPR models
    VirtualLab.ML()
    # compare accuracy of the three models
    VirtualLab.DA()

elif CreateModel:
    raise ValueError("Unknown ModelType '{}'. this must either be 'GPR' or 'MLP".format(
        ModelType))

# ==============================================================
# create performance envelope of power versus variation
if PVAnalysis:
    if ModelType=='GPR':
        DA = Namespace()
        DA.Name = "Analysis/{}/PowerVariation/GPR_Analysis".format(CoilType)
        DA.File = ['PowerVariation','Insight_GPR']
        DA.MLModel = "PV/{}/GPR/Matern_2.5".format(CoilType) # chose a single model to gain insight from
        main_parameters = Namespace(DA=DA)

    elif ModelType=='MLP':
        DA = Namespace()
        DA.Name = "Analysis/{}/PowerVariation/MLP_Analysis".format(CoilType)
        DA.File = ['PowerVariation','Insight_MLP']
        DA.MLModel = "PV/{}/MLP/32_32".format(CoilType) # chose a single model to gain insight from
        main_parameters = Namespace(DA=DA)

    VirtualLab.Parameters(main_parameters)

    VirtualLab.DA()
```

Listing A.1: Run file used to perform coil configuration optimisation

```python
#!/usr/bin/env python3

import requests
import os
from types import SimpleNamespace as Namespace
from Scripts.Common.VirtualLab import VLSetup

CoilType='Pancake'
PCA_Analysis = False
ModelType = 'GPR' # this can be GPR or MLP
CreateModel = True
InverseAnalysis = True

GUI = True

# ====================================================================
# Setup VirtualLab
VirtualLab=VLSetup('HIVE','ML_analysis')

VirtualLab.Settings(Launcher='sequential',NbJobs=1,Mode='t')

# ====================================================================
# check data has been created
DataFile = '{}_coil/TempNodal.hdf'.format(CoilType)
if not VirtualLab.InProject(DataFile):
    print("Data doesn't exist, so downloading. This may take a while")
    # download data
    DataFileFull = "{}/{}".format(VirtualLab.GetProjectDir(),DataFile)
    r = requests.get('https://zenodo.org/record/8300663/files/TempNodal.hdf')
    os.makedirs(os.path.dirname(DataFileFull),exist_ok=True)
    with open(DataFileFull,'wb') as f:
        f.write(r.content)
    # download mesh
    r = requests.get('https://zenodo.org/record/8300663/files/HIVE_component.med')
    os.makedirs(VirtualLab.Mesh.OutputDir,exist_ok=True)
    with open("{}/HIVE_component.med".format(VirtualLab.Mesh.OutputDir),'wb') as f:
        f.write(r.content)

# ====================================================================
# calculate reconstruction error vs the number of principal components
if PCA_Analysis:
    DA = Namespace()
    DA.Name = 'Analysis/{}/InverseSolution_T/PCA_Sensitivity'.format(CoilType)
    DA.File = ('MLtools','PCA_Sensitivity')
    DA.TrainData = [DataFile, 'Features', 'Temperature',{'group':'Train'}]
    DA.TestData = [DataFile, 'Features', 'Temperature',{'group':'Test'}]

    main_parameters = Namespace(DA=DA)

    VirtualLab.Parameters(main_parameters,RunDA=PCA_Analysis)

    VirtualLab.DA()

# ====================================================================
# Create ML model
if ModelType=='MLP' and CreateModel:
    # Create MLP model
    main_parameters = Namespace()

    ML = Namespace()
    ML.Name = 'Temperature/{}/MLP'.format(CoilType)
    ML.File = ('NN_Models','MLP_PCA_hdf5')
    ML.TrainingParameters = {'Epochs':1000,'lr':0.005}
    ML.TrainData = [DataFile, 'Features', 'Temperature',{'group':'Train'}]
    ML.ValidationData = [DataFile, 'Features', 'Temperature',{'group':'Test'}]
    ML.ModelParameters = {'Architecture':[8,16,8]}
    ML.Seed = 100 # initial weights of MLP are randomised so this ensures reproducability
    ML.Metric = {'nb_components':20}
    main_parameters.ML = ML

    VirtualLab.Parameters(main_parameters)

    # generate GPR models
    VirtualLab.ML()

elif ModelType=='GPR' and CreateModel:
    # Create GPR model
    main_parameters = Namespace()

    ML = Namespace()
    ML.Name = 'Temperature/{}/GPR'.format(CoilType)
    ML.File = ('GPR_Models','GPR_PCA_hdf5')
    ML.TrainingParameters = {'Epochs':1000,'lr':0.05}
    ML.TrainData = [DataFile, 'Features', 'Temperature',{'group':'Train'}]
    ML.ModelParameters = {'kernel':'Matern_2.5','min_noise':1e-8,'noise_init':1e-6}
    ML.Metric = {'nb_components':20}
    main_parameters.ML = ML

    VirtualLab.Parameters(main_parameters)

    # generate GPR models
    VirtualLab.ML()

elif CreateModel:
    raise ValueError("Unknown ModelType '{}'. this must either be 'GPR' or 'MLP'".format(
        ModelType))

# ==================================================================
# Use model to perform analysis
if InverseAnalysis:
    main_parameters = Namespace()

    DA = Namespace()
    if ModelType=='GPR':
        DA.Name = 'Analysis/{}/InverseSolution_T/GPR'.format(CoilType)
        DA.File = ('InverseSolution','AnalysisT_GPR')
        DA.MLModel = 'Temperature/{}/GPR'.format(CoilType)
    elif ModelType=='MLP':
        DA.Name = 'Analysis/{}/InverseSolution_T/MLP'.format(CoilType)
        DA.File = ('InverseSolution','AnalysisT_MLP')
        DA.MLModel = 'Temperature/{}/MLP'.format(CoilType)

    DA.MeshName = 'HIVE_component' # name of the mesh used to generate the analysis (see
        DataCollect.py)
    DA.TestData = [DataFile, 'Features', 'Temperature',{'group':'Test'}]
    # create comparison plots for the following indexes of the test dataset. This can be any
        number(s) from 0 to 299 (the size of the test dataset)
    DA.Index = [2]
    # solve inverse problem for reaching specific temperature
    DA.DesiredTemp = 600
    DA.PVGUI = GUI
    main_parameters.DA = DA

    VirtualLab.Parameters(main_parameters)

    VirtualLab.DA()
```

Listing A.2: Run file used to identify temperature related invserse solutions

```python
#!/usr/bin/env python3
'''
This script demonstrates how the data collected in DataCollect.py
can be used to create 3D surrogate models of the temperature and
Von Mises stress fields, which can be used to identify inverse solutions.

This script assumes that temperature surrogate models have already been generated, see
InverseSolution_T.py for more details on this.
'''

import requests
import os
from types import SimpleNamespace as Namespace
from Scripts.Common.VirtualLab import VLSetup


CoilType='Pancake'
PCA_Analysis = False
ModelType = 'GPR' # this can be GPR or MLP
CreateModel = True
InverseAnalysis = True

GUI = True

# ====================================================================
# Setup VirtualLab
VirtualLab=VLSetup('HIVE','ML_analysis')

VirtualLab.Settings(Launcher='sequential',NbJobs=1,Mode='t')

# ====================================================================
# check data has been created
DataFile = '{}_coil/VMNodal.hdf'.format(CoilType)
if not VirtualLab.InProject(DataFile):
    print("Data doesn't exist, so downloading. This may take a while")
    # download data
    DataFileFull = "{}/{}".format(VirtualLab.GetProjectDir(),DataFile)
    r = requests.get('https://zenodo.org/record/8300663/files/VMNodal.hdf')
    os.makedirs(os.path.dirname(DataFileFull),exist_ok=True)
    with open(DataFileFull,'wb') as f:
        f.write(r.content)

# ====================================================================
# calculate reconstruction error vs the number of principal components
if PCA_Analysis:
    DA = Namespace()
    DA.Name = 'Analysis/{}/InverseSolution_VM/PCA_Sensitivity'.format(CoilType)
    DA.File = ('MLtools','PCA_Sensitivity')
    DA.TrainData = [DataFile, 'Features', 'VonMises',{'group':'Train'}]
    DA.TestData = [DataFile, 'Features', 'VonMises',{'group':'Test'}]

    main_parameters = Namespace(DA=DA)

    VirtualLab.Parameters(main_parameters)

    VirtualLab.DA()

# ====================================================================
# Create ML model
if ModelType=='MLP' and CreateModel:
    # Create MLP model
    main_parameters = Namespace()

    ML = Namespace()
    ML.Name = 'VonMises/{}/MLP'.format(CoilType)
    ML.File = ('NN_Models','MLP_PCA_hdf5')
    ML.TrainingParameters = {'Epochs':1000,'lr':0.005}
    ML.TrainData = [DataFile, 'Features', 'VonMises',{'group':'Train'}]
    ML.ValidationData = [DataFile, 'Features', 'VonMises',{'group':'Test'}]
    ML.ModelParameters = {'Architecture':[8,16,8]}
    ML.Seed = 100 # initial weights of MLP are randomised so this ensures reproducability
    ML.Metric = {'nb_components':20}
    main_parameters.ML = ML

    VirtualLab.Parameters(main_parameters)

    # generate GPR models
    VirtualLab.ML()

elif ModelType=='GPR' and CreateModel:
    # Create GPR model
    main_parameters = Namespace()

    ML = Namespace()
    ML.Name = 'VonMises/{}/GPR'.format(CoilType)
    ML.File = ('GPR_Models','GPR_PCA_hdf5')
    ML.TrainingParameters = {'Epochs':1000,'lr':0.05}
    ML.TrainData = [DataFile, 'Features', 'VonMises',{'group':'Train'}]
    ML.ModelParameters = {'kernel':'Matern_2.5','min_noise':1e-8,'noise_init':1e-6}
    ML.Metric = {'nb_components':20}
    main_parameters = Namespace(ML=ML)

    VirtualLab.Parameters(main_parameters)

    # generate GPR models
    VirtualLab.ML()

# ==================================================================
# Use models (Von Mises and temperature) to perform analysis
if InverseAnalysis:
    main_parameters = Namespace()

    DA = Namespace()
    if ModelType=='GPR':
        DA.Name = 'Analysis/{}/InverseSolution_VM/GPR'.format(CoilType)
        DA.File = ('InverseSolution','AnalysisVM_GPR')
        DA.MLModel_T = 'Temperature/{}/GPR'.format(CoilType)
        DA.MLModel_VM = 'VonMises/{}/GPR'.format(CoilType)
    elif ModelType=='MLP':
        DA.Name = 'Analysis/{}/InverseSolution_VM/MLP'.format(CoilType)
        DA.File = ('InverseSolution','AnalysisVM_MLP')
        DA.MLModel_T = 'Temperature/{}/MLP'.format(CoilType)
        DA.MLModel_VM = 'VonMises/{}/MLP'.format(CoilType)

    DA.MeshName = 'HIVE_component' # name of the mesh used to generate the analysis (see
        DataCollect.py)
    DA.TestData = [DataFile, 'Features', 'VonMises',{'group':'Test'}]
    # create comparison plots for the following indexes of the test dataset. This can be any
        numbers up to 300 (the size of the test dataset)
    DA.Index = [2]
    DA.DesiredTemp = 600
    DA.PVGUI = GUI
    main_parameters.DA = DA

    VirtualLab.Parameters(main_parameters)

    VirtualLab.DA()
```

Listing A.3: Run file used to identify Von Mises related invserse solutions

```python
#!/usr/bin/env python3
'''
This script demonstrates how the temperature field surrogate
model generated in InverseSolution.py can be used to predict
the temperature field throughout the component from a handful of
surface thermocouple measurements.
'''

from types import SimpleNamespace as Namespace
from Scripts.Common.VirtualLab import VLSetup

CoilType='Pancake'
ModelType = 'MLP' # this can be GPR or MLP
EstimateField = True
Sensitivity = False
Optimise = False

GUI = True

# ====================================================================
# Setup VirtualLab
VirtualLab=VLSetup('HIVE','ML_analysis')

VirtualLab.Settings(Launcher='sequential',NbJobs=1,Mode='t')

DataFile = '{}_coil/TempNodal.hdf'.format(CoilType) # data already downloaded for previous
      analysis

# ====================================================================
# Identify the full temperature field using only the thermocouples data
# and create plots comparing this with the simulation
if EstimateField:
    main_parameters = Namespace()

    DA = Namespace()
    DA.Name = 'Analysis/{}/Thermocouple/{}/EstimateField'.format(CoilType,ModelType)
    DA.File = ('Thermocouple','FullFieldEstimate_{}'.format(ModelType))
    DA.MLModel = 'Temperature/{}/{}'.format(CoilType,ModelType)
    DA.MeshName = 'HIVE_component' # name of the mesh used to generate the analysis (see
        DataCollect.py)
    DA.TestData = [DataFile, 'Features', 'Temperature',{'group':'Test'}]
    # create comparison plots for the following indexes of the test dataset. This can be any
        numbers up to 300 (the size of the test dataset)
    DA.Index = [7]
    # Location of thermocouples
    DA.ThermocoupleConfig = [['TileSideA',0.5,0.5],
                             ['TileFront',0.5,0.5],
                             ['TileSideB',0.5,0.5],
                             ['TileBack',0.5,0.5],
                             ['BlockFront',0.5,0.5],
                             ['BlockBack',0.5,0.5],
                             ['BlockBottom',0.5,0.5]]
    DA.PVGUI = GUI
    main_parameters.DA = DA

    VirtualLab.Parameters(main_parameters)

    VirtualLab.DA()

# ====================================================================
# Show the sensitivity of the results to the placement of the thermocouples
if Sensitivity:
    main_parameters = Namespace()

    DA = Namespace()
    DA.Name = 'Analysis/{}/Thermocouple/{}/Sensitivity'.format(CoilType,ModelType)
    DA.File = ('Thermocouple','Sensitivity_{}'.format(ModelType))
    DA.MLModel = 'Temperature/{}/{}'.format(CoilType,ModelType)
    DA.MeshName = 'HIVE_component' # name of the mesh used to generate the analysis (see
        DataCollect.py)
    DA.TestData = [DataFile, 'Features', 'Temperature',{'group':'Test'}]
    DA.CandidateSurfaces = ['TileSideA','TileSideB','TileFront','TileBack','BlockFront','
        BlockBack','BlockBottom']
    DA.NbThermocouples = 4
    DA.NbConfig = 5 # number of random combinations of thermocouple placements to test
    DA.PVGUI = GUI
    main_parameters.DA = DA

    VirtualLab.Parameters(main_parameters)

    VirtualLab.DA()

# ======================================================================
# Optimise the location of the thermocouples
if Optimise:
    NbThermocouple = 4

    main_parameters = Namespace()

    DA = Namespace()
    DA.Name = 'Analysis/{}/Thermocouple/{}/Optimise_{}'.format(CoilType,ModelType,
        NbThermocouple)
    DA.File = ('Thermocouple','Optimise_{}'.format(ModelType))
    DA.MLModel = 'Temperature/{}/{}'.format(CoilType,ModelType)
    DA.MeshName = 'HIVE_component' # name of the mesh used to generate the analysis (see
        DataCollect.py)
    DA.TestData = [DataFile, 'Features', 'Temperature',{'group':'Test'}]
    DA.CandidateSurfaces = ['TileSideA','TileSideB','TileFront','TileBack','BlockFront','
        BlockBack','BlockBottom']
    DA.NbThermocouples = NbThermocouple
    DA.GeneticAlgorithm = {'NbGen':5,'NbPop':20,'NbExample':5,'seed':100}
    DA.PVGUI = GUI
    main_parameters.DA = DA

    VirtualLab.Parameters(main_parameters)

    VirtualLab.DA()
```

Listing A.4: Run file used to estimate full temperature and stress field using thermocouple data and optimise their location