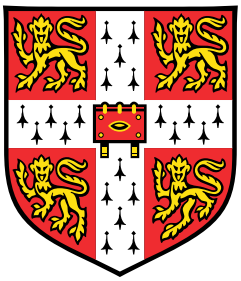
Surrogate Model Optimisation for PWR Fuel Management

Application of advanced surrogate models to optimisation problems in nuclear fuel loading pattern generation



Andrew James Whyte

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Jesus College December 2020

Revision hash: ab83498

To my parents, Carol and Martin Whyte.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, footnotes, tables and equations and has fewer than 150 figures.

Andrew James Whyte
December 2020

Acknowledgements

Many people have contributed to this PhD, either directly or indirectly. The author hopes that the most important ones are acknowledged here, and apologises to anyone who went so far as to read this and has been left out. You must have helped me out, or you wouldn't be looking, please remind me and I'll buy you a pint.

University staff who are owed recognition, include: Peter Benie who has lent his expert level IT skills, and demonstrated the better part of valour while commenting on my efforts. I would like to thank Prof Joan Lasenby, who arranged my EPSRC funding at the last minute. Thanks to Zhiyao Xing and Dr Eugene Schwageraus, for the contribution of simulation results used in Appendix C for their insights and advice with respects to this section of the thesis and for allowing me to collaborate with them on their research. Prof Berloff and Kirill Kailin, who took the idea of fuel loading pattern optimisation and developed the concept of polariton based simulation that is mentioned in Section 6.8.

Thanks to Dr Geoff Parks for his unwavering support and timely advice, for taking the risk of accepting a headstrong, cantankerous, mature student, helping to guide him through the admissions process and for sticking with him on this journey. Also, the the fastidious corrections of Prof Jennifer Whyte and Geoff – without whom all would be lots.

I am grateful to the staff at ANSWERS, Jacobs. Particularly to Chris Bazel and Dr Ben Lindley who made running WIMS on the Cambridge HPC possible. I enjoyed the support of Alan Charles on using WIMS and for inspirational conversations on optimisation.

Thanks to Dr Janine Ochoa for marrying me, supporting and proofreading my work, and sharing our time in college. Also, thanks to Chris Mangnall, for helping me through GCSE and A-Level maths and, more recently, being my companion on the Stanford University, École Normale Supérieure, and University of Toronto MOOCs on quantum mechanics, statistical mechanics, and quantum machine learning that lead to, and shaped this PhD.

Finally, I should thank all of the friends who I have invited to eat with me in college over the past three years. If you were then forced into a conversation about the problems of global energy, nuclear power, optimisation, neural networks or quantum simulators, you are the ultimate proof that there is no such thing as a free lunch!

Abstract

Pressurised Water Reactor (PWR) fuel management is an operational problem for nuclear operators, requiring solutions on a regular basis throughout the life of the plant. A variety of conflicting factors and changing goals mean that fuel loading pattern design problems are multiobjective and, by design, have many input variables. This causes a combinatorial explosion, known as the 'curse of dimensionality', which makes these complex problems difficult to investigate.

In this thesis, the method of surrogate model optimisation is adapted to PWR loading pattern generation. Surrogate models are developed based around three approaches: deep learning methods (convolutional neural networks and multi-layer perceptrons), the fission matrix and simulated quantum annealing. The models are used to predict core parameters of reactors in simplified optimisation scenarios for a microcore, a small modular reactor, and a 'standard' PWR. The experiments with deep learning models show that competitive results can be obtained for training sets using a much lower number of simulations than direct optimisation. Fission matrix experiments demonstrate the method to predict core parameters for the first time, with interesting preliminary results. Novel experiments using simulated quantum annealing demonstrate the technique is able to generate loading patterns by following heuristic rules and is suitable for application to custom optimisation hardware.

The principal contribution of this work is to show that surrogate model optimisation can be used to augment fuel loading pattern optimisation, generating competitive results and providing enormous computational cost reduction and thus permitting more investigation within a given computational budget. These methods can also make use of new computational hardware such as neural chips and quantum annealers. The promising methods developed in this thesis thus provide candidate implementations that can bring the benefits of these innovations to the sphere of nuclear engineering.

Table of contents

List of figures

List of tables

Nuclear Terms:

k_{eff}	Effective multiplication factor is a defining attribute of a nuclear system. It represents the average number of neutrons created for every neutron absorbed or lost by leakage.
BP	A Burnable Poison is a material that absorbs neutrons and is converted to other isotopes. In this way it effectively reduces the local neutron flux available for fission reactions, whilst reducing in concentration over time. Eventually an ideal BP is used up and its effect is suppressed.
burnup	Burnup is term used to measure the amount of energy that has been extracted from fuel. It is measured in MWd/kgHM, *End Of Cycle (EOC)* burnup is critical factor for a nuclear reactor; by extracting more energy from the same amount of fuel, then fuel costs and down time are reduced.
checkerboarding	Alternating the arrangement of nuclear fuel assemblies in a 2D layout task, this achieves lower power in high reactivity fuel and increases burnup in low reactivity assemblies
cycle length	Cycle length refers to the amount of time from when a reactor is refueled to the *EOC*.
DNBR	Departure from Nucleate Boiling Ratio is a significant safety factor for PWRs. Defined as the predicted critical heat flux (the point where nucleate boiling stops and a layer of steam at the surface of the fuel cladding inhibits heat transfer to the coolant) divided by the local heat flux.

enrichment	Enrichment when used for uranium fuels refers to the weight percentage of the isotope U235 in the material.
EOC	The End Of Cycle refers to the reactor at the time when *effective multiplication factor*= 1, at this time reactors such as PWRs must stop operation to be refuelled.
GDA	The Generic Design Assessment (termed DCD under the US regulatory body) is a detailed design document which must be supplied to a regulatory body, such as the UK's Office of Nuclear Regulation (ONR) for approval before a license to build a nuclear reactor is issued.
LP	Loading Pattern refers to the arrangement of fuel in the core of a reactor, this is usually the arrangement of fuel assemblies, in terms of types of fresh assembly designs and the effective rotation and movement of assemblies that have been in the core for a number of cycles.
LWR	A Light Water Reactor is a catch all term for any type of reactor that uses water as a coolant.
PPF	Power Peaking Factor is a core performance metric that is calculated from highest power density region divided by the mean power density in the reactor core. Regions can be at the pin or assembly level. For assembly design the local power peaking factor is also considered. Lower PPFs generally increase the total power produced by the generators.
PWR	A Pressurised Water Reactor is a type of *Light Water Reactor (LWR)* that uses water at high pressure to simultaneously cool and moderate the core. PWRs form the majority of the reactor fleet in operation today.

SMR	Small Modular Reactor refers to any of a range of small modern reactors that are designed to be modular and therefore more easily mass produced.
SNUPPS	A 4-loop PWR design by Westinghouse, implemented at Wolf Creek and Callaway in the US, and in a modified form at Sizewell B in the UK.

Optimisation Terms:

ε-tube	The ε-tube refers to the space that is taken up around the $\hat{f}$ of a Support Vector Regression. The points that lie outside of the ε-tube are called the *support vectors*.
activation function	The activation function refers to any of a number of functions (e.g. $arctan(x)$, relu, leaky relu, sigmoid) that is used to keep the inputs bounded in neural network nodes.
ANN	An Artificial Neural Network is an algorithm for information processing named after its ability to ape one process for learning observed in biological neural cell clusters.
brute force	Brute force refers to a method of computational search where the solution is found by systematically evaluating input values until all values have been evaluated or an acceptable solution is found. It can prove a competitive approach for some classes of difficult problem.
CFD	Computational Fluid Dynamics refers to the use of computers to predict the complex motion of fluids by solving or approximating the Navier-Stokes equations.
CNN	Convolutional Neural Networks are a leading type of *Artificial Neural Network (ANN)* for spatial data. Convolution layers are followed by a feed-forward network. A major benefit of convolutional neural networks is the unsupervised feature extraction.

CPU	Central Processing Units are the primary component of a computer that carries out arithmetic logic according to loaded (programmable) instructions
curse of dimensionality	If the number of inputs to a system increases linearly, the solution space increases exponentially, making search more difficult. The increased challenges associated with solving problems in high dimensionality spaces are known generally as the *Curse of Dimensionality*.
deep network	Deep networks are usually defined as neural networks with three or more hidden layers. The term is used more generally to refer to developments in ANN topologies that have been developed in the last fifteen years.
DSO	Direct Simulation Optimisation is the term used in this thesis to refer to iterative optimisation on the original simulation or system.
dual objective function	Dual objective function refers to the objective function that is derived by elimination of the original variables from the Lagrange function, so that it is written solely in terms of the *dual variables*.
dual variables	Dual variables or *Lagrange variables* refers to the variables introduced to create the Lagrange function, in Lagrangian optimisation.
error backpropagation	An algorithm for deducing the weight of connection signals between nodes in neural networks with hidden layers.
error-based exploitation	Exploitation of the search space based on indications of the error between the surrogate model and the objective function.
FPGA	Field Programmable Gate Array is a type of programmable logic integrated circuit, that can carry out complex calculations in a single clock cycle at the cost of silion real estate

GIS	Geographic Information System is an electronic system for storing and analysing spatial or geographic data.
GPU	Graphical Processing Units are types of processor architecture developed for 3D graphics and subsequently applied to scientific computing
KKT	The Karush-Kuhn-Tucker conditions are that, at the point of the solution, the product between dual variables and constraints goes to zero in a Lagrangian dual problem.
Kriging	An algorithm for interpolation between sampled points using Gaussian distributions.
Lagrange function	Lagrange function refers to an objective function that is derived by the addition of *dual variables*, (also called Lagrange multipliers) to an optimisation function. An optimum is then known to be at the saddle point, where the dual variables are maximised and the original objectives are minimised.
Latin hypercube sampling	A method of sampling based on the *latin square*, which is a square with one sample per row and column. The extension to continuous variables and many dimensions requires that the each dimension is split into a number of equal regions and one sample is made per *axis-aligned hyperplane*.
Least squares criterion	Criterion for minimum scalar error: $$\min_{w} \Sigma \left[y^{(i)} - \hat{f}(x,w) \right]^2 \quad (1)$$
MAE	An error measure used in computing. In a computer, it is computationally more efficient to calculate the Mean Absolute Error rather the *RMSE*. Although the output is similar the mathematical operation is different.
meta-model	See *surrogate model*.

MIMO	Multi Input Multi Output describes objective functions that have many variate inputs and many variate outputs (both **x** and **y** are vectors).
MLP	Multi Layer Perceptrons are a common type of *ANN* where nodes connect from the input layer through a number of hidden layers to the output layer and each layer connects only to the subsequent layer.
MO	Multiobjective Optimisation describes optimization problems that have more than one output objective.
MOC	Method of Characteristics is a technique for solving partial differential equations by converting them in to ordinary differential equations. It is primarily referenced in this text as a method for solving the neutron transport equation.
NDF	The Non Dominated Front also known as the Pareto frontier of a MO problem is the $N-1$ dimensional space that describes the space where one or more objectives are optimal. The NDF is useful for decision makers as any solution that lies on this frontier cannot be improved for one objective without impairing another objective.
NFL	The No Free Lunch theorem states that, for the set of all problems, no optimisation strategy outperforms any other. This means that information gleaned by successive trials of the optimisation algorithm also implies assumptions about the objective function, f
overfitting	A model is said to be overfitted if it describes the sample set well, but does not perform well at predicting the function $f(x)$. Unless data is noise-free, care must be taken to avoid overfitting.

polynomial regression	Polynomial regression is a surrogate model based on generated relationships between input x and output variables y using sums of polynomial functions of different orders.
prediction-based exploitation	Exploitation of the search space based on the predictions given by the surrogate model.
QA	Quantum annealers are machines that optimse a function by expressing the function in the form of an 'Ising model', the global optimum of which is found by transitioning to it via a quantum adiabatic process. (see Section 7.2.1)
RBF	A Radial Basis Function refers to a function that is symmetrical radially from a centre point, used in estimation of $\hat{f}(\mathbf{x})$. An example is the $sinc(x)$ function, which forms the basis function for wavelet transforms.
RSM	Response Surface Model a method similar to polynomial regression.
SA	Simulated Annealing is a method of iterative optimisation which exploits an analogy between the way a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system.[176]
SMO	Surrogate Model Optimisation is optimisation of a system by running the optimisation on a model of the system that is not generated by adherence to the original physics of the system.
surrogate model	A model of the system that is being studied, allowing the investigation of a search space when the system being studied is difficult, costly or slow to sample.

SVR — Support Vector Regression is a method of regression where points within an error margin $\pm e$ are ignored, the results being dominated by the outlying or *support vectors*. SVR is generally regarded as a special case of Support Vector Machines.

XOR gate — Exclusive OR is a basic logic function, which processes true when exactly one input is true but *not* both, so the truth table is:

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Mathematical Nomenclature:

RMSE	The Root Mean Squared Error.
$\Pi(x)$	The rectangular function, a function defined as: $$\Pi(x) = \begin{cases} 0.0 & \text{if } \lvert x\rvert > 0.5 \\ 0.5 & \text{if } \lvert x\rvert = 0.5 \\ 1.0 & \text{if } \lvert x\rvert < 0.5 \end{cases} \quad (2)$$
$\hat{f}(\mathbf{x})$	The Surrogate Model Function, a function that is built based on observations of $f(\mathbf{x})$.
$\mathbf{G}$	The Gram matrix, defined as: $$\mathbf{G}_{i,j} = g\left(\lVert x^{(i)} - c^{(j)}\rVert\right), \qquad j = 1,\ldots,n \quad (3)$$ where x and c are vectors on a Euclidean space, and g is an arbitrary function.
$\mathbf{V}$	The Vandermonde matrix, a matrix whose terms are a geometric progression in each row.
$\mathbf{w}$	The weight vector or matrix, a set of real numbers that scale different inputs in many regression methods. For example: $\mathbf{w} = [w_1, w_2, \ldots, w_n]$.
$\mathbf{x}$	The input variable vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$.
$\mathbf{y}$	The output vector $\mathbf{y} = [y_1, y_2, \ldots, y_n]$.

$\mathscr{U}(a,b)$	The Uniform Random distribution, a random function whose probability is evenly distributed between bounds, a & b. $$PDF = \begin{cases} \frac{1}{b-a}, & \text{for } x \in [a,b] \\ 0, & \text{otherwise} \end{cases} \quad (4)$$
d	Number of dimensions or input variables x_i of $f(\mathbf{x})$.
$f(\mathbf{x})$	The objective function that describes the aspect, cost or other performance metric of the problem to be optimised.
m	The order of a polynomial model.
r^2	The correlation coefficient of functions or samples.
$sinc(x)$	Usually defined as: $$sinc(x) = \begin{cases} 1, & \text{for } x = 0 \\ sin(2\pi x)/2\pi x, & \text{otherwise} \end{cases} \quad (5)$$ A symmetrical function which is common in engineering and signal processing. For example, it is used as the basis function of wavelet transforms.
central limit theorem	A theorem that states that, for sets of independent random variables, their properly normalized sum tends toward a normal distribution.

Chapter 1

Introduction: Optimisation and PWR Fuel Management

1.1 Pressurised Water Reactors, Design and Management of Fuel Assemblies

This thesis investigates the applicability of *Surrogate Model Optimisation (SMO)* to *Pressurised Water Reactor (PWR)* loading patterns. In this chapter, we will introduce PWR fuel optimisation, firstly through a description of the process of PWR fuel reload design with a focus on the kind of issues facing designers. Then some common *PWR* core designs are presented, followed by an introduction to iterative optimisation and the landscape of multiobjective optimisation today.

1.1.1 Operation and Fuel Reloading

A *PWR* is a batch fuelled Nuclear Power Plant (NPP) that normally operates for between twelve and twenty-four months between refuelling [16]. In theory, a reactor can run until the multiplication factor $k_{eff} \leqslant 1.0$ when the reactor is no longer critical. However, due to contractual obligations, it is normal for the refuelling of a reactor to be carried out at a planned outage. Planned outages are usually arranged to be at seasonally low baseload times. These are summer in the UK or spring or autumn in the US, where air conditioning units account for a significant power draw (28.6% of residential power use [22]).

The process for fuel reload design in an example utility company's organisational structure is described by Glasstone and Sesonske [75, p. 604]. Figure 1.1 shows Glasstone and Sesonske's organisational groups and a graphical representation of the process. There are three main groups: the *plant scheduling group*, who are responsible for operation management; the *fuel*

procurement group, who are responsible for economic management analysis and procurement; and the *in-core reactor analysis group*, who are responsible for the reactor physics analysis. The first step in the process is for the plant scheduling group to establish the planned outage window for refuelling and to generate requirements for the next batch in terms of predicted energy requirements. The fuel procurement group will generate requirements in terms of plant economics, energy needs and the available fuel for refuelling. The in-core reactor physics group then use this information to generate the best loading pattern possible over a process involving two fidelities of simulation. Once every group is satisfied with the proposed solution, a detailed simulation is carried out for regulatory purposes, and the regulator will expect evidence that the proposed loading pattern is be within safe operating parameters.

Within safe operating parameters, some of the trade-offs that are possible are:

- Maximising discharge burnups and uniformity of discharge burnup
- Reducing feed batch size;
- Extending cycle time, increasing plant availability;
- 'Low leakage' fuel loading, which extends the life of the pressure vessel by minimising the neutron leakage from the core;
- Reducing feed enrichment whilst retaining a high enough reactivity to achieve the requisite cycle length;
- Minimising the *Power Peaking Factor (PPF)* and other metrics to achieve a more even coolant temperature profile; or
- Minimising *Burnable Poison (BP)*, due to the 'residual poisoning penalty' of high neutron capture isotopes that build up during exposure. [175, 75]

Due to the changing requirements for operators caused by changing economic and regulatory conditions and the number of groups involved in the decision, it is often better to understand the trade-offs between different *Loading Patterns (LPs)* than it is to try to establish a single 'best' core *LP*[175]. To do so electricity utility companies are interested in developing fast, efficient procedures that explore a wide range of fuel scenarios. [75, p. 607].

Section 4.1 will introduce a simplified model of Figure 1.1 used in this thesis, as it has not been possible to recreate the working groups of a utility company. Due to its commercially sensitive nature actual scenario data has not been available in this thesis. However, possible requirements have been selected from the above list, and the activity of the in-core reactor physics group has been approximated.

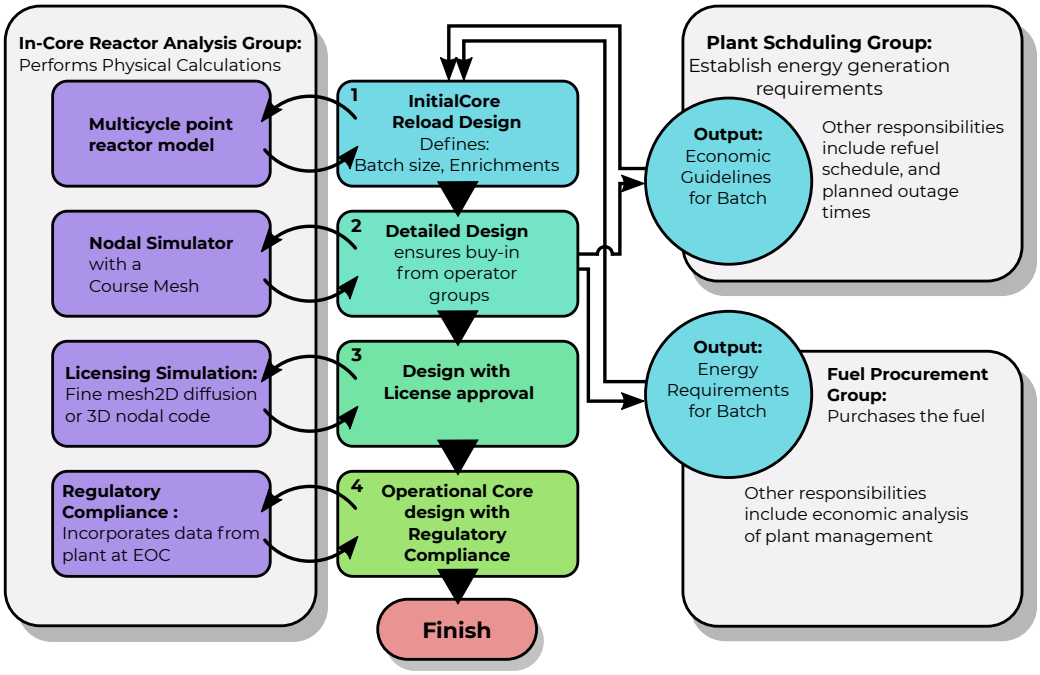


Fig. 1.1 The fuel reload design process showing active organisational groups in a utility company, based on the description in Glasstone[75]. Stage four must occour after operational data has been collected from the previous batch.

1.1.2 In-core Reactor Physics

The in-core reactor analysis group is mainly interested in the modelling of physical processes that occur within the core of the nuclear reactor. These include neutron transport, *Computational Fluid Dynamics (CFD)*, heat transfer and material degradation effects.

The neutron transport equation is a differential equation, based on a the balance of neutrons being produced by fission, and neutrons lost from the geometry or absorbed by materials. The behaviour of neutrons can be understood from a 3D coordinate r, the energy E, a solid angle $\hat{\Omega}$, velocity $v(E)$ and time t. When considering civil power reactors, it is often convenient and reasonable to assume scattering is isotropic on average and that the neutron interactions can be simplified into a few energy groups.

Unfortunately, complexity arises from the vast array of isotopes being created as fission products (whose atomic mass forms a spectrum between 70 and 160) and daughter nuclides created by the decay chain of unstable fission products [18, p 30]. The interactions of the neutrons with these materials is an extremely complicated process, but is governed by a well established equation:

$$\mathbf{M} \cdot \Psi(\vec{r}, E, \hat{\Omega}) = \frac{1}{K} \frac{\chi(E)}{4\pi} S(\vec{r}) \tag{1.1}$$

The neutron transport equation, and the fact that each isotope of every element in the core has its own set of cross-section Σ values forms the basis of much of the complexity of neutronics, which is mentioned in more detail in Section 4.2.3. However, for a more detailed reference, there is also a significant body of work on neutronics, including seminal textbooks by Bennet and Thompson [18] (a qualitative overview), Lewis [129] and Oka and Kiguchi [249] (with insights on the complexity of simulation).

The coupling of neutronics with heat transfer and *CFD* is a matter of some interest in the field at this time [228], but due to the complexity of the stable coupling of complex systems of differential equations and the added problems associated with automating this for optimisation, it is not within the scope of this thesis.

1.1.3 PWR Data Sources

Optimisation of the fuel can be at the assembly level (known as fuel design), in-core fuel management, and out-of-core fuel management. In this section, a number of commercial and research reactor core designs, for which reliable core data is available to the academic researcher. In-core fuel management is the arrangement of the assemblies of fresh and usually older batches of fuel in the core. Logging of the fuel history, the conversion ratio, and the residual reactivity would allow the extension of techniques in this thesis to mixtures of batches of fuel, called reload cores. Out-of-core fuel management includes all of the design, manufacture, handling, purchasing and storage tasks associated with fuel before it is put into the core. This is controlled by the 'fuel procurement group' mentioned in Section 1.1.1. This thesis will mainly consider in-core fuel management, of the kind that would be carried out by the in-core reactor analysis group. The ultimate goal of *PWR* fuel management would be to unite the reload core with the specific choices of fresh fuel design. This has been described as the 'grand challenge' of fuel management [220] and progress has begun made to break this division, for example by Maldonado, [140].

Expert knowledge of the operation of a nuclear reactor is usually kept confidential by the utility companies. However, a number of reliable sources of information about PWR cores are available. Firstly, in order to be licensed, detailed design data is supplied to the regulatory body in the form of a *Generic Design Assessment (GDA)*. In some countries, such as the UK and US, this information is publicly available. The *GDA* is used in Sections 1.1.4 and 1.1.5 for the description of EPR[1] and Nuscale designs, respectively.

[1] EPR used to stand for "European Pressurised Reactor, which was changed to "Evolutionary Power Reactor" to encourage international adoption, but is no longer being used as an acronym and is now simply being called EPR.

Although *GDAs* give significant insight into the geometry of the designs, the operation of the plant is considered commercially sensitive information, and so it is not usually released by the utility companies. A recent trend in the nuclear industry is for organisations like the World Association of Nuclear Operators (WANO) to be actively working to generate 'self-regulation' of the nuclear industry through a process of peer-review. This means the sharing of expertise between utility companies and is an economically sensible step, since a nuclear accident anywhere will have negative effects on the nuclear industry worldwide. For example, following the Fukushima incident in Japan, every NPP in Germany was shut down – a decision that politicians directly attributed to the accident. In general, this has not extended to the sharing of data or expertise between utilities and academia. However, this important trend parallels similar trends in a diverse array of scientific fields. For example, drug discovery, astronomy, particle physics and computer science have open data policies and many academic journals have adopted guidelines on the open publishing of data [157].

A significant exception in the nuclear industry is the Benchmark for Evaluation and Validation of Reactor Simulations (BEAVRS) benchmark [86]. In this case, a considerable amount of operator knowledge has been released to the academic community. The BEAVRS benchmark includes the actual start-up core design and operational data at the level of sensor outputs from a reactor in the United States and is examined in Section 1.1.4, which is on the most common design of *PWR*, a configuration of 193 assemblies.

1.1.4 'Full Sized' PWRs

PWRs contribute around 67% of all active NPPs [91], which has led to a fairly standard core arrangement of 193 fuel assemblies, as shown in Figure 1.2 for example in *SNUPPS* reactors from Westinghouse and the EPR design by Framatome.

Most modern *PWR* fuel is based on a 17×17 array of pins called an assembly (a notable exception is the Russian designed VVER[2]). This near universal design has been made by Westinghouse and Babcock and Wilcox since at least 1979 [229], and the geometric designs remain almost completely unchanged in modern reactors [13], while important innovations have occurred in the materials used for cladding, use of *BP* pins and in the recognition of the benefits of fuel design and loading pattern optimisation.

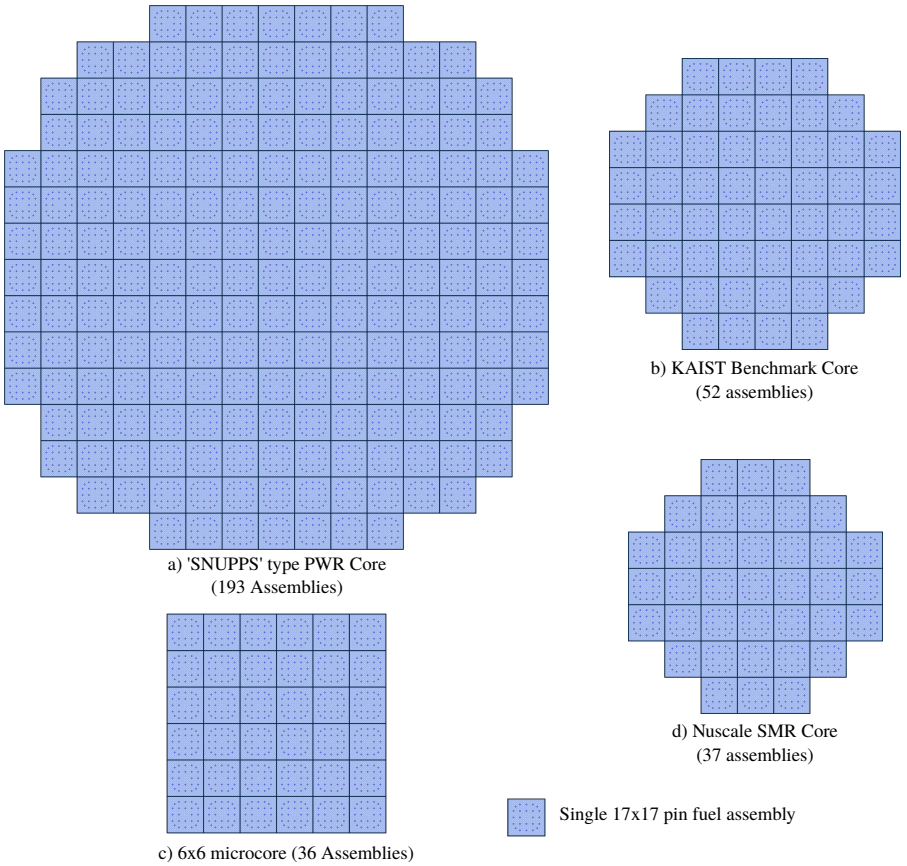


Fig. 1.2 The geometry of different PWR cores showing the locations of fuel assemblies, a) a Westinghouse *Standard Nuclear Unit Power Plant System (SNUPPS)* type PWR, b) the KAIST benchmark small research core [36], c) a 6×6 microcore used for experiments in this thesis and d) an SMR core by Nuscale [168].

EPR

The UK EPR - *GDA* [13] has been redacted since the beginning of this project, so a combination of the author's notes and the U.S. Nuclear Regulatory Commission's design control documents [12] have been used instead as evidence.

Figure 1.3 on p. 7 shows the assembly designs that have been authorised in the EPR. Two main variables are considered: the use of burnable poisons and the fuel enrichment. An example of core loading is described in the *GDA* documents, demonstrating examples of start-up and ongoing core loading patterns. Apart from subsituting a control rod for an instrumentation tube, all of the designs have four axes of mirror symmetry. The allowed fuel designs are a major limitation for the core loading pattern designers, who are limited to only authorised assembly

[2] VVER is the acronym from the Russian: *Vodo-Vodyanoi Energetichesky Reaktor*

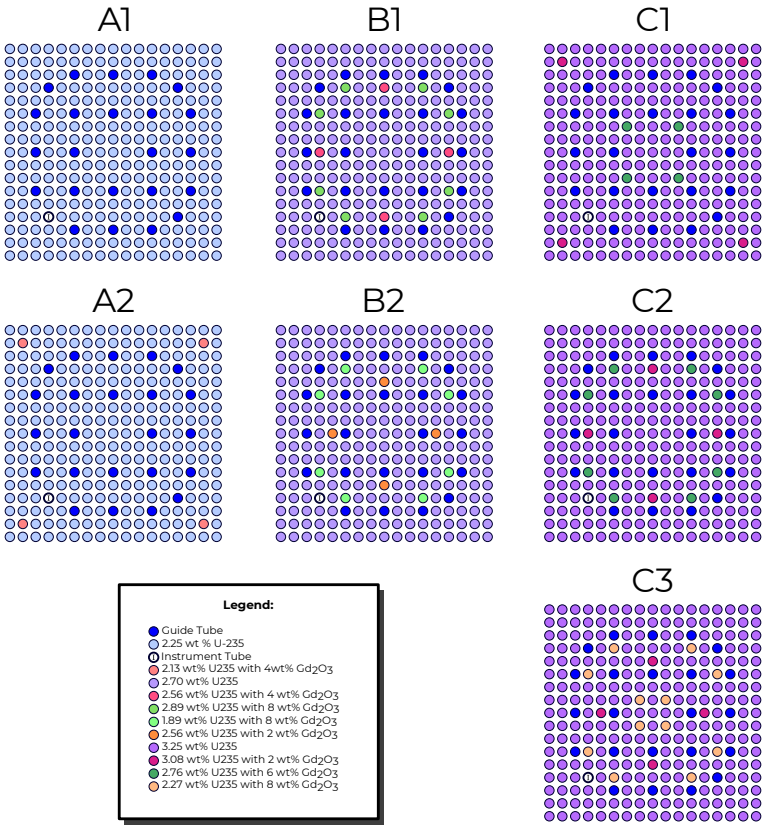


Fig. 1.3 The PWR fuel assembly designs licensed by the EPR GDA, shown in a consistent colour scheme, based on [12] and similar to the now redacted [13].

designs and will be further limited by the stock that has been purchased by a separate group within the organisational structure.

BEAVRS

The BEAVRS benchmark defines itself as follows:

> "This document introduces a new benchmark that addresses many of the shortcomings of previous LWR benchmarks by providing a highly-detailed [sic] PWR specification with two cycles of measured operational data that can be used to validate high-fidelity core analysis methods." [86]

The data includes an initial core loading pattern and two reload configurations, transitioning to an ongoing batch plan. The operational data, including sensors and burnups are also included.

The BEAVRS benchmark initial core loading pattern is used as a comparison in the results in Chapter 7, where full start-up core loading pattern designs are generated using a quantum annealer.

1.1.5 'Small' PWRs

There is a current trend for *Small Modular Reactors (SMRs)*. These reactors are smaller than typical *PWRs*. The advantage of this is that they can be process manufactured, leading to standardisation and reduced cost per unit. However, many of the current designs can be fitted with conventional PWR fuel assemblies. These smaller geometries offer novel and potentially interesting opportunities for optimisation.

Chapters 5 and 6 of this thesis investigate a theoretical reactor, as shown in Figure 1.2c. This reactor would not be optimal for real operation. However, it has a comparable size to SMRs such as Nuscale [168] and Multi-Application Small Light Water Reactor (MASLWR) [153]. The microcore was created as a minimal 'toy problem' that has not previously been investigated.

KAIST

The benchmark from KAIST is a small reactor simulation based on 13 standard PWR assemblies with reflected boundary conditions [36]. The KAIST benchmark does not represent a real power reactor design, but is freely available and popular for experimentation. However, due to the the sensitive nature of real commercial reactor data, it is often used in academic and simulation literature. The core is formed by a 4×4 array of fuel with three missing assemblies from the corner and reflected boundary conditions (as shown in Figure 1.2b).

Nuscale

Nuscale is a relatively new US company, which entered the SMR market in 2011 with the stated aim of brining their novel design to the market by the end of the decade [167]. Although current estimates now put this at 2026 [169], the company has demonstrated the ability to generate regulatory compliance documentation.

It is expected that a number of cores will be used at any site, with each SMR producing just 45 MW. The core is shown in Figure 1.2d, and can be analysed with rotationally symmetry with 10 assemblies per quadrant [168].

1.2 Optimisation in Nuclear Power Plants

Despite significant advances in research on optimisation in nuclear power plants ([220, 219, 84, 162]), and the development of a number of tools for the purpose of in-core fuel management with automatic optimisation capabilities (*e.g.* FORMOSA-P [104], CM-PRESTO [193] and PANTHER [156]), it was claimed by Turinsky in 2005 that expert operators following heuristic

rules will outperform automated in-core optimisation tools [219]. Despite the age of this claim, it would seem that utility companies have not significantly changed their approach, and in-core fuel management optimisation is still carried out by expert operators using heuristic rules [220].

The alternative to heuristic rules for global optimisation, and the one favoured by this thesis is iterative optimisation; techniques such as *Simulated Annealing (SA)* or evolutionary algorithms. These approaches are useful since they work without complex manipulation of the objective function. Due to the very high complexity of neutron transport simulations or *CFD*, system behaviour can change significantly with small changes to neglected variables. So is useful to be able to optimise without manipulating the objective mathematically, since this builds robustness to this kind of potential failure.

SA and evolutionary approaches along with other Metropolis methods form the foundation for of a vast array of optimisation techniques based on stochastic iterative optimisation. Rather than finding the differential operator of the objective function, it is sufficient to solve the problem by running a simulator many times. This broad class of optimisation techniques will be termed *'iterative optimisation'*.

Iterative optimisation algorithms are sometimes referred to as *'black box'* optimisation, because they are able to optimise when the objective function is neither visible nor mathematically defined. While this is an advantage in practical terms, opinion is divided within the scientific community, since these techniques can be applied without careful analysis.

Iterative optimisation often uses some form of metaphor as an inspiration for the algorithm. In our day to day lives, people routinely use metaphor as a powerful tool in gaining insight and understanding, but it is not apparent that the real-world uses metaphor in its function[3]. In other words, the search landscape of a nuclear fuel loading pattern does not necessarily lend itself to the behaviour of crystals cooling, wolf pack dynamics or jazz musicians.

The enthusiastic usage of metaphors in the search for novel algorithms, led to further criticism of research into iterative optimisation techniques. An insightful critique of metaphorical approaches in optimisation is given by Wayland [231] who singles out (by his own admission, unfairly) the so-called 'harmony search' algorithm and proves that it is clearly a subset of evolutionary algorithms.

However, it has been shown many times that iterative optimisation strategies are performant under a wide range of problems, particularly the more common approaches, such as *SA* or evolutionary algorithms [111, 152, 175]. A central theme of this thesis is how and when engineers, in general, and nuclear operators, in particular, might consider a metaphorical method, what caveats must be understood and what advantages the approach might bring.

[3] The author appeals to the reader's common sense here, since it is the domain of philosophers such as Dennett [54, pp. 38–66] to discuss this rigorously.

Iterative optimisation problems in nuclear engineering and *PWR* fuel management often require repeated solutions for neutron transport problems or thermal hydraulics, which are notoriously computationally complex, requiring a long time to execute. Execution time of computer programs is usually measured in time per *Central Processing Unit (CPU)*.

Since iterative optimisation typically requires solving the problem multiple times, the following accounting can be done in order to evaluate the motivation for this work:

$$t_{tot} = N\left(t_a + It_p\right) \tag{1.2}$$

where:

$$\begin{aligned} t_{tot} &= \text{total computational time, } CPU \text{ hours} \\ N &= \text{number of generations} \\ t_a &= \text{time for execution of the optimisation algorithm} \\ t_p &= \text{time for evaluation of the objective problem algorithm} \\ I &= \text{Number of evalutions of the objective per generation} \end{aligned}$$

For nuclear engineering optimisations; where the optimisation algorithm may have a t_a in the order of hundreds of seconds, the burnup calculations have t_p measured in hours, then:

$$t_p \gg t_a$$

Therefore, the additional cost of increasing the complexity of the optimisation algorithm is justified if it results in improvements in I, up to the point where kt_a becomes comparable to It_p, where k is the rate of increase in computation time of t_a per I.

1.2.1 Multiobjective Optimisation

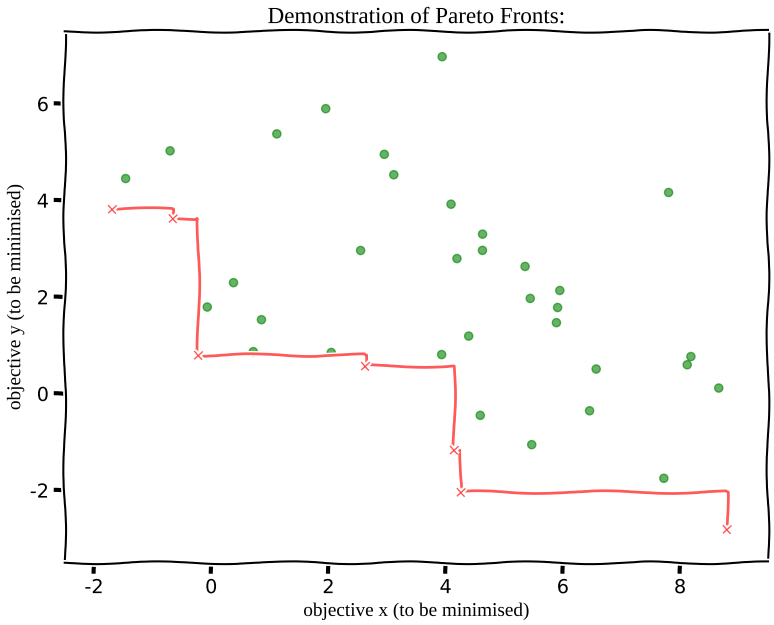


Fig. 1.4 Illustration showing a *Non-Dominated Front (NDF)*, the points in red, of a set of points in 2D, based on the `pareto.py` script (*seed* = 61).

It is sometimes possible to amalgamate objectives, by creating a 'cost' function, in which the cost of many objectives can be traded off as if the system was a single output metric (in fact, it becomes a single objective optimisation). If the variables are not orthogonal, then a cost function between the variables can be created and the *Multiobjective Optimisation, (MO)* collapses into a single variable optimisation of the cost function. However, creating a cost function for the trade-off between orthogonal variables is not always possible[147]. For example, running a reactor at a higher flux density (and therefore evolved power) will increase neutron leakage and decrease the operational life of the reactor vessel. The correct choice depends on factors that may not be substitutable or may be unknown to the utility. Variables such as flux leakage and power generation may not be easily traded: since the utility company will want to extend the NNPs life and to profit by generating the maximum power possible. However, increasing the power ouput will decrease the plant longevity, while plant life may also be affected by political or physical constraints that are naturally unpredictable. Furthermore, predicting energy requirements over periods longer than the 'contract to supply' is also difficult.

Due to the conflicting parameters being optimised and the involvement of many parties, in in-core fuel management, there is a focus on understanding trade-offs such as those described

in Section 1.1.1. It is preferential, in an N objective problem, to identify an $N-1$ space that is known as the *Pareto* or Non Dominated Front. *NDF* generation from solution points was proposed by Gass and Saaty [73]. The *NDF* are solutions that cannot be improved for any objective without reducing the performance of another objective, that is to say that they are *not dominated*. Identification of the *NDF* allows the Decision Maker (DM) to choose suitable solutions that meet subjective (or un-modelled) requirements of the system, or to propose a range of solutions to other groups within the utility company.

Whilst it might be possible to compare the monetary cost of a reactor verses the power evolved, this would ignore other factors that the DM might be aware of. These might include, for example, future energy demand, the importance of ensuring a reliable energy supply to the country's economy, the relative difficulty with which new reactors can be built, and the chance of downtime from higher power operation, as well as pressure from the contract to supply. The idea of an *NDF* is shown in Figure 1.4, where the green dots are solutions where another solution gives less y for the same or less x, known as the 'dominated set'. The non-dominated set are depicted by red crosses. These solutions represent the 'best' set of solutions when x and y are orthogonal objectives. The *NDF* forms a $DN-1$ space for any problem of dimensionality DN. This approach is common in economics, such as in the discussion of economic strategies that distribute wealth between individuals [147]. If the trade-off between x and y is not known, then the *NDF* gives the DM the knowledge of whether a given solution is among the set of best possible solutions, for conflicting objectives.

1.2.2 The No Free Lunch Theorem

A basic assumption of any system that is to be optimised by iterative, 'black-box' methods is that successive trials will impart some information about the underlying system. It is easy to intuit that all optimisation strategies will perform equally poorly when operating on a random function. Since each input and output value are independent, no information is gained from one result to the next, so no meaningful inference about the problem can be made. [4]

By extending this result to the set of all functions, it is fairly easy to qualitatively interpret the theorem of Wolpert and MacReady [241], the so-called *No Free Lunch (NFL)* theorem for optimisation.

> "For both static and time-dependent optimization problems, the average performance of any pair of algorithms across all possible problems is identical." [241]

[4] Although truly random functions cannot be created algorithmically, pseudo random functions, such as the Mersenne Twister algorithm [149] are considered good approximations.

Thus, no search algorithm can be more efficient than any other at locating the global optima (or any optima) based on previous results, *across this set of all functions*. Furthermore, for every example where a particular optimiser gives an improved performance, there is a counter example where that optimiser is less performant on that objective function than another optimisation strategy.

This result appears to run against the obvious effectiveness of search and optimisation algorithms, but, in fact, it simply points towards the underlying structure, and therefore, the predictability of physical systems and a way forward for iterative optimisation. The set of all functions is overwhelmingly 'Kolmogorov Random' (conceptual random functions that are entirely made of sets of independent random variables) and are, therefore, only computable with an infinitely sized (or function sized) lookup table. In fact, the known universe can be shown to have structure, meaning that information about the search space can be intuited from the observed data for real systems.

This is equivalent to saying that in real world systems, the Occam's razor principle applies (*i.e.* that systems of lower complexity are more common than systems of higher complexity). Occam's razor has been argued as a response to the *NFL* by Whitely and Watson [232] and Giraud-Carrier and Provost[74] in 2005. An argument for Occam's razor can be constructed as follows. Consider that real problems exist inside a framework of fundamental particles that obey rules. For example, each particle in a Hamiltonian physical system can be seen to be represented by a six-dimensional tensor, and the interactions of the particles considered to be calculations of these tensors. Real fundamental particles follow more complex dynamics but are still essentially predictable within the bounds of probability functions.

Extending this premise for the entire universe is a daunting task. Luckily, the 'heavy lifting' of calculating the complexity for quantum systems, extending it to every particle, and considering corner cases like the beginning of the universe when radiation was dominant, was done by Lloyd [136], thereby giving us an upper limit to the complexity of real systems:

> "The universe can have performed no more than 10^{120} operations on 10^{90} bits."[136]

Although these results may be anecdotal, they do imply the existence of a numerical upper limit to the amount of information that the universe encodes. While this is necessarily more than can be considered, it is less information than a conceptual Kolmogorov random function would contain. Any example of a real system in an engineering problem will concern itself with a tiny fraction of the universe, and so contains much less information. Furthermore, it will likely be appropriate to consider the system at a higher level than the atomic one, where still less information will be required.

The significance of the *NFL* theorem, stated another way, is that an arbitrary algorithm that outperforms a brute force method, or any other algorithm, is using aspects of the structure of the physics of the objective function, either intentionally or inadvertently.

The surrogate model approach in this thesis is an attempt to create an alternative algorithm, that actively incorporates knowledge of the design space into a model. The *NFL* is relevant to the justification of a surrogate model as the model is a metaphorical model of the real system. Normally it is a simplification of the original system, and therefore it is important to be aware that the physical structure imparted from the objective function to the surrogate model that guides the optimisation is based on the assumptions of that particular surrogate modelling approach.

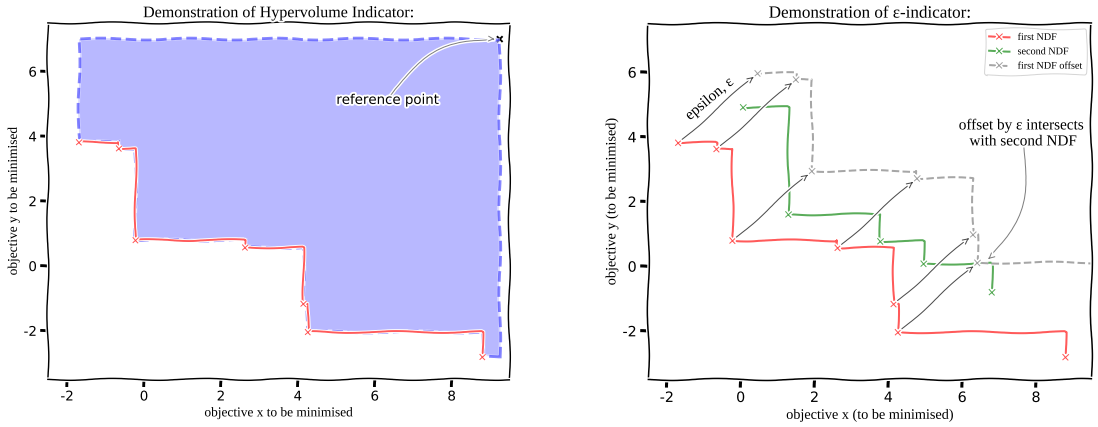
The *NFL* theorem of optimisation, on the one hand shows us that there will always be work to be done to show that some algorithm operates more effectively on a given class of problems than another, while on the other hand, it shows us that this result will only be helpful in the domain of the class of problems being investigated. For a truly unknown objective function, it is acceptable to use a search algorithm that makes few assumptions about the search space. Logically, it also follows that, when even very rudimentary aspects of the objective function are known, intentional use of the problem structure in the optimiser is advantageous. Armed with this knowledge, a framework for optimisation that utilises aspects of the objective function within the optimiser is seen as a significant goal. With the insight that we must use physical information in optimisation problems, it is now possible to start to look at how to develop a system of optimisation that begins to break open the 'black box' of iterative optimisation algorithms.

1.2.3 Surrogate Model Optimisation

The original aim of this PhD was to investigate ways to alleviate the computational burden of objective function evaluation – the main cost of iterative optimisation. Equation 1.2 tells us that the biggest gains can be made in the cost of evaluating the objective function. Three approaches to reducing the cost of iterative optimisation can be attempted. Firstly, to try to optimise with fewer iterations (see, for example, the work by Charles [33]) or, secondly, try to decrease the complexity of the objective function (for example, in nuclear simulations, a coarser mesh or a model with fewer energy groups might be implemented). The third approach, which will be investigated here, is to replace the objective function with another function that has properties that reduces the computational complexity. The resulting approach, Surrogate Model Optimisation (SMO), inevitably requires an understanding of when it is acceptable to use a function as a metaphor for another function.

For metaphorical methods that are developed by testing the original objective function, a balance between the cost of generating the training data and the cost of the original optimisation process must be considered.

1.2.4 Evaluating the Performance of *MO*



(a) The hypervolume indicator

(b) ε-indicator by addition

Fig. 1.5 Hypervolume and ε-indicators, the two main indicators of performance used for *MO* (`hypervolume.py, epsilon.py seed=61`)

When comparing methods of single objective optimisation, the performance of the optimisation algorithm depends on how much computational budget is used and the average value of the optimal solution found. Unlike single objective optimisation, the performance of a multi objective optimisation must be derived from the *NDF*, an N-1 dimension frontier.

The usual methods of measuring the performance of *MO*'s results are the *hypervolume indicator* and the *binary* ε *indicator* [34].

The hypervolume indicator, is the volume of the region between a large-valued fixed reference point and the *NDF*. For example, in Figure 1.5a, a two-objective problem, the hypervolume indicator is the area shaded in blue. Problems can occur with the hypervolume indicator if values from the *NDF* have a larger magnitude than the reference point. If the *NDFs* are known in advance, then the maximum value for each objective can be used or it is common practice to normalise objectives and use a reference point of 1.0.

The ε indicator measures how far an *NDF* from one optimisation would need to move in order to be dominated by another. It can be applied when comparing two *NDFs*. It can be generated by multiplication or by addition of the *NDF* points by a target vector ε until it is dominated by the other at every point (termed *strictly dominated*).. In essence, it is a measure of

the distance at the closest point from one frontier to the other, and can be used to measure how much better one set of solutions is than another. The ε indicator (by addition) is demonstrated in Figure 1.5b, the first *NDF* is moved by the 45^{o} vector ε until it intersects with the second *NDF*, at this point the ε-indicator is the magnitude of ε.

A disadvantage of both the hypervolume and ε indicators is that they treat the normalised objectives as equivalent. While this makes sense on benchmark problems where the objectives are often of equally weighted, this is expressly not the case for real world *MO* problems, where one objective may have a substantially different distribution from the other. It is also not possible to use relative values *a priori*, meaning that this step can only be carried out after the completion of a study. Another approach for comparing *SMO* with *Direct Simulation Optimisation (DSO)*, rather than just measuring performance, it would be useful to understand how closely the distribution of points from *SMO* matches the *DSO* results. In order to do this, a number of statistical tests can be considered.

Non-parametric statistical tests for similarity

A number of statistical tests for significance exist and are routinely used for the comparison of *NDFs* [112]. Statistical tests, such as the ones mentioned below, act on univariate analysis and can be applied to the hypervolume or ε-indicators.

- **Kolmogorov-Smirnov (K-S) Test:** The most popular measure of 'goodness of fit' [14], the K-S test is based on the 'supremum' difference between ranked members of a set and a distribution, or two sets of data. It has also been extended to multivariate forms [178, 61].

- **Mann Whitney Wilcoxon U Test:** A rank-based estimate of likelihood of similarity to the median, similar to the ANOVA test. This method is sensitive to variance of distributions and deviations from normality [59].

- **Kruskal-Wallis Test:** A rank-based statistic that gives an indication of the likelihood of two sets of samples being from the same distribution. It does not require the normality assumption of other tests [119].

These approaches use ranked data, which reduces the effect of outlier data.

Another approach is to perform a Multivariate Analysis of Variance (MANOVA). A MANOVA is an analysis of variance of multiple, multivariate sample sets. It has a number of advantages for certain types of data, such as being able to find differences not shown in multiple ANOVAs on individual variables [212, pp. 322–390].

1.2.5 Understanding Fuel Management Problems in terms of Dimensionality

The input and output spaces of a problem are counted in either the number of states, or the number of dimensions for continuous variables.

Before problems are investigated, it is important to assess how many dimensions are being searched and whether this is feasible. As an example, consider the complexity of fuel assembly design vs a *PWR* core. If both problems have $^1/_8$ symmetry, optimising the enrichment of U235 per uniform assembly in the core and enrichments per pin in the assembly, then there are 31 variables in in the full sized 'SNUPPS' core and 39 variables per assembly. *SMR* cores have significantly fewer input variables, meaning that single batch SMR cores are significantly simpler problems for optimisation than a single assembly, specially, if symmetry is allowed and the inputs are continuous, or have the similar numbers of possible values.

The reason that this thesis concentrates upon core level optimisation is because, surprisingly, this is a more tractable problem than fuel design from a numerical point of view. Furthermore, the 'mean free path' of a neutron makes assembly design a roughly linear problem, with each pin's power correlated to its own enrichment and equally connected to all other pins. This makes the assembly design problem relatively structured, but still numerically difficult. Furthermore, fuel designs are part of the GDA, so they are rarely added after the initial construction of a reactor.

The reasons that most of the core design happens at the assembly level is partly because the Regulatory Design Assessment usually only authorises a small number of assembly designs. This means that the procurement team only make purchases from these options. Since the procurement team must ensure that there is always an inventory of fuel available for the plant, they will often purchase the fuel months or even years in advance. Since the purchasing team are disconnected from the immediate needs of the reactor designers, it is often the case that only a small selection of different fuel types are available.

1.3 Aims and Objectives

This thesis aims to investigate the suitability of *SMO* for the management of fuel in nuclear reactors. It investigates the technique of *SMO* by designing a framework for evaluation of *SMO*, which will be tested with a number of fuel loading scenarios. The framework is designed to maximise the likelihood of success of the surrogate models and to demonstrate the benefits of a framework approach. Using a framework of code facilitates experiments on the application of *SMO* to be carried out in a systematic way for a variety of scenarios.

This thesis describes three novel approaches to nuclear fuel management optimisation, each one using a different 'surrogate' model to estimate the objective to be optimised. Collectively, they are examples of *SMO*. These techniques are based on export of standard methods from fields like deep learning or quantum computing (Chapter 5 and 7), or adapted from a method used in source convergence of Monte Carlo simulation – the Fission matrix, as in Chapter 6.

Due to the enormous potential economic and social benefits of nuclear energy and the unprecedented safety requirements, the simulation of nuclear reactors has been investigated with a high level of detail over the last seventy years. Although novel work is ongoing, and unsolved problems exist in areas such as the stable coupling of differential equations for neutron transport,fluid dynamics, heat transfer and material degradation [228] and some uncertainty remains in the libraries of neutron cross-sections used, very accurate simulations of reactors are now possible (e.g. [114]). However, the opportunity exists to investigate models of nuclear systems with a very low computational cost for the purposes of optimisation. In light of this opportunity, the objectives of the thesis are:

- To create a framework for the evaluation of *SMO* problems
- To test the framework on a number of different nuclear fuel management scenarios
- To develop a number of surrogate model designs and evaluate their suitability in the above scenarios
- To evaluate the value of these *SMO* techniques as applied to in-core PWR fuel management

1.4 Guide to the Thesis

This chapter introduces the concept of fuel management in PWRs, as well as key concepts in multi objective optimisation and the core concept of *SMO*. Chapter 2 provides a detailed introduction to a number of current techniques for creating surrogate models. Chapter 3 reviews the literature around *SMO*, deep learning and fuel management in *PWRs*.

Chapter 4 introduces the methods and tools used to achieve the goals of the thesis, Glasstone and Sesonke's model (introduced in Section 1.1.1) will be used as a template for the simplified process described in Chapter 4 of this thesis.

Studies using the framework of software tools are presented in Chapters 5, 6 and 7. In Chapters 5 and 6 two radically different surrogate modelling approaches are developed and evaluated. The aim of Chapter 5 is to introduce and investigate the suitability of deep learning surrogate models, which have been successfully applied in a number of other industries.

Chapter 6 investigates the possibility of converting the well understood statistical model, the fission matrix, into a surrogate model for the purposes of optimisation. Chapter 7 considers the application of novel experimental hardware, *quantum annealers (QAs)*. Chapter 8 considers the three different surrogate models in the context of the contribution to nuclear engineering and the trends in optimisation research.

1.5 Summary

This chapter has attempted to outline the context of *PWR* fuel loading pattern optimisation. This is an ongoing problem, and one that is solved manually on a routine basis. The approaches fall into two major categories: Heuristic approaches based on rules adhered to by an expert operator, such as those of Galperin [72], and Metropolis or iterative type approaches used by Parks [175] and others [84, 219, 60, 120, 33].

This thesis aims to extend the iterative approach, by investigating surrogate modelling techniques as a way to contend with the combinatorial explosion of solutions which commonly complicate the problems. Chapters 5 and 6 will introduce novel techniques for this.

Heuristics create simplified models that are easy to optimise but do not necessarily represent the problem, in Chapter 7 a quantum simulated annealing approach is used to encode a simple set of heuristic rules. In this way, surrogate modelling is able to show a connection between these disparate approaches. The problem is so large [71] that concluding that the global optimal solution has been found cannot normally be done with this method.

This thesis is an investigation of *SMO*, methods that use regression or other simplified modelling techniques to reduce the computational complexity of the search space. These methods have become increasingly popular, and are beginning to be successfully applied in the field of nuclear engineering.[5]

[5] Code used in this section is available for audit, reproducibility and derived works. A copy can be obtained from the repository under the permissive two-clause Berkeley Standard License [3]:

https://bitbucket.org/ajw287/chapter1-demos.git

Chapter 2

Surrogate Model Optimisation Techniques

2.1 Introduction

Surrogate Model Optimisation (SMO) has been applied successfully in many different areas of engineering to optimise a function that is expensive to sample. In contrast, there have been limited examples of surrogate methods being applied in nuclear engineering. When surrogate methods have been used in nuclear engineering, for example Faria and Pereira [60] or Kim *et al.* [107] the language has not been used. So there is potential for conceptual advantages of the approach to be brought to the subject by a thorough study. This chapter reviews a number of methods of surrogate modelling for engineering problems in general, with a view to application in nuclear engineering. The choice of mathematical notation has been modified somewhat from reference texts to be compatible with common symbols used in nuclear engineering to aid clarity during application.

2.1.1 Surrogate Model Definition

In 2018 Skinner and Zare-Behtash [200] defined a surrogate model as:

> "Surrogate assisted optimisation aims to alleviate the computational burden of the [...] optimisation process by defining a simplified mathematical relationship allowing for fewer numerical simulations to be required" [200]

This is usually a simplified model of a complex, computationally expensive computer simulation, and the usual justification for a surrogate model is decreased computational expense [213].

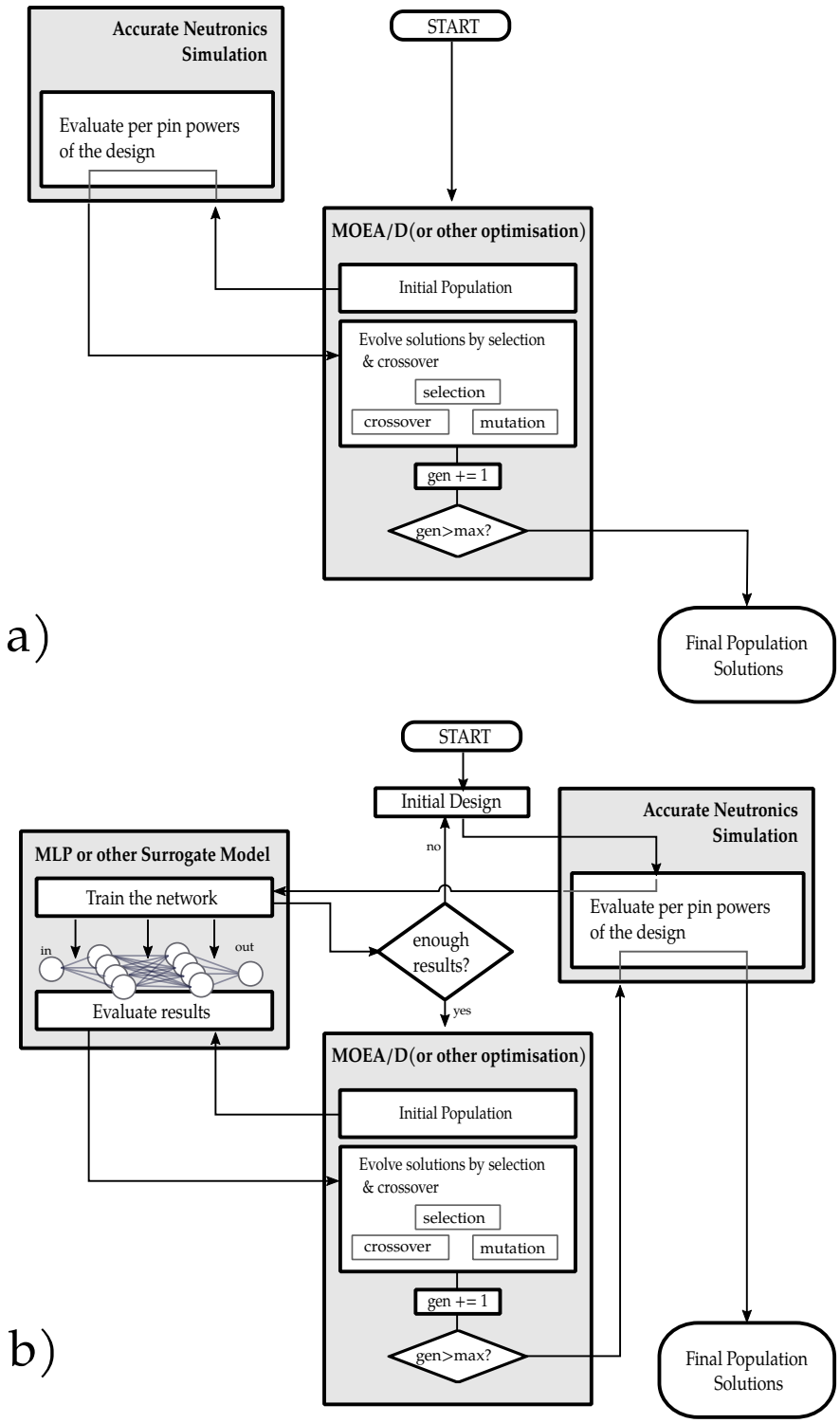


Fig. 2.1 Architecture of (a) direct iterative optimisation, and (b) surrogate model optimisation

However, all mathematical models of a system in engineering can be considered to be simplified models. So, this thesis will use the following definition:

A surrogate model is a simplified model of the system that is being studied that is not generated by adherence to the original physics of the system.

The surrogate model is therefore a metaphorical model that apes the complex physical process to allow exploration of the search space in the surrogate model more economically than results could be obtained from a physics based simulation. Surrogate models can also be used to model real-world systems where experimental sample collection takes a long time or large cost.

Direct iterative optimisation techniques follow a flow diagram similar to Fig. 2.1a. In these algorithms, solutions are tested and, by a variety of techniques ,improvements to the design are made. This kind of optimisation relies only on very basic assumptions about the objective function, which may not be differentiable.

The problem can be represented by an objective function $f(\mathbf{x})$, with an input vector $\mathbf{x}$ of length d, called the number of dimensions. The output vector $\mathbf{y}$, which may be single valued or multiple valued, is defined such that $\mathbf{y} = f(\mathbf{x})$, furthermore visibility of $f(\mathbf{x})$ is usually only available through discrete samples. Figure 2.1(b) shows a flow diagram for SMO: first, a training set is created; machine learning techniques are applied to the training set to create a model; optimisation takes place using the model; then the resulting optimised solutions are tested with the original accurate neutronics simulation.

SMO is useful when a high cost is associated with obtaining these samples from $f(\mathbf{x})$. The cost that is incurred to retrieve a sample can take a variety of forms. For instance, cost could be computational, as is the case for neutron transport calculations or for *CFD*. It could be monetary, for example when recovering mining core samples. There might be temporal cost, when predicting events like earthquakes or a less tangible cost, for example when samples require a biopsy. Whatever form the cost takes, it should be economical in terms of development effort to create a surrogate function $\hat{f}(\mathbf{x})$, compared to collecting samples directly from $f(\mathbf{x})$.

The aim of the surrogate model is to approximate sufficiently $f(\mathbf{x})$ accurately for the required purposes, which are assumed to be optimisation of the engineering problem, f in terms of $\mathbf{x}$. Methods are normally extensible to objective functions that are multi-variate in terms of inputs and outputs (called *MIMO*). It is common practice to use the univariate version $y = f(x)$ when discussing aspects of the methods that are not directly related to the multi-variate aspects.

2.1.2 Surrogate Modelling Methods

Skinner and Zare-Behtash [200] recently reviewed surrogate models for aerospace design covering *polynomial regression* models, *Radial Basis Functions (RBFs)*, *ANNs*, and *Kriging*.

Another recent study, Palar *et al.* [172], lists the following surrogate modelling methods: *polynomial regression*, *RBFs*, *Kriging*, *Support Vector Regression (SVR)*, *ANNs* and Hu *et al.* [88], who also mention the method of non-intrusive polynomial chaos expansions (NIPCE). This chapter will describe *polynomial regression* models, *RBFs*, *Kriging*, *SVR* and *ANNs*. For historical reasons each method has evolved separately and different labels are used for similar concepts. An attempt has been made to use a uniform mathematical lexicon across the methods, to demonstrate the similarities and promote a comprehensive understanding of the subject.

Skinner and Zare-Behtash [200] split surrogate models into *parametric* and *non-parametric* and by regression method:

> "Parametric approaches (such as kriging or polynomial regression) are model dependant [*sic*], forming a functional relationship between the response variables and the design variable samples that are known. Non-parametric approaches (such as radial-basis functions or neural networks) use local models in different regions of the sample data to build-up an overfall [*sic*] frame work of the model. Furthermore, surrogates can also be classified into regression type (polynomial regression, radial basis functions) which tend to be better suited to noisy functions, and interpolation type (kriging) creating best-fit response models." [200]

The truth is more complex. This chapter will show that these divisions, though conceptually useful, are somewhat arbitrary. Although *polynomial regression* and basis function methods form two basic methods of regression, it is possible to have parametric basis function methods, meaning that the definitions are not mutually exclusive.

2.2 Sampling Plans

Each of the metaphorical methods of system modelling discussed in this chapter are generated by choosing a flexible surrogate function $\hat{y}$, and fitting it to the objective function y by taking results or samples. In this section a consideration of the sampling plan is made. Sampling of the output function $f(x)$ for a d dimensional design problem $x \in D \subset \mathbb{R}^d$ with n samples, can be defined as:

$$\left\{x^{(i)} \rightarrow y^{(i)} = f(x^{(i)})|_{i=1,\ldots n}\right\} \tag{2.1}$$

In order to have certainty that the samples in (2.1) are representative of f, the *sampling plan* should be selected to effectively cover the domain of the problem. When d is large, the *curse of dimensionality* makes effective sampling vastly more difficult.

2.2.1 Types of Sampling Plan

Early work in sampling is described by Wang and Shan [227] as coming from the theory of design of experiments, and concentrated samples at the boundaries. Since computers mainly produce systematic errors rather than random errors it was argued by Sacks *et al.* [192] that space filling methods were superior.

Pure Random Sampling

The simplest method of generating a sample set is to randomly select each value of x, such that $x \in \mathscr{U}(a,b)$. This method appears attractive on initial inspection as each dimension is considered independently and there is no preference for a particular value. However, there is also nothing to *stop* clustering of sample points, so should a cluster occur then the model may be trained and fit the clustered samples well. If a surrogate model is trained on a set of samples that are clustered, it is likely to perform poorly in predicting the objective function for points outside the cluster. The probability that points generated by random sampling will cluster becomes lower as the number of samples increases. So, a larger set of samples is required to reduce the probability of a cluster occurring to an acceptable level.

A further problem can be observed with random sampling in multivariate systems. Here if the system is nonlinear, then sensitivity to the interaction of variables means that the sampling method may not adequately cover areas of the search space that are of interest. For example, in a system that is sensitive to the mean of a set of input variables, random sampling would generate sets of data that tend towards a Gaussian distribution of the input mean due to *central limit theorem*. In Section B.1 an attempt is made to generate a random set that is as close to flat in terms of mean value as possible.

Although the pure random sampling method has been justifiably out of favour for many years, a resurgence has been seen in random sampling methods. This is due to the increase in performance of computers for applications where there is a high dimensionality of the input vector; here the benefits of alternative methods of sampling are lower. Random sampling has been found to be advantageous for highly nonlinear systems [142]. Methods such as Latin Hypercube Sampling require many more computer operations to generate the initial samples, so it can be practical to use random sampling when the dimensionality is very high.

Systematic Sampling

An intuitive method of generating samples is to create samples at regular intervals across the search space. For each input dimension, n samples are selected over a range R, at a value change of k where $k = R/n$. Each permutation of the input samples must be evaluated to establish a

hypergrid shaped sample space. The input sample space grows with d as n^d; so, as the number of inputs increases, the number of permutations of inputs rapidly increases. Essentially, in developing a training set, it is important to be able to *'sample until'* [183, p. 313] the surrogate model can be trained. The problem with systematic uniform sampling is that the granularity of the sampling must be decided in advance. Uniform random sampling is advantageous here as it is unbiased and because the problem is very multimodal, making the likelihood of inputs being clustered low.

Latin Hypercube Sampling

Latin hypercube sampling ensures that there is a uniform distribution of samples by dividing the range by the number of samples and selecting a random point in each division of the range for each dimension. The approach is attributed to McKay [150] and the method was shown to provide significant improvements in performance on computers at the time. Since clustering cannot occur, it was shown that the results converge more quickly in an example using estimators of a *CFD* problem [150]. As seen in Figure 2.2, Latin hypercubes are not necessarily structure-free. As early as 1989, Hunter [89] warns that the Latin hypercube produces biased averages as a result of the structure in the sampling programme for selected problems.

Orthogonal Sampling

Orthogonal sampling was proposed by Tang [215] in 1993. It is a modification of *Latin hypercube sampling*, where each region of the space is selected like in *Latin hypercube sampling*. However, there is also a criterion that each $d-1$ dimensional section contains a number of samples, similar to a sudoku puzzle. Figure 2.2 shows that this avoids accidental structure occurring in the samples by ensuring that each variable is equally sampled in the subsection of the space.

Sobol Sampling

$LP\tau$ or Sobol sequence sampling was proposed by Sobol in 1979 [207]. The description here is based on the Anotonov and Saleev variant of Sobol Sampling ([11] article in Russian) described by Bratly and Fox [27] and Press *et al.* [183]. Sobol sampling is a method of creating a sample set that is uniformly and spread across the search space and each new sample fills gaps compared to previous samples. Thereby minimising clustering, like orthogonal sampling, with the advantage that the samples are deterministic, with new values in the sequence filling empty spaces between samples.

Sobol sequences can be generated for arbitrary numbers of dimensions leading to the potential to generate deterministic training sets that have a known value of space coverage.

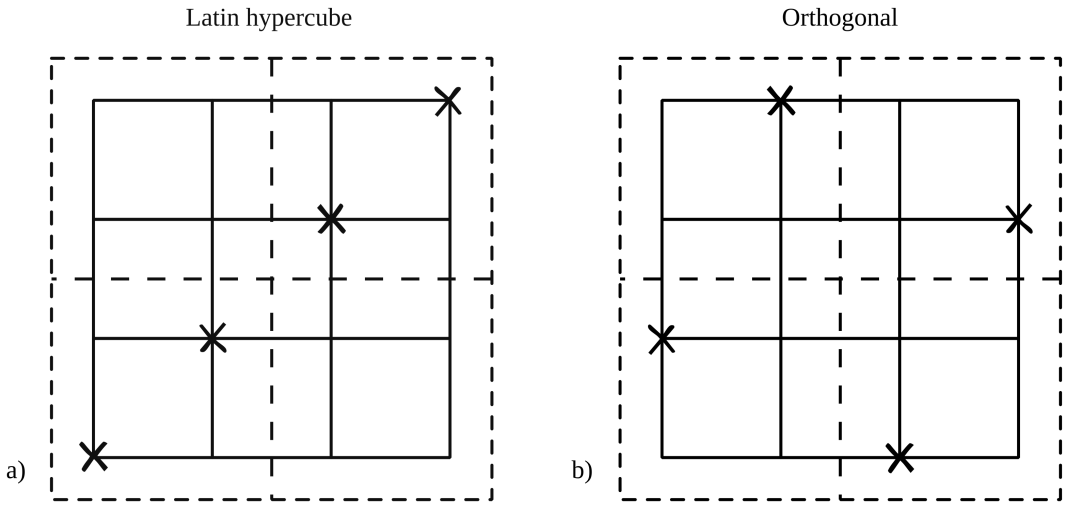


Fig. 2.2 (a) an example of a legal (albeit specifically selected) Latin hypercube; and (b) an example of the orthogonal sampling. Orthogonal sampling demonstrates better spreading by enforcing a sample in each quadrant of the space. Diagram adapted from Tang [215]

Other Methods

Random sampling and Latin hypercube represent the most common sampling methods in use today [39]. Other methods exist, such as including proposed solutions from Faure, Niederreiter and others [183, p 314]. Research continues in this area, with Montgomery *et al.* [154] proposing three new methods aimed at unbiased experiment design. In Appendix B, Section B.1 the author experiments with a sampling plan that combines random variables in an attempt to create a uniform distribution. As the problem landscape changes, for example when computers get faster, methods become more or less favourable.

2.2.2 Prediction-based Exploitation

When the surrogate model is used to identify optimal locations to explore, it is called *prediction-based exploitation*. This is done by taking an initial sample set from f, generating a surrogate model $\hat{f}$, then using $\hat{f}$ to generate solutions in order to identify input 'neighbourhoods' where the output variables appear to be optimal. The objective function f is then sampled around this area and the model is improved, or *localised* based on the new samples (termed *in-fill* samples) (*c.f. prediction-based exploitation* with *error-based exploitation* discussed on page 34).

The strategies for sampling in the previous sections are aimed at one stage sampling or for generating the initial sampling plan. They aim to find a way to spread samples across the search space. It is generally assumed that the number of samples is limited by the expense of collection, and so the problem is 'solved' by finding an acceptable solution or when the amount of resource available to spend on gathering samples is exhausted. When online sampling, Sórbester *et al.* [206] have shown that, for a defined maximum number of samples, it is advantageous to spend $0.35n$ of the n samples on the initial sampling plan and the remainder of the samples on in-fill samples. If a limit to the number of samples is set and in-fill sampling is not to be used, then there is not a clear way to establish how many samples are required. Sórbester refers to some researchers using *'rules of thumb'*:

> "Based on past experience, we find that about $n = 10d$ points are needed in the initial design." [97]

Considering the nature of regression, the number of samples required is dependent on the complexity of the function being modelled. Despite the academic value of such a *'rule of thumb'* being questionable, it is useful to know what other researchers have found to be sufficient in the context of real problems.

2.3 Evaluating Accuracy of Estimates

In order to evaluate the accuracy of a surrogate model it is important to use tools that can quantify the error of $\hat{f}(\mathbf{x})$ compared to $f(\mathbf{x})$. The simplest measure of error of a set of points to another is the *Root Mean Squared Error (RMSE)* defined in (2.2):

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n_t}(y^{(i)} - \hat{y}^{(i)})^2}{n_t}} \tag{2.2}$$

The *RMSE* forms the basis of much of the work to minimise errors between $\hat{y}$ and y. However, another metric, called the *correlation coefficient*, is important for the development of surrogate models for optimisation. In optimisation the objective is to develop a globally accurate model for the purpose of establishing important input configurations. The correlation coefficient compares the landscapes of the functions and not the values. It is defined by Forrester *et al.* [67, p. 37] as:

$$r^2 = \left(\frac{cov(y,\hat{y})}{\sqrt{var(y).var(\hat{y})}}\right)^2 \tag{2.3}$$

and expands to:

$$= \left(\frac{n_t \Sigma_{i=1}^{n_t} y^{(i)} \hat{y}^{(i)} - \Sigma_{i=1}^{n_t} y^{(i)} \Sigma_{i=1}^{n_t} \hat{y}^{(i)}}{\sqrt{\left[n_t \Sigma_{i=1}^{n_t} y^{(i)2} - (\Sigma_{i=1}^{n_t} y^{(i)})^2 \right] \left[n_t \Sigma_{i=1}^{n_t} \hat{y}^{(i)2} - (\Sigma_{i=1}^{n_t} \hat{y}^{(i)})^2 \right]}} \right)^2 \tag{2.4}$$

These two methods of measurement of the error and correlation of $\hat{f}$ to f form the basis of correction of surrogate models. It is worth noting that the *RMSE* is partly designed to scale the error for the purposes of human inspection. An equivalent measurement is the *Mean Absolute Error (MAE)* which preferred since it is more economical computationally. Some surrogate development schemes use a variety of error measurement techniques such as $log(error^2)$ in order to establish a greater penalty for larger errors, for example Chollet *et al.* [37].

2.4 Surrogate Model Construction

2.4.1 Polynomial Models

Polynomial regression models are generated from the sum of weighted polynomials to reconstruct f, as shown in (2.5). Each order of the polynomial is multiplied by a real number, called a weight, $\mathbf{w}$. The *Response Surface Model (RSM)* is a term coined in 1951 by Box and Wilson [26] while working on polynomial regression models. *RSM* might be considered more general, but it has become associated with polynomial regression models to the point where it is now ambiguous and the term *polynomial regression* model will be used in this thesis.

$$\hat{f}(x,m,w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_m x^m \tag{2.5}$$

As noted by Box and Draper in 1987, a polynomial expression of order m that best approximates a function $f(\mathbf{x})$ would be the Taylor expansion of f, truncated to $m+1$ terms [25, p. 423]. It follows logically that a higher order model would fit the sample set more effectively. However, as the order increases the model becomes more *specialised* for the dataset that created it. This is particularly true for higher order models which diverge faster than lower order models. If the data contains noise (from systematic error in simulations or measurement error in experimental data), then there is a balance between generalising the data to a low order model that ignores outlying data and *overfitting* f to too high an order that will not predict well for new samples. This issue is known as an *overfitting* problem.

For a single output example, with n sample points of x, where y has been established, the basis for creating a *polynomial regression* model from a set of points using least squares

estimation is the *Vandermonde matrix,* $\mathbf{V}$, as shown in (2.6):

$$\mathbf{V} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} \tag{2.6}$$

Then by using the *Least squares criterion* for estimation:

$$\vec{w} = \left(\mathbf{V^T V}\right)^{-1} \mathbf{V^T y} \tag{2.7}$$

$\left(\mathbf{V^T V}\right)^{-1} \mathbf{V^T}$ is known as the Moore-Penrose pseudo-inverse, $\mathbf{V}^+$, and is widely supported in mathematical programming languages.

In order to extend a *polynomial regression* model to more outputs, the complexity increases rapidly with the dimensions required for the $\mathbf{V}$ matrix dimensionality. Furthermore inputs are considered to be independent variables, so interaction of these variables cannot be modelled. Typically this is described as the difficulty describing a continuous approximation of an *XOR gate*, where an input, x_i, changes the effect that the other input, x_j, has on the output. A realistic example could occur where a burnable poison in a nuclear reactor fuel pin affects the burnup of fuel in adjacent pins. The number of neutrons being produced by both of these pins would then affect other pins that are further away so the input variables (fuel enrichment and poison mass fraction) cannot be transformed linearly to produce accurate pin power predictions.

As previously mentioned, selection of optimal order m is critical to *polynomial regression*. If the order is too high, then the model will not extrapolate well beyond the data points. If the order is too low, then the model will not fit the data well.

Order Estimation

In order to carry out polynomial regression, a choice of m must be made, and this can be done in a number of ways. If domain-specific knowledge can be leveraged, then it should be. For example, if the task were to model the motion of an object under a force, it can be expected that it will have a trajectory that follows a cubic law for position. In this case, selecting a higher order polynomial could be predicted to exhibit *overfitting* on the data.

Since m is always an integer and unlikely to be a large number, it is usually feasible to estimate the order by *brute force* [67, p. 40]. The *cross-validation* ε_{cv} is calculated by splitting the samples into a number of groups and then estimating the prediction error for each group when the model is trained on the rest of the sample data. This process must be repeated for

each group:

$$\varepsilon_{cv}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}[y^{(i)} - \hat{f}(x^{(i)}, w')]^2 \tag{2.8}$$

where w' are the weights associated with each polynomial $\hat{f}$. For a given order of $\hat{f}$, the mean cross-validation $\bar{\varepsilon}_{cv}$ when trained on each of the subsets of data left after the removal of the test group, gives an estimate of the prediction error [67, p. 36].

As the number of groups tends to the full set of data (albeit $n-1$ is the maximum), then ε_{cv} becomes a better estimate of the interpolation *inside* the sample range. There is a trade-off between the number of groups and computational expense for calculating the cross-validation.

Polynomial Models Summary

Polynomial models are relatively easy to understand and provide very computationally-efficient surrogate models. They are also extremely useful in the quantification of uncertainty of a model. Three major disadvantages of traditional polynomial fitting are the inability to model nonlinear combinatorial problems, the *curse of dimensionality* and *overfitting*.

If the problem is a *MO*, then the dimensionality of $\mathbf{V}$ increases linearly. This means that the number of calculations required increases very quickly and the problem becomes computationally expensive. However, recent advances made in the computation of polynomials have changed the *status quo*. For example, the *Effective Quadratures* library allows computationally-efficient polynomial regression in many dimensions, Seshadri and Parks [195].

2.4.2 Radial Basis Function Models

The term Radial Basis Function (RBF) refers to any *function* that is symmetrical (*radial*), and is used as the *basis* of the surrogate model. They are also known in the literature as *Radial Basis Function Networks*. In this thesis $g(r)$ will be used to define the *RBF*, where r is the Euclidean distance $|x - c^{(i)}|$, although an interpolation step can be used to estimate c (see [179] for a description). This text will describe the case where the basis functions are centred at the sample points, $c^{(i)} = x^{(i)}$, and where n_c is the number of samples:

$$\hat{f}(x) = \mathbf{w}^T\mathbf{g}(\mathbf{r}) = \sum_{i=1}^{n_c} w_i g\left(|x - c^{(i)}|\right) \tag{2.9}$$

The advantage of using radial functions where r is the radial distance (e.g. r^3 or a thin plate spline $r^2 ln(r)$) is strictly associated with the intuitive understanding that real-world functions are continuous, and so values close to a sample will have a value relation to the sample. Symmetry is not necessarily true in real systems, but, without *a priori* system knowledge, it

is a reasonable assumption. A number of possible choices for $g(r)$, the *RBF*, are shown in Table 2.1. Insight may be used by the practitioner to match the basis function to the real-world application. For example, if the objective function were to involve the Brownian motion of gas particles, then a Gaussian basis function would be the natural choice, since the random motion of a population of particles in a medium naturally forms a Gaussian function [19, pp. 6–10].

Table 2.1 Table of *RBFs*, examples described by [67, p. 46].

Basis Function	**Estimation Complexity**	**Comments**
linear $g(r) = r$	very low	Linear decay with distance r
cubic $g(r) = r^3$	very low	Cubic decay with distance r
thin plate spline $g(r) = r^2 \ln(r)$	low	A spline curve r
multiquadratic $g(r) = (r^2 + \sigma^2)^{1/2}$	high	More general approximation, due to σ parameter
Gaussian $g(r) = e^{-r^2/2\sigma^2}$	high	More general approximation, due to σ parameter, *c.f.* Kriging, p. 33

The weight vector can be estimated independently of the basis function, by substituting the *Gram matrix (***G***)* defined as:

$$\mathbf{G}_{i,j} = g\left(|x^{(i)} - c^{(j)}|\right), \qquad j = 1, \ldots, n \tag{2.10}$$

Substituting (2.10) into (2.9) and switching to matrix notation:

$$\hat{f}(x^{(j)}) = \mathbf{wG} = \mathbf{y}^j, \qquad j = 1, \ldots, n \tag{2.11}$$

From (2.11) the weights can be established by:

$$\mathbf{w} = \mathbf{G}^{-1}\mathbf{y} \tag{2.12}$$

Note that (2.12) is identical to the *error backpropagation* algorithm used in neural networks with a single hidden layer (see Artificial Neural Networks p. 38). If the radial basis function has other parameters, such as σ, then these must also be established. This is covered in the section on Kriging on p. 33.

RBF Summary

Radial Basis Functions represent the objective function f as the sum of basis functions. This is an attractive method due to the ease of implementation. For some situations *RBF* methods can be shown to be a universal predictor [67, p. 75]. However, how well the surrogate model fits the objective function can be basis function dependent, so physical insight is useful. Radial basis functions have been applied in many areas of engineering, including aerodynamics [63, 131], mechanical engineering [110] and physical geography (see [188] for a good summary).

2.4.3 Kriging

Kriging is a special case of an *RBF* approach, with a basis function shown in (2.13). This method was developed by Matheron [146]. Kriging aims to carry out the reverse process of the Weirstrass transform described in Equation B.4. Due to its popularity, features and independent development, it is usually considered separately from other *RBFs*.

$$g^{(i)} = e^{\left(-\sum_{j=1}^{d} \theta_j |x_j^{(i)} - x_j|^{\mathbf{p}_j}\right)} \tag{2.13}$$

The Kriging basis function, (2.13), exhibits a variable vector $\mathbf{p}$ that varies the exponent and the θ from the Gaussian equation in Table 2.1 is replaced with an n dimensional vector $\theta = \{\theta_1, \theta_1, \ldots, \theta_n, \}$, so a Kriging model can contain basis functions with different widths.

To generate a Kriging model, the definition of correlation, (2.3), is used on the vector of outputs $\mathbf{Y}$ to generate a correlation matrix:

$$\mathbf{R} = \begin{bmatrix} r[Y(x^{(1)}), Y(x^{(1)})] & \ldots & r[Y(x^{(1)}), Y(x^{(n)})] \\ \vdots & \ddots & \vdots \\ r[Y(x^{(n)}), Y(x^{(1)})] & \ldots & r[Y(x^{(n)}), Y(x^{(n)})] \end{bmatrix} \tag{2.14}$$

Taking roots, and rearranging (2.3):

$$cov(y, \hat{y}) = r.var(y).var(\hat{y}) \tag{2.15}$$

Then, substituting y for $\hat{y}$, the covariance matrix is found to be:

$$\mathbf{Cov}(y, y) = var(y)^2 \mathbf{R} \tag{2.16}$$

In order to estimate θ and p, an assumption is made that ε is an independent random distribution in $\mathbf{x}$ and $\hat{y} \rightarrow y$, using σ^2 to denote variance. There is then a logical inversion of the conditional

probability for a normal distribution to say that *Likelihood L* is :

$$L(\mathbf{Y}^{(1)},...,\mathbf{Y}^{(n)})|\mu,\sigma) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{\left(-\frac{\sum(\mathbf{Y}^{(i)}-\mu)^2}{2\sigma^2}\right)} \tag{2.17}$$

Substituting in (2.16) puts (2.17) in terms of the sample data:

$$L = \frac{1}{(2\pi\sigma^2)^{n/2}|\mathbf{R}|^{1/2}} e^{\left(-\frac{(\mathbf{y}-\mathbf{1}\mu)^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{1}\mu)}{2\sigma^2}\right)} \tag{2.18}$$

and taking natural logs:

$$ln(L) = -\frac{n}{2}ln(2\pi) - \frac{n}{2}ln(\sigma^2) - \frac{1}{2}ln(|\mathbf{R}|) - \frac{(\mathbf{y}-\mathbf{1}\mu)^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{1}\mu)}{2\sigma^2} \tag{2.19}$$

By taking derivatives w.r.t. μ and σ to find the maximum likelihood, the parameters $\hat{\mu}$ and $\hat{\sigma}$ can be found to be optimal at:

$$\hat{\mu} = \frac{\mathbf{1}^T\mathbf{R}^{-1}y}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}} \tag{2.20}$$

and

$$\hat{\sigma}^2 = \frac{(\mathbf{y}-\mathbf{1}\mu)^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{1}\mu)}{2\sigma^2} \tag{2.21}$$

which substitutes back into (2.19) to create a polynomial time calculation of $ln(L)$:

$$ln(L) \approx -\frac{n}{2}ln(\hat{\sigma^2}) - \frac{1}{2}ln(|\mathbf{R}|) \tag{2.22}$$

Although (2.22) is in terms of σ and R, it cannot be differentiated, so the values θ and p must be found by an iterative search method, such as simulated annealing, tabu search or an evolutionary algorithm.

Error-based Exploitation

An interesting feature of the Kriging method is that it allows an estimation of the error of $|f(x) - \hat{f}(x)|$. When this is used to guide the sampling plan, it is termed *error-based exploitation*. This can be useful to guide the sampling plan to generate in-fill samples that generate the most accurate surrogate model possible.

Kriging Summary

Kriging is one of the most popular methods for surrogate modelling in use today. It has been widely applied in aerospace, mining, physical geography and nuclear engineering [76, 188, 189, 199, 242]. It can be seen as a special case of *RBFs* or as a method of Gaussian Process Regression, since the radial basis function is a normal curve.

2.4.4 Support Vector Regression

Support Vector Regression works to minimise the number of points in the collected samples $y_{(i)}|i = 1,\ldots,n$ that have an $\hat{f}(x_{(i)})$ error that is greater than ε.

The surrogate function for *SVR* shown in (2.23) is similar to an *RBF*. However, it is important to note that the parameters bias b and weights $\mathbf{w}$ are calculated in a different manner [67, p. 64]:

$$\hat{f}(\mathbf{x}) = b + \sum_{i=1}^{n} w^{(i)} g(\mathbf{x}, \mathbf{x}^{(i)}) \tag{2.23}$$

The description of *SVR* will follow the same process as [202], on which this description is based, by starting with a linear regression of one output y, then generalising. For a linear regression of f, so (2.23) becomes:

$$\hat{f}(\mathbf{x}) = b + \mathbf{w}\mathbf{x} \tag{2.24}$$

where $\mathbf{w}$ is a vector of real numbers and b is a real number. The selection of optimal parameters is given by:

$$\begin{aligned} &\underset{w}{\text{minimise}} \quad \frac{1}{2}|\mathbf{w}|^2 \\ &\text{subject to} \quad \begin{cases} y^{(i)} - \langle w, x^{(i)} \rangle - b \leq \varepsilon \\ -y^{(i)} + \langle w, x^{(i)} \rangle + b \leq \varepsilon \end{cases} \end{aligned} \tag{2.25}$$

Since it is not always possible to find a line that fits all the points in a set with error less than ε, a slightly more complex function than (2.25) is needed. The solution is to incorporate a cost function to the minimisation term:

$$\begin{aligned} \underset{w}{\text{minimise}} \quad & \frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{n} \xi^{+(i)} + \xi^{-(i)} \\ \text{subject to} \quad & \begin{cases} y^{(i)} - \langle w, x_{(i)} \rangle - b \ \leq \varepsilon + \xi^{+(i)} \\ -y_{(i)} + \langle w, x_{(i)} \rangle + b \leq \varepsilon + \xi^{-(i)} \\ \xi^{+(i)}, \xi^{-(i)} \qquad \geq 0 \end{cases} \end{aligned} \tag{2.26}$$

where $\xi^{+(i)}$ and $\xi^{-(i)}$ are called *slack variables* and increase the sensitivity of $\hat{f}$ when $\left(f(x) - \hat{f}(x)\right) > \varepsilon$. Note that in this section bracketed superscripts are used to denote *different variables*, rather than powers. The volume that is taken up by $\hat{f}(x) \pm \varepsilon$ is called the ε*-tube*.

Using the Lagrangian to Solve the Dual Problem

In order to optimise 2.26, a *Lagrange function* is defined, by adding *dual variables* (also called *Lagrange multiplier variables*), η^+, η^-, α^+, and α^-, which must be maximised. The problem is then solved by searching for the *saddle point* where the original variables, $\mathbf{w}$ and b, are minimised and Lagrange multiplier variables have been maximised. The Lagrange function looks like:

$$\begin{aligned} L = & \frac{1}{2}|\mathbf{w}|^2 + \\ & C\frac{1}{n}\sum_{i=1}^{n}(\xi^{+(i)} + \xi^{-(i)}) \\ & - \sum_{i=1}^{n}(\eta^{+(i)}\xi^{+(i)} + \eta^{-(i)}\xi^{-(i)}) \\ & - \sum_{i=1}^{n}\alpha^{+(i)}(\varepsilon + \eta^{+(i)} - y^{(i)} + \mathbf{w}\cdot\mathbf{x}^{(i)} + b) \\ & - \sum_{i=1}^{n}\alpha^{-(i)}(\varepsilon + \eta^{-(i)} - y^{(i)} + \mathbf{w}\cdot\mathbf{x}^{(i)} - b) \end{aligned} \tag{2.27}$$

Although (2.27) looks complicated, it is fairly logical, with a similar expression for each of the variables. At the saddle point sought:

$$\frac{\partial L}{\partial \mathbf{w}} = 0 = \mathbf{w} - \sum_{i=1}^{n}(\alpha^{+(i)} - \alpha^{-(i)})\mathbf{x}^{(i)} \tag{2.28}$$

$$\frac{\partial L}{\partial b} = 0 = \mathbf{w} - \sum_{i=1}^{n}(\alpha^{+(i)} - \alpha^{-(i)}) \tag{2.29}$$

$$\frac{\partial L}{\partial \xi^{+}} = 0 = \frac{C}{n} - \alpha^{+(i)} - \eta^{+(i)} \tag{2.30}$$

$$\frac{\partial L}{\partial \xi^{-}} = 0 = \frac{C}{n} - \alpha^{-(i)} - \eta^{-(i)} \tag{2.31}$$

Substituting (2.28) into (2.24), then $\hat{f}$ for the case where $g(\mathbf{x})$ is a straight line, the surrogate function becomes:

$$\hat{f}(\mathbf{x}) = b + \mathbf{w} - \sum_{i=1}^{n} (\alpha^{+(i)} - \alpha^{-(i)})(\mathbf{x}^{(i)} \cdot \mathbf{x}) \tag{2.32}$$

The purpose of this substitution is to put the objective function in terms of *only* the dual variables. This is called the *dual objective function* and can be seen in (2.33).

$$\begin{aligned} \text{maximise} \quad & \begin{cases} -\frac{1}{2} - \sum_{i=1}^{n} (\alpha^{+(i)} - \alpha^{-(i)})(\mathbf{x}^{(i)} \cdot \mathbf{x}) \\ -\varepsilon \sum_{i=1}^{n} (\alpha^{+(i)} + \alpha^{-(i)}) + \sum_{i=1}^{n} y^{(i)} (\alpha^{+(i)} + \alpha^{-(i)}) \end{cases} \\ \text{subject to} \quad & \begin{cases} \sum_{i=1}^{n} (\alpha^{+(i)} - \alpha^{-(i)}) = 0 \\ \alpha^{+(i)}, \alpha^{-(i)} \in [0, C] \end{cases} \end{aligned} \tag{2.33}$$

The final variable that must be established is the *bias*, b. This can be established using a number of methods, either by analysis of (2.34) and (2.35) or by interior point optimisation, used for complex $g(x)$ as described by Keerthi *et al.* [103].

Smola and Schölkopf summarise the *Karush-Kuhn-Tucker (KKT)* conditions as:

> "These [Karush-Kuhn-Tucker (KKT) conditions] state that at the point of the solution the product between dual variables and constraints has to vanish." [202]

So:

$$\begin{aligned} 0 &= \alpha^{+(i)} (\varepsilon + \xi^{+(i)} - y^{(i)} + (\mathbf{w} \cdot x^{(i)}) + b) \\ 0 &= \alpha^{-(i)} (\varepsilon + \xi^{-(i)} + y^{(i)} - (\mathbf{w} \cdot x^{(i)}) - b) \end{aligned} \tag{2.34}$$

$$\begin{aligned} 0 &= (C - \alpha^{+(i)}) \xi^{+(i)} \\ 0 &= (C - \alpha^{-(i)}) \xi^{-(i)} \end{aligned} \tag{2.35}$$

Equations (2.35) show us that $\alpha^{\pm(i)} = C$ is a requirement for results $(x^{(i)}, y^{(i)})$ to be outside of the ε-*tube*, and therefore a *support vector*. A support vector cannot be simultaneously on both sides of the ε-*tube* so $\alpha^{+(i)} \alpha^{-(i)} = 0$ (one of them must be zero). From the second factors in (2.34) it can be seen that $\alpha^{+(i)}, \alpha^{-(i)}$ vanish for $|\hat{f}(x^{(i)}) - y^{(i)}| < \varepsilon$ so only samples that are outside of the ε-*tube* are required for the derivation of the surrogate model with *SVR*. This is a

very important result for *SVR* models. Figure 2.3 shows the final $\hat{f}$ function graphically for *SVRs*.

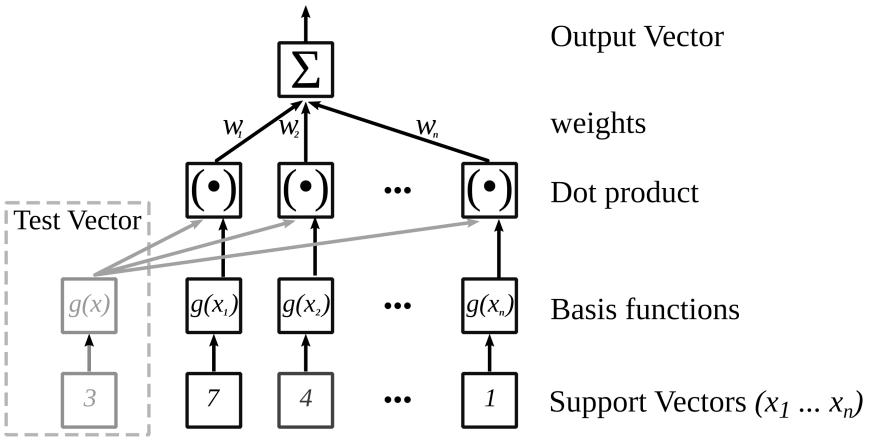


Fig. 2.3 The surrogate model $\hat{f}$ acting on a test vector in a trained Support Vector Surrogate Model, adapted from [202].

SVR Summary

This description of the construction of a surrogate function, $\hat{f}$, using *SVR* is only for a basis function, $g(x)$, which is a straight line. However, the method extends to more complex basis functions (the derivations of this can be found in [202]). The method allows the derivation of surrogate models from a sparse set of points, essentially only support vectors are required to completely model the system.

2.4.5 Artificial Neural Networks

ANNs are information processing algorithms inspired by one aspect of learning behaviour exhibited by biological neural cells. This origin story is usually distracting from the value of the algorithm, so this thesis will aim to keep the description relevant to surrogate models and nuclear engineering. However, it is worth noting that neural networks have been used in many other applications and have generated significant progress in areas such as character and image recognition, as well as video, speech and audio processing [123].

Neural networks have previously been used in nuclear engineering to predict core parameters, for example [163] (found in [220]) and in optimisation of core loading patterns by Faria and Pereira [60] and Kim *et al.* [107]. An *ANN*, such as the one shown in Figure 2.5, is

usually set up with low random initial weights between nodes. The process of *training* the network (modifying the parameters to make the output $\hat{f}$ approximate f well) is then typically as follows:

1. Initialise the network topology with low valued, random weights **w**.
2. Perform a feedforward evaluation of $\hat{f}$ to establish error ε and value at each node in the network.
3. Execute the *error backpropagation* algorithm (p. 39) to get error derivatives for each of the weights in the network.
4. Modify the weights using gradient descent
5. Terminate, if the number of iterations has been satisfied or the network has converged.
6. If not, then return to step 2.

The Backpropagation Algorithm

The method by which *ANNs* adapt to approximate the objective function is called *error backpropagation*. This algorithm is essentially a recursive application of the *chain rule for differentiation*. The creation of the backpropagation algorithm is often attributed to Rummelhart and Williams [191], though it is posited by LeCun *et al.* [123] that the algorithm has been discovered independently at least three times (the earliest, 1974 by Werbos [230, p. 9]). It is described below, following the procedure in the lecture by Winston [238], and based on the descriptions in [56] and [24, chap. 4] for *Multi Layer Perceptrons (MLPs)*.

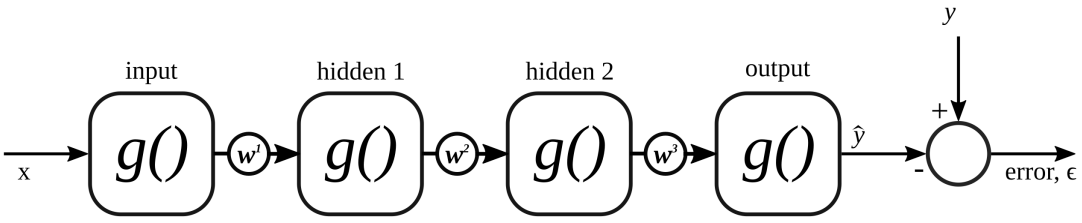


Fig. 2.4 A simplified single input, single output *ANN* with two single hidden layer nodes for explanatory purposes.

Consider an *MLP* such as shown in Figure 2.4. The layers of the network are input x, hidden 1 $h1$, hidden 2 $h2$ and output, $\hat{y}$. They are calculated as follows:

$$\begin{aligned} \hat{y} &= g(w^3\ h2) \\ h2 &= g(w^2\ h1) \\ h1 &= g(w^1\ g(x)) \end{aligned} \tag{2.36}$$

The transfer function is:

$$\hat{y} = g(w^3\ g(w^2\ g(w^1\ g(x)))) \tag{2.37}$$

Differentiating w.r.t. w^1 and using the chain rule, as mentioned previously:

$$\frac{\partial \hat{y}}{\partial w^1} = \frac{\partial}{\partial h2}\hat{y}\ \frac{\partial}{\partial h1}h2\ \frac{\partial}{\partial w^1}h1 \tag{2.38}$$

A simple error function ε is used, defined as:

$$\varepsilon = \frac{1}{2}(\hat{y}-y)^2 \tag{2.39}$$

and

$$\frac{\partial \varepsilon}{\partial \hat{y}} = (\hat{y}-y) \tag{2.40}$$

Equation (2.38) allows the calculation of the sensitivity of w_1 w.r.t. the error ε

$$\frac{\partial \varepsilon}{\partial w^1} = \frac{\partial \varepsilon}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h2} \cdot \frac{\partial h2}{\partial h1} \cdot \frac{\partial h1}{\partial w^1} \tag{2.41}$$

Although $g()$ (called the *activation function*) can be a wide variety of nonlinear functions, this example will use the sigmoid function, defined as follows:

$$g(a) = \frac{1}{1+e^{-a}} \tag{2.42}$$

The reason that the sigmoid function has been popular is due to a peculiar property of its derivative[1]:

$$\begin{aligned} \frac{dg(a)}{da} &= \frac{d}{da}(1+e^{-a})^{-1} \\ \frac{dg(a)}{da} &= g(a)(1-g(a)) \end{aligned} \tag{2.43}$$

Thus, the derivative of the sigmoid function can be expressed in terms of the original function. Using the sigmoid *activation function*, (2.42), and substituting its differential property, (2.43), into (2.36). It is now possible to substitute the results and (2.40) into (2.38) to create the result:

[1] Although the sigmoid function is used in this derivation. Activation functions such as ReLu are now considered superior [160] and will be used in the experimental section of this thesis.

$$\begin{aligned}\frac{\partial \varepsilon}{\partial w_1} = \; & (\hat{y}-y)\cdot \\ & w^3 g(w^3\ h2)(1-g(w^3\ h2))\cdot \\ & w^2 g(w^2\ h1)(1-g(w^2\ h1))\cdot \\ & w^1 g(g(x)\ w^1)(1-g(g(x)\ w^1))\end{aligned} \tag{2.44}$$

Equation (2.44) allows us to calculate the effect of changing w_1 based only on weights that are further down the network (which are calculated first in the weight update scheme 2.4.5) and *local variables*. These *local variables* have already been calculated in the forwards pass, item 2 in the list on p. 39. By using this method to modify weights, *ANNs* are able to scale linearly with the number of inputs, outputs and network complexity.

It has been shown above that it is possible to find the differential of the error w.r.t. each weight in a network with a single input and output. In the next paragraphs some housekeeping will be done to extend this to multiple input, multiple output networks, with multiple hidden layer neurons.

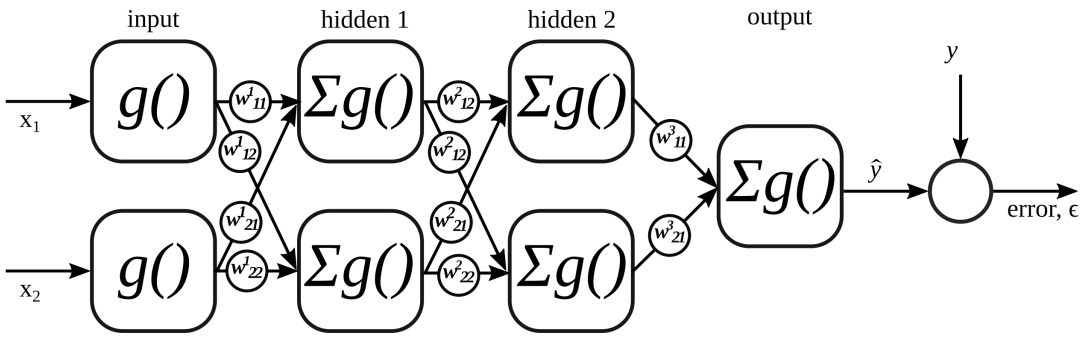


Fig. 2.5 A slightly more complex network, the weights $(\mathbf{w}^1, \mathbf{w}^2, \mathbf{w}^3)$ now form matrices.

Considering Figure 2.5, it is fairly easy to see that the weights will become matrices, and the inputs $\mathbf{x}$ can be converted to a vector. The transfer function from (2.37) becomes:

$$\hat{y} = \sum_{i,j} g(\mathbf{w}^3_{ij} \sum_{k,l} g(\mathbf{w}^2_{kl} \sum_{m,n} g(\mathbf{w}^1_{mn}\ g(\mathbf{x})))) \tag{2.45}$$

Similarly, the gradient of error for a given weight is given by (for example, with a weight in the first layer):

$$\begin{aligned}\frac{\partial \varepsilon}{\partial \mathbf{w}^1_{pj}} = \quad & (\hat{y}-y)\cdot \\ & g(\sum_{i,j}\mathbf{w}^3_{ij}\, g(\sum_{k,l}\mathbf{w}^2_{kl} g(\sum_{m,n}\mathbf{w}^1_{mn}\, g(\mathbf{x}))))(1-g(\sum_{i,j}\mathbf{w}^3_{ij}\, g(\sum_{k,l}\mathbf{w}^2_{kl} g(\sum_{m,n}\mathbf{w}^1_{mn}\, g(\mathbf{x})))))\cdot \\ & g(\sum_{k,l}\mathbf{w}^2_{kl}\, g(\sum_{m,n}\mathbf{w}^1_{mn}\, g(\mathbf{x})))(1-g(\sum_{k,l}\mathbf{w}^2_{kl}\, g(\sum_{m,n}\mathbf{w}^1_{mn}\, g(\mathbf{x}))))\cdot \\ & g(\mathbf{w}^1_{pj}\, g(\mathbf{x}))(1-g(\mathbf{w}^1_{pj}\, g(\mathbf{x})))\end{aligned} \tag{2.46}$$

The significance of the results in (2.46) should not be understated. It allows the fast computation of all derivative of weights of large networks with a great deal of repeated expressions in the calculation. Furthermore, multiple outputs can improve the learning rate of neural networks by increasing the value of the gradient of weights, making neural networks well suited to *MO* problems.

This section describes very simple feedforward networks called an *MLPs*. These are characterised by the outputs of each layer feeding into the inputs of the subsequent layer. However, a wide variety of network topologies have been experimented with [9, chap. 5] describes the derivation of *error backpropagation* for *feedback* networks, where the outputs of nodes can go to any layer, forwards or backwards.

Weight Optimisation

In order to modify the weights of the network, the amount that each weight should change can be modified by gradient decent:

$$\Delta \mathbf{w}_{ij} = -\alpha \frac{\partial \varepsilon}{\partial \mathbf{w}_{ij}} \tag{2.47}$$

where α is a constant called the *learning rate*. In order to start a training sequence, the weights should be initialised in the correct range:

> "With large initial weights, [*ANNs*] typically find poor local minima; with small initial weights, the gradients in the early layers are tiny, making it infeasible to train [*ANNs*] with many hidden layers" [85]

Gradient descent is not a very advanced optimisation technique; however, it is effective enough to find the global optimum if the weights are initialised with suitable values [85]. More advanced algorithms, such as the 'Adam' algorithm [109] have now superseded stochastic gradient decent.

Artificial Neural Network Topology

Relatively early work in *ANNs* showed that, like *RBFs*, even simple networks could act as universal function approximators [46], so work concentrated on simple networks until the recent interest in *deep networks* began, summarised concisely by LeCun *et al.* [123]. *Deep networks* showed a marked improvement on simple (or *shallow*) networks. Telgarsky [217] showed in 2015 that there are, in fact, functions that cannot be represented efficiently by shallow networks, and that a large set of functions are more easily represented in *deep networks*. A great deal of progress has been made in the last ten years. In 2016, Mhaskar *et al.* described the kind of problems for which deep or shallow learning is advantageous. They show that, for complex problems, orders of magnitude of improvement of error performance are achieved with three or more hidden layers. Even on only moderately complex problems the performance is shown to be better with *deep networks* [151].

Since the development of deep learning techniques, there has been an explosion of industrial usage of deep learning neural networks by companies like Google Inc. [6] and Facebook Inc. [96, 37]. These companies have released libraries of professionally developed code, under permissive licenses, allowing public, academic and commercial use.

Artificial Neural Network Summary

Khurana *et al.* [106] suggest that:

> "Neural networks work well when the relationship between the design space and the objective function is complex, less intuitive, and highly-dimensional." [200]

Comparison of (2.11) with the definition in (2.37) shows that *RBFs* are equivalent to a *MLP* with no hidden layers, $g(r)$ representing the neural *activation function* or radial basis function [179] (see pp. 32 and 40) . For this reason (and perhaps an interest in aligning the research) *RBFs* are often called *Radial Basis Function Networks* in the literature.

Although *ANNs* have been hyped a great deal, the advantages and flexibility of the approach, as well as the libraries of standardised code from international corporations, make this approach extremely attractive.

2.5 Summary of Methods

Nuclear engineering has a number of highly nonlinear, computationally expensive problems that make them ideal for surrogate modelling. Recent papers, for example Leniau *et al.* [126]

in 2015 and Wu *et al.* [242] in 2018, point towards an increasing trend of surrogate models in this area.

Although surrogate methods can be usefully described using the categories of *parametric*, where model parameters are tuned to fit the sample set and *non-parametric* where basis functions are added without modification and by regression type (polynomial regression or radial basis functions), we have seen that this is not the full story. Polynomial models are parametric, but radial basis functions can be parametric or not. Furthermore, most methods are equivalent as shown in Figure 2.6. Polynomial regression fits a polynomial curve of an externally chosen order *m* to the sample set. It is simple to implement but suffers computationally at higher input modality whereas Radial Basis Function Regression fits a set of basis functions with different central positions, to the data set.

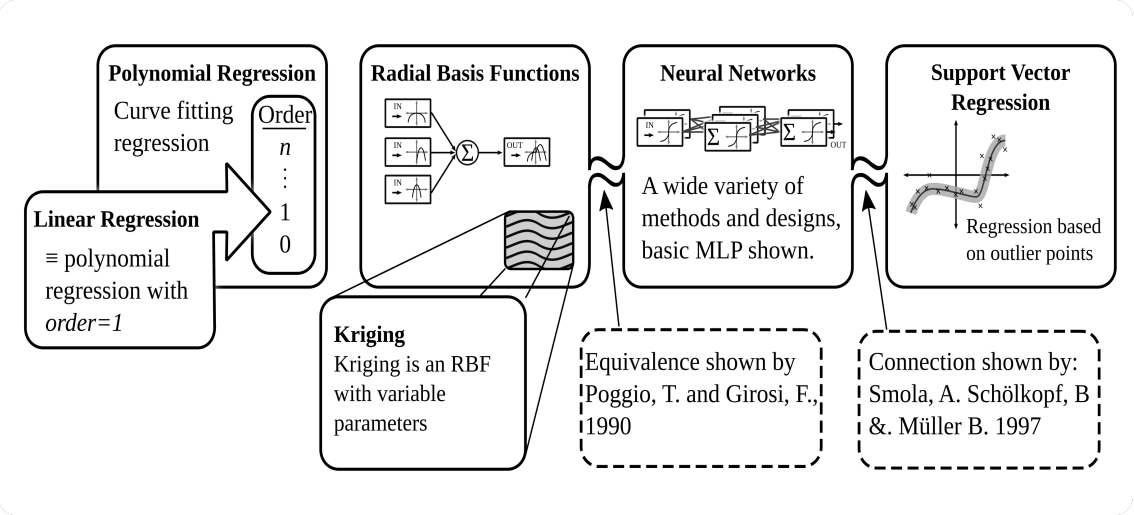


Fig. 2.6 Relationships between Surrogate Modelling techniques, references [179] and [203]

This chapter has described the implementation of five methods of surrogate model development. These are *polynomial regression*, *RBFs*, *Kriging*, *ANNs* and *SVR*. At first glance each of these methods appears disparate. However, the aim of this chapter has been to show the intimate connections between the different methods, by formulating each method using a common mathematical language.

Since there are many similarities between methods, an awareness of techniques from each method is important to avoid wasted effort. A great deal of work has been repeated in the last century that could have been avoided if researchers had been aware of efforts in related fields. Selection of a technique for further work is usually made based on an author's personal

preference, as remarked by Razavi *et al.* [188]. The author's choice of surrogate method for nuclear engineering is *ANNs* based on the nonlinearity of the problem requiring a method that can achieve extremely complex representations. Also recent innovations in the field may bring advantages and an active research community, coupled with the availability of comprehensive peer-reviewed libraries supplied under permissive licenses.

This chapter has described at a technical level a number of different surrogate models that have been used in surrogate model optimisation. The technical aspects of these technologies are intended to inform the reader when evaluating the application of these methods in nuclear engineering as well as in the wider engineering, discipline in the next chapter.

Chapter 3

Surrogate Model Optimisation, Deep Learning and PWR Fuel Management

While the previous chapter introduced *SMO* methods, this chapter examines the literature around in-core fuel management optimisation, *SMO* and the growing trend for deep learning in nuclear engineering.

Nuclear science has a long history of pioneering modern numerical techniques. For example, Monte Carlo [221] and iterative optimisation [174], but this would no longer appear to be the case. If the community continues to ignore modern techniques, it will inevitably struggle to attract talented scientists.

Until recently, there was almost no appetite for deep learning among the academic community in nuclear engineering. The proceedings of PHYSOR 2018 [1] included no papers with the terms 'machine learning', 'deep learning', 'neural network' or 'surrogate model' in the title. This bias has now been rightly discarded. There is now much, perhaps too much, enthusiasm for deep learning in nuclear engineering, and a rapidly growing corpus of literature. At the now cancelled PHYSOR 2020, there would have been eight papers presented that referenced 'machine learning', 'deep learning' or 'neural networks' in their titles [5]. The author is lucky to have worked at the forefront of these techniques, and has been able to contribute meaningfully to the adoption of deep learning techniques in *SMO* and as a part of the wider nuclear engineering community.

3.1 Applications of Surrogate Models

The following sections concisely describe *SMO* techniques that have been applied in other areas of engineering.

3.1.1 Mechanical Engineering

Examples of applying surrogate models in the area of mechanical engineering are concentrated in the area of *CFD*, but many other examples exist, including Yang *et al.* [248] who use surrogate models for vehicle frontal impact simulation, and Marklund *et al.* [143] who use first- and second-order polynomial approximations to investigate impact on the 'B-pillar' component of a car. Forrester and Keane [66] demonstrated the application of surrogate models to the design of a passive vibration isolating structure, creating novel performant structures, and later demonstrated helical spring design and cantilever 'Nowacki' beam design by using surrogate modelling [165],[67, pp. 198–201].

3.1.2 Computational Fluid Dynamics and Aerospace Design

Perhaps the most successful application of surrogate approximation for the purposes of optimisation is in the calculation of flows around objects in fluid dynamics. This is due to the high computational cost of performing *CFD* calculations, such as the direct numerical simulation of Navier-Stokes equations in three dimensions. *CFD* lends itself to surrogate methods due to the validity of the assumptions of continuity and conservation laws, and the similarity of the shape of responses with common basis functions.

Aircraft wing optimisation is an extremely popular aerodynamic optimisation problem for surrogate modelling. Robinson and Keane [189] derived a family of *RBFs* for use as surrogate models when optimising supercritical aerofoil morphology, and Keane [102] designed wing morphology using *Kriging* with multi-fidelity models in 2003. Forrester *et al.* [67, p. 197] demonstrate eighth-order *polynomial regression* to model drag coefficients on aerofoil design in 2008. Research continues in 2013, when Li *et al.* [131] investigated the nacelle/pylon (engine) position on wings with *RBFs*. Laurencau [121] compared advanced methods of *Kriging* effectiveness on aerodynamic shape optimisation, while Fincham and Friswell [63] successfully applied *RBFs* to wing morphology in 2015. Khurana *et al.* [106] investigated optimising aerofoil design using neural network surrogate models. Recently, in 2016, Palar *et al.* [172] compared a number of multi objective optimisation algorithms while using a second-order local surrogate model, which is updated at each point in the search space.

Helicopter rotor blades are another design problem commonly investigated using surrogate optimisation. Collins [41] applied fourth-order polynomial regression to rotor design, and Glaz *et al.* [76] used a *Kriging* method to investigate vibration dampening in helicopter blades. In addition Massaro *et al.* [145] used a neural network with an evolutionary algorithm to optimise efficiency at multiple objective altitudes and speed conditions in 2011.

Compressor design is another *CFD* problem that has benefitted from the application of surrogate models. Lian and Liou [134] used second-order polynomial regression and Kipouros *et al.* [110] applied *RBF* models in the context of multi objective optimisation in compressor design. Queipo *et al.* [185] applied polynomial surrogate models in the optimisation of the design of fuel injectors for NASA rockets in 2005. In 2008, Jaeggi *et al.* [94] compared *Kriging* with a related Gaussian process method called the *Sparse Pseudo-input Gaussian Process* (SPGP) finding that the SPGP outperformed *Kriging* on computation cost by a factor of four and produced better predictions in multi-fidelity models of diffuser design.

3.1.3 Meteorology and Physical Geography

Weather prediction has long been used for the development of models and surrogate models. Forrester *et al.* note that meteorologists often work with surrogate models:

> "The pressure [contour] chart is, of course, a surrogate model of the real pressure distribution, based on measurements at a network of weather stations"[67, p. 111]

Chorley [38] reviewed a number of surrogate methods in use in physical geography in 1965, including linear and polynomial regression, showing that this was already common practice at the time.

Razavi *et al.* [188] conducted and excellent review of surrogate methods related to water-resource modelling, covering polynomial regression, *RBFs*, and *Kriging* methods, and referencing 48 papers on surrogate methods in water resources and many more over physical geography. Their conclusion on the selection of methods is as follows:

> "It is not trivial to suggest the best function approximation technique for the purpose of response surface modeling, and metamodel developers typically pick a technique based on their preference and level of familiarity as well as software availability. Function approximation techniques that are able to (1) act as exact emulators, (2) provide a measure of approximation uncertainty, and (3) efficiently and effectively handle the size of the data set [...] of interest" [188]

3.1.4 Mineral Prospecting and Mining

Building a mine or an oil well is an expensive undertaking. Therefore, it is important in any geophysical survey to establish the minimum number of samples in order to make informed decisions. However, extracting a core sample (a sample is denoted as x in our description) is an activity that varies in cost – from the collection of surface samples (which have negligible cost)

to remote-area, *ultra-deep*, offshore core samples, which are collected from kilometres down, where the specialised drilling rigs alone cost many millions of USD per week [196], and there is significant danger to personnel [40].

For this reason, surrogate modelling methods have always been salient to geophysicists. Most models use *Geographic Information Systems (GIS)* map coordinates as the input, with a third dimension being the concentration of ore in samples. *Kriging* was invented by Danie Krige [116, unsubstantiated], a South African mining engineer, and its history is described by Cressie [44], who states that the term was coined by Matheron [146] in 1962.

As well as *Kriging*, other methods have also been applied to geological problems, notably a method developed by Singer and Kouda [199], who successfully applied neural networks to predict of mineral deposits in Japan. Research continues with further examples demonstrated by Porwal *et al.* [181] and Corsini *et al.* [43], who have both applied glsterms:rbf models to predict the location minerals and groundwater, respectively.

3.2 Surrogate Models in Nuclear Engineering

The complexity of in-core fuel management and fuel design has long been recognised as being highly nonlinear. The US Atomic Energy Commission [222, pp. 605–614], for example, discussed the nonlinear effects of control rods on adjacent fuel pins back in 1955. The in-core fuel management search space was investigated by Galperin [71] in 1995, who found the space to be extremely complex, with 10^{10} local optima in a search space with 10^{12} states. Despite the early recognition of the complexity of the problem, the label of surrogate modelling has not been used in the context of nuclear engineering until recently. There have, however, been a few attempts at using surrogate models. As early as 1993, Kim *et al.* [107] used shallow (single, hidden layer) *ANNs* to model PWR core parameters for a surrogate based optimisation of in-core fuel management. They concluded that the accuracy of their predictions was too low for application to *SMO*. They were achieving between 5% and 10% prediction error in networks that were a few hundred neurons in size. However, other works, such as Lysenko *et al* [137] used neural networks to control parts of the calculation in concert with analytical techniques and demonstrated utility. Another attempt was made by Faria and Pereira [60] in 2003, who again used smaller, shallow *ANNs* for core-reload pattern design and optimisation. This time, in-fill sampling was used to continually update the networks. They found that they were able to consistently design better systems than the reference design. They investigated the model using three different types of fuel preparation (MOX, AIROX and co-processing) but the experiment only considered the movement of fuel around the core rather than swapping fuel in and out. In 2008, Bae et al. [17] used *SVR*, kernel method is used to train *ANNs* for

prediction of *PPF* in fuel assemblies, concluding that the approach "sufficiently accurate for using in power peaking factor monitoring". However, no optimisation methodology or control strategy is considered.

Polyomial regression surrogate models such as the CESAR 5.3 code [224] have been successfully applied in mean cross-section prediction over burnup.

In recent years there has been an increasing trend of papers directly referencing *meta-models*, meta-modelling or surrogate modelling applied to nuclear engineering problems. In particular, Leniau *et al.* [126] used shallow *ANNs* to predict the plutonium content of MOX fuel, isotopic concentrations and cross-sections over burnup.

> "*MLPs* showed a very good accuracy since the average error on plutonium content determination is 0.37% with a standard deviation associated to the error of 1.55%." [126]

In 2018, Wu *et al.* [242] applied first-order *polynomial regression* and *Kriging* to the BISON nuclear fuel simulation code for modelling the release of fission gases. They found that "Kriging metamodels are applied to greatly alleviate the computer burden during [Markov Chain Monte Carlo] MCMC sampling." [242]

Despite the mixed results of previous studies using surrogate models to optimise in-core fuel management, and other aspects of nuclear engineering, it is clear that the problems can be tackled using this technique. Furthermore, progress during the past ten years means that there is the possibility that further successes can be achieved by applying modern machine learning techniques to the area of fuel design and in-core fuel management.

3.3 Deep Learning Surrogate Models

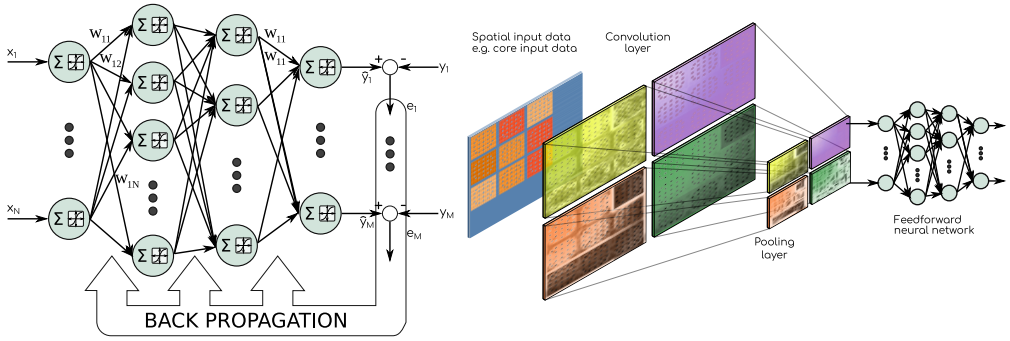
As mentioned previously, there is now a growing trend for the application of machine learning and particularly deep learning in the nuclear industry, based on methodological innovations such as *Convolutional Neural Networks (CNNs)* [123] and availability of tools mentioned in section 2.4.5. For example, at the PHYSOR 2020 conference there would have been eight papers presented by other authors on 'machine learning', or 'neural networks' [252, 133, 210, 65, 216, 141, 186, 205]. Compared to the previous PHYSOR [1], with no papers using these keywords in the title.

Of the PHYSOR 2020 papers, Zhang et al. [252] refer to the use of *CNNs*, in this case, it is investigated for simulation of flux and power distribution compared to the diffusion equation, with a view to extending the work to optimisation. However, unlike this thesis the optimisation step is not achieved.

3.3.1 Deep Learning Model Architecture

The first type of neural network surrogate model used in this thesis is an MLP type – a simple feedforward neural network commonly used for regression and classification tasks. As shown in Figure 3.1a, an array of processing nodes connect inputs, x, to outputs, y; each node sums the inputs and applies a transform to the data; and between nodes a weighting factor $w_{i,j}$ is applied. The weights, w, define an approximation of the required outputs, y. The backpropagation algorithm, made popular by Rummelhart *et al.* [191], is used to optimize the weights of the network.

The second type of deep learning surrogate model is called a *CNN* [122], represented diagrammatically in Figure 3.1b. This network configuration is well-suited to spatial data, such as recognition of images [99]. In this architecture, a population of filter kernels perform 'convolutions' on the input image. The resulting images have a sensitivity to the pattern of the kernel that gave rise to them. These images are usually put through a 'pooling layer' (scaled down), and these layers are repeated a number of times. The network finishes up with a deep feedforward neural network similar to an MLP. The system is successful because it does not require the designer to specify the base 'kernels' that will perform low-level image processing; they arise naturally from the convolution layers of the *CNN*. This type of network has proven extremely effective at recognising images, due to its translational invariance. Although translational invariance does not automatically occur in nuclear core design, by selecting inputs to represent the water gap and control/instrumentation pins, the effects of geometric variance are encoded into the inputs of the networks. The *CNN* represents an example of a typical advanced deep learning tool.



(a) A simplified diagram of an MLP. (b) A diagram of the CNN layer structure.

Fig. 3.1 Deep learning topologies [233].

3.4 PWR Fuel Design and Management

Table 3.1 shows a concise selection of the in-core fuel management literature in chronological order. The earliest research found by the author was conducted by Haling, who proposed an optimal strategy for the power distribution of a BWR [81]. Although the optimisation strategy is simply heuristic, the work recognises that the subject is complex and of interest to operators. Despite this early contribution, very little research was conducted for the next twenty years, despite significant contributions from Naft and Sesonike [159] and Ahn and Levine [7], because the computational power for detailed neutronics simulation and iterative optimisation was not readily available. As computer technology advanced in the 1980s, interest in approaches to iterative optimisation was revived. Development of the UK AGR programme coincided with the increased availability of computational power, which made iterative optimisation approaches possible (e.g. [173, 174]). Arrangement of fuel is inherently a combinatorial problem. Although iterative optimisation strategies cope well with combinatorial explosion, the search space is still extremely large and the number of iterations required makes them computationally expensive. The implications of this were investigated by Galperin [71], who concluded that since the computational power was not available at the time to conduct an iterative search, a heuristic approach was required to make the problem tractable. Thus, two broad approaches were developed largely in parallel: either to iterate over solutions measuring the objective or to simplify the problem to the point where the objective is relatively easy to find.

As computational power increased rapidly during the 1990s a large number of optimisation strategies have followed the stochastic optimisation approach. A selection of these papers includes research by Mahlers [139], Šmuc *et al.* [204], DeChaine and Feltus [51], Chapot *et al.* [32], and Francois and Lopez [70]. One of the main advantages of stochastic iterative optimisation is that it is possible when very little is known about the problem. Heuristic approaches were championed by Galperin and his students, and hybrid techniques have also been investigated, for example Stevens *et al.* [209].

Iterative optimisation does not guarantee the best solution (or non-dominated set), and is costly in terms of computational power. However it has proven extremely robust and is thus able to deliver acceptable results for a wide range of problems. The algorithms used – typically simulated annealing and evolutionary algorithms – have been successfully applied to a wide range of engineering problems [67] and can be applied when no mathematical analysis of the system is possible.

On the other hand, heuristic approaches must be carefully designed to correspond with the real world. An optimal solution for the heuristics does not guarantee a real-world solution and,

inevitably, simplification must take place in order to create a tractable set of heuristic rules, which may mean that the optimisation is happening on the wrong problem.

SMO aims to unite these approaches by generating a model from analysis of the data and automating the heuristic rule generation. It has the potential to be unbiased by human designers, and potentially avoids the heuristic approach's pitfall of optimising the wrong problem, while significantly reducing the computational load of the iterative optimisation algorithm. Furthermore, with the advent of parallel computation techniques, it may be possible to develop the model in parallel with the evaluation of the objective function, essentially running a novel optimisation based on the new data produced at each objective evaluation.

Although *SMO* is a relatively recent development in the nuclear field, a number of researchers have used *SMO*, without applying the label. These include Faria and Pereira [60] and Ortiz and Requena [171] who have, in the past, applied neural networks as surrogate models to the problem of in-core fuel management.

In recent years, the nuclear academic community has begun to consider *SMO* and deep learning strategies more seriously, overcoming initial scepticism. With the publication of review papers such as [162] and [77], we are seeing an acceptance of the significance of the innovations that have revolutionised subjects, such as natural language processing, image recognition and speech synthesis.

This chapter has discussed the trends and techniques that are used for in-core fuel management, without investigating the tools that are required for the implementation of such a system. The next chapter will introduce the tools that have been selected to develop an extensible framework for *SMO* of in-core fuel management problems.

Table 3.1 Selected literature on in-core fuel management. Abbreviations used in this table: ACS=Ant Colony System, BE=Binary Exchange, GA=Genetic Algorithm, GPT=General Perturbation Theory, HR=Heuristic Rules LP=Linear Programming, NN=Neural Network, NC=Nodel Code, SA=Simulated Annealing

First Author	Reactor	Algorithm	Evaluation	Software	Yr.	Ref.
Haling	BWR	HR	1 group model	-	1963	[81]
Fenech	-	-	-	-	1970	-[a]
Naft	PWR	Direct Search	Semi-analytic	JUMBO	1971	[159]
Ahn	PWR	-	-	-	1985	[7]
Parks	AGR	Metropolis/ SA	Custom code	AMETROP	1989	[174]
Poon	-	SA, GA	GPT	FORMOSA	1993	[180]
Mahlers	PWR	SA / LP	Green's method	-	1994	[139]
Kropaczek	PWR	SA	GPT	FORMOSA	1995	[118]

First Author	Reactor	Algorithm	Evaluation	Software	Yr.	Ref.
Šmuc	PWR	SA	1½D Diffusion Eq.	MOCALPS	1995	[204]
Stevens	PWR	SA	NC	SIMAN	1995	[209]
DeChaine	PWR/ BWR	GA	NC	CIGARO	1995	[51]
Parks	PWR	GA	GPT, NC	FORMOSA, PANTHER	1996	[175]
Galperin	PWR	Tree search + HR	-	-	1997	[71]
Yamamoto	PWR	GA, SA, GA+BE	diffusion	-	1997	[247]
Chapot	PWR	GA	NC	GENESIS	1999	[32]
François	BWR	GA	NC	PRESTO-B	1999	[70]
Faria	PWR	NN	NC	WIMS, CITATION	2003	[60]
Ortiz	BWR	GA+NN	-	CM-PRESTO	2004	[171]
Martín-del-Campo	BWR	GA	-	CM-PRESTO	2004	[144]
De Lima	PWR	ACS	-	RECNOD	2008	[49]
Esquivel-Estrada	BWR	ACS		CM-PRESTO	2011	[58]
Hill	PWR	tabu, SA, GA	GPT	FORMOSA-P	2015	[84]

[a] Found in [159] 1972 and others (original article by Fenech "Application of Optimization Methods to Nuclear Power Plant Design and Optimization, Trans. Am. Nucl. Soc., 13, 90 (1970)" not located).

Chapter 4

Framework and Methodology for Surrogate Model Optimisation

In the last three chapters, the concept of *PWR* in-core fuel management has been introduced, as well as the concept of iterative *MO* and the rationale for *SMO*. In this chapter, the tools used by the author will be described as well as the software architecture used to carry out experiments.

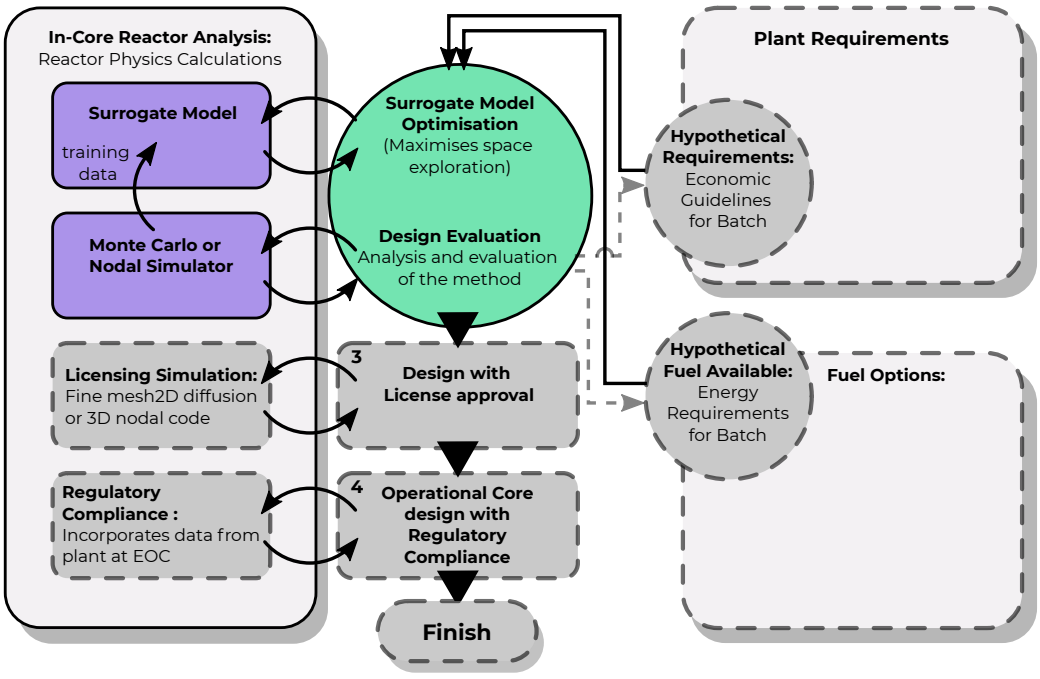


Fig. 4.1 The simplified in-core fuel design process used in this thesis (*c.f.* Utility organisational structure, Figure 1.1)

This thesis proposes and discusses two advanced surrogate models which have not been previously applied to nuclear engineering, including transferring a novel optimisation approach to the nuclear field. During the investigation, experiments often informed what steps would subsequently be taken. So, rather than describing the step-by-step experiments without results, this chapter will describe the software architecture, the tools used, and the framework that has been developed for the experiments. While sequential descriptions of the experiments, informed by experimental results, are presented in the following chapters, this chapter describes the discrete building blocks of the environment where experiments can take place.

4.1 A Simplified Structure for SMO Fuel Loading Optimisation

It is not possible to develop a model of the full interactions of the working groups of an organisation employing hundreds of people in a single PhD thesis, so a simplified subset of the interactions as a simulated environment is created. This allows the in-core fuel management experiments to be carried out. Figure 4.1 shows the implemented activities in colours, while the groups or activities that are substituted by requirements are shown in grey.

For the purposes of this thesis, a software framework has been developed to allow rapid experimentation with a number of requirements, models and optimisation algorithms.

4.2 Software Architecture

In order to rapidly develop advanced systems, the author adopted software best practices, such as unit testing, version control and modular design. In addition, in order to enable good scientific practices, such as reproducibility and further development an open data approach has been adopted. Source code and data are made available with publications and this thesis under a permissive open source license for the benefit of the scientific community and interested parties.

Furthermore, by utilising the open source Application Programming Interfaces (APIs), the author has been able to ensure the quality of implementations of standard algorithms like NSGA2 [50], convolution and deep learning architectures. These have been selected from a peer-reviewed, portable, implementations that are actively tested by a community of users and developers. Other authors can also download and verify the results presented in this thesis – a concept called “standing on the shoulders of giants” after the famous Newton quote. A graphical representation of the software architecture is shown in Figure 4.2. Three major

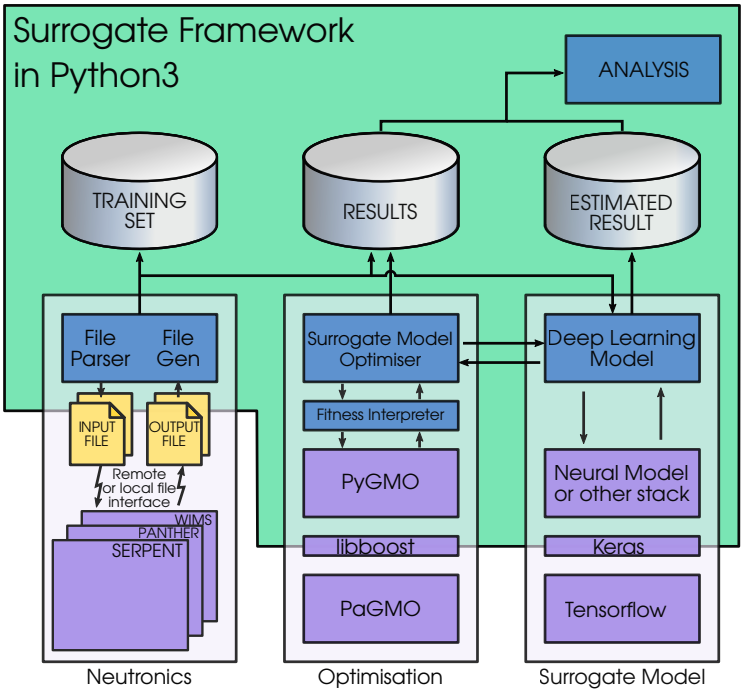


Fig. 4.2 Software architecture diagram, showing software components, and the flow of data within a re-useable framework for experiments

external software solutions have been leveraged, namely neutronics, optimisation and surrogate models. By building clear interfaces to these software libraries, it has been possible to swap components in and out for use in different experiments in this thesis.

It has been shown that open source software is able to respond to security threats more quickly than proprietary software [194], and that opening the source of a project can lead to improved modularity in the architecture [138]. With this in mind, the choice of optimisation and deep learning libraries has been influenced by a preference for permissively open-source licenses.

With an explosion of excellent software tools in recent years, there are often multiple tools available to solve a problem. The author does not claim that the chosen tools are necessarily superior to achieve the required computing techniques; instead the following sections describe the chosen software tools, with reasons for choosing or not choosing them where reasons exist. The focus in the selection is to ensure that the chosen API *was suitable for the task* rather than to try to claim that it is *the best* tool. Alternative options are listed in Table 4.6, at the end of this chapter.

4.2.1 Python

Python is the default programming language used in this thesis, it is an open-source programming language developed by Van Rossum [223] that is commonly used in scientific computing

Choosing a programming language that is 'in vogue' is a justifiable design choice as it allows the development to benefit from modern API releases in areas like deep learning, quantum computing and optimisation with up-to-date community support for libraries, and also so that other researchers can download and test claims made in this thesis and supporting publications with relative ease. As of February 2020, Python is the third most popular programming language according to the TIOBE index for programming language popularity [218].

4.2.2 C++

C++17 is a standardized programming language that evolved from an object-oriented version of ANSI C. ISO accreditation means that this is one of the major programming languages used in software development today [90].

Like ANSI C, C++ is a high level language, with access to low level features. This means that there are structures like objects, vectors and linked lists natively implemented in the language, but also pointers and 'direct' (managed) memory access.

These features make C++ a good choice for very computationally efficient programming. In addition, PaGMO2 had been implemented in C++. The original plan for this thesis did not involve C++ programming and merely used the PyGMO API as an interface to the PaGMO2 library. However, when certain features were required by the author (see 4.3.1), using this programming language became a necessary. As of February 2020, C++ is the fourth most popular programming language according to the TIOBE index for programming language popularity [218].

4.2.3 Simulation of the Neutron Transport Equation

A simulation is required to model the neutron population in a reactor in terms of the geometry, materials and energy spectrum.

At steady state, in a critical reactor, the neutron transport equation is a balance equation describing the flux of neutrons, in terms of gains and losses. For a multiplicative medium with no external source, one form of the neutron transport can be described based using the following nomenclature:

E is the neutron Energy (MeV)

$\vec{r}$	is the cartesian position vector
$\hat{\Omega}$	is the direction vector (radians)
Σ_T	the absorption cross-section (barns)
Σ_S	The scattering cross-section (barns)
Σ_F	The fission cross-section (barns)
Ψ	is the angular neutron flux ($n/(cm^2 sradian)$)
ν	mean number of neutrons per fission event
K	the effective multiplication factor
χ	fission emission spectrum

by a 'loss operator', described by Carney et al. [30] as :

$$\mathbf{M}\cdot\Psi(\vec{r},E,\hat{\Omega}) = \underbrace{\hat{\Omega}\cdot\nabla\Psi(\vec{r},E,\hat{\Omega})}_{leakage} + \underbrace{\Sigma_T(\vec{r},E)\Psi(\vec{r},E,\hat{\Omega})}_{absorption} - \underbrace{\int_0^{\infty}\int_0^{4\pi} dE'd\hat{\Omega}'\Sigma_S(\vec{r},E'\rightarrow E,\hat{\Omega}'\rightarrow\hat{\Omega})\Psi(\vec{r},E',\hat{\Omega}')}_{scattering} \tag{4.1}$$

and a 'source term':

$$S(\vec{r}) = \underbrace{\int_0^{\infty}\int_0^{4\pi} dE'd\hat{\Omega}'\nu\Sigma_F(\vec{r},E')\Psi(\vec{r},E',\hat{\Omega}')}_{fission} \tag{4.2}$$

This allows the neutron transport equation to be expressed as:

$$\mathbf{M}\cdot\Psi(\vec{r},E,\hat{\Omega}) = \frac{1}{K}\frac{\chi(E)}{4\pi}S(\vec{r}) \tag{4.3}$$

Complexity arises due to cross-sections being a function of energy of neutrons and individual to each isotope. In any medium where fission occurs, a wide range of fission products will build up and decay so the behaviour of the system also depends on the neutron population history.

There are two broad approaches to solve this equation for neutronic systems. Firstly, deterministic approaches; where Equation 4.3 is solved explicitly. Secondly, a stochastic approach called the Monte Carlo method.

Of the deterministic approaches, two methods are relevant to this thesis: *Method of Characteristics (MOC)* such as WIMS[135] and nodal methods such as PANTHER [156]. *MOC* solutions, are characterised by the transport equation being solved along straight lines through the geometry in a finite number of directions. Whereas nodal methods consider the transport at a discretised mesh. Nodal methods require homogenised cross-sections typically gener-

ated by other methods such as *MOC*. A good reference for the interested reader on nodal implementation is found in Smith [201].

Monte Carlo simulation of the neutron transport can be carried out by software packages such as Serpent [128]. In this approach a probabilistic simulation is made of a set of simulated neutron histories from the point of birth, migration through a modelled geometry to the point of absorption or leakage. The behaviour of a population of simulated neutrons gives an approximation of the behaviour of the (many orders of magnitude larger) population of neutrons in a similar geometry. Thus a Monte Carlo simulation can be made to generate very accurate results for arbitrary geometries, albeit at a considerable comuptational cost.

Two methods of solving neutronics problems have been used in this thesis. In Chapters 5 and 6 the Serpent software [128] has been used to evaluate the power level and reactivity of designs proposed by the optimisation algorithms, while in Chapter 7 the PANTHER nodal code [156] has been used in conjunction with outputs from the WIMS [135], to generate lattice physics solutions to neutronics problems.

Contemporary academic texts usually attempt to carry out nuclear optimisation problems using deterministic codes, such as nodal codes (see table 3.1) or *MOC*. WIMS software [135], an *MOC* solver, was applied as the evaluator in a number of problems during several stages of the PhD that have not been presented. Homogenised cross-sections from WIMS are used in the PANTHER simulations in Chapter 7, due to organisational issues with the management of jobs on the shared computer resources, it was easier to simply redevelop the code to run on nodes of the 'skylake' Cambridge High Performance Computer (HPC). At the point where massive parallelisation is available, a modern code that is designed to be multi-processor aware offered significant advantages. This led to investigations using Serpent, which has a modern, more easily machine-readable input and output file format, which lends itself to algorithmic optimisation experiments.

Serpent is a Monte Carlo neutron transport solver with burnup capabilities developed by VTT in Sweden [128]. The Monte Carlo method generates a non-deterministic solution, which means that there is stochastic noise on the output values. Although evolutionary algorithms are robust to stochastic noise, it is a problem for *MO*, where, close to the *NDF*, the stochastic noise contributes significantly to error. For this reason, *MO* methods require special treatment in order to operate on stochastic objective functions. A number of solutions are compared and discussed by [29], including the noise-hardened NSGA2 algorithm. These methods compensate for the noise but multiply the computational cost. The author's approach to this has been to avoid the problem entirely by setting the seed of the Monte Carlo simulations *per run* of NSGA2 algorithm. This means that for each run of the NSGA2, the system is optimising a deterministic system that approximates the neutron transport solution with some estimable disturbance.

In Chapter 7 a standard PWR core with 193 fuel assemblies was investigated, similar to the one shown in Figure 1.2a. The computational expense of the Monte Carlo simulations increases with the area of the simulation, so a fast 'nodal' code was used. Here, results from the *MOC* Solver in WIMS [135] were used as building blocks to simulate cores, from an eighth core with diagonal and orthogonal symmetry in PANTHER [156]. This was done for validation of the heuristically evolved loading patterns.

4.2.4 Deep Learning Frameworks

A number of deep learning libraries have been released in the past ten years, significantly improving the situation for researchers, and in turn helping to maintain the momentum in this field of research. At the start of this project (when the decision about software architectural choices was made), it was unclear which APIs would be successful. The author has been fortunate to have chosen libraries that continue to be actively supported, and developed. The libraries chosen are Keras [37], which acts as an abstraction layer from the Tensorflow [6] library which manages intensive calculations efficiently. Table 4.6 shows a list of other options which were consdered.

Keras

Keras is a high-level deep-learning library which uses Tensorflow or Theano [20] as a backend and allows efficient development of standard neural network architectures. [37], alternative high level deep learning packages include PyTorch [177] and Café [96] – which has now evolved into the Café2 project that will merge into pyTorch.

Tensorflow

Tensorflow is a library developed by Google, which is capable fo performing memory-efficient tensor operations [6]. A rival open-source library called Theano [20] was in development when software design was being carried out. However, it was retired in 2019, with the authors agreeing that Tensorflow should become the *de facto* standard.

4.2.5 *MLP*

A simple deep MLP is used to predict the PPF and horizontal and vertical positions of the hottest pin. Using the w/o U235 on a per-assembly level as the inputs, this means that the input dimension for a quarter core is nine, while the output dimension is three (the hottest pin coordinates and PPF). In order to select an appropriate topology for the MLP, a short

Table 4.1 Comparison of optimisation frameworks

Language	Lanuage/Interfaces	Comments
PISA	C++/files	Optimisation library with many working optimisation algorithms. No longer developed by the authors.
JMetal	Java/JNI-	Actively developed API with many implemented optimisation algorithms. Many authors contribute their own optimisation algorithms
PaGMO/ PyGMO	C++14/ BOOST/ Python	Actively developed library for optimisation.

study of network topology was carried out, which examined the numbers of hidden layers and neurons per hidden layer. The results of the study are shown in Figure 5.2. The choice to keep the number of neurons per layer constant across the network was made in order to reduce complexity and keep the design tractable on a two-dimensional heatmap. The performance seen in Figure 5.2 confirms that MLP networks are robust to a wide variety of topological values. This has been discussed in the literature by many authors (see [24, p. 130] and more recent deep learning interest is sometimes regarded as being set in motion by Le Cun [123]). A fairly large network with nine layers of 80 neurons was chosen for these experiments. The weights are modified using backpropagation and the 'Adam' algorithm [109], and the loss function used was the *MAE*. The network was trained for 500 epochs. In each training epoch, the system trained on a set of 50 randomly selected samples from the training set, N. This helps to reduce overfitting of the network to the training set compared to training on the whole set.

4.2.6 *CNN*

The topology of the CNN used is shown in Table 5.2 in Section 5.2.5. It was chosen based on reasonable compromises for memory limits and the number of training examples available and with reference to reference CNN topologies such as LeNet-5 [124], AlexNet [117], or the VGG series of networks [198] (current versions are VGG-16 and VGG-19). Three convolution and pooling layers are followed by six layers of fully connected feedforward layers. As with the MLP, 'Adam' was used to modify weights based on a MAE loss function. The training was carried out for 100 epochs with a sample size of 50. Unlike in the MLP surrogate model, the *CNN* directly predicts the per-pin powers. The objective functions are then calculated from the

surrogate outputs by the usual means. These values were chosen based on comparison with existing example networks from the literature [198, 211, 87] and experimentation to fit the input image size and output data.

4.2.7 Optimisation

Table 4.1 shows a comparison of different APIs for optimisation. This thesis uses Parallel Global Multiobjective Optimizer (PaGMO). PaGMO is a computationally efficient optimisation package available under the MIT license. The complete Python interface, called and PyGMO Global Multiobjective Optimizer (PyGMO), operates computationally intensive tasks in C++11 through the LIBBOOST interface.

The PyGMO documentation describes it as:

> "a scientific Python library for massively parallel optimization. It is built around the idea of providing a unified interface to optimization algorithms and problems, and to make their deployment in massively parallel environments easy.
>
> Efficient implementantions [sic] of bio-inspired and evolutionary algorithms are sided to state-of-the-art optimiation algorithms (Simplex Methods, SQP methods, interior points methods, ...) and can be easily mixed (also with your newly-invented algorithms) to build a super-algorithm exploiting algorithmic cooperation via the asynchronous, generalized island model." [47]

Features of PyGMO that have been ustilised include:

- Non-dominated Sorting Genetic Algorithm 2 (`pygmo.nsga2`)
- Exact hypervolume algorithm for two dimensional points (`pygmo.hv2d`).
- Fast Non-dominated Sorting (`pygmo.fast_non_dominated_sorting`)
- Pareto dominance (`pygmo.pareto_dominance`)
- Population selection utilities (`pygmo.sort_population_mo, pygmo.select_best_N_mo, pygmo.decompose_objectives`)

4.2.8 Approaches to Parallelism in Optimisation

Due to the high economic cost of *DSO*, many researchers have investigated ways to run an iterative optimisation in parallel. For algorithms where most of the computational expense is

in fitness evaluation, evaluating more than one individual's fitness at the same time bestows significant advantages.

Table 4.2 describes how a number of APIs have approached optimisation parallelism. The different approaces are described below.

Algorithmic Parallelism

Algorithmic parallelism refers to the separation of the algorithm into separate processes, that can then be run at the same time. This parallel operation is either spread across compute nodes on a cluster of computers, on different cores in the same machine or as processes running on a single processor and given alternate time slices by the operating system.

This approach to parallelism is usually more involved since the programmer must decide how communication between different processes occurs. Since it was not possible for the author to complete projects without parallel operation, algorithmic parallelism had to be implemented instead. The approach used is described in Section 4.3.1.

Island and Archipelago Parallelism

The approach to parallelism that is commonly used in optimisation is termed 'Island and Archipelago'. It is a simple approach to the parallelisation of tasks that relies on the expectation that many runs of an algorithm are required. The approach splits separate instances of the algorithm to run on different processors – for example, the PyGMO implementation is described by Izzo [93]. This is ideal for running instances of problems that can be distributed to different nodes in a cluster. However, if the objective function requires large amounts of computational budget, then Island and Archipelago parallelism gives limited benefits.

4.2.9 Dimod

Dimod [42] is a hardware-neutral software library for the implementation of quantum annealing optimisation problems in Python. For Chapter 7 the dimod library is used. The choice of this library was based on the hardware neutrality, open source license and having the highest number of features in Table 2 of [64].

However, it is worth noting that, at this time, the only implemented backend for the Dimod software is the developer's hardware, the D-Wave Quantum Annealer and the D-Wave Simulated Quantum Annealer (a software implementation of the company's product).

4.3 Parallel and Serial Optimisation Algorithms

In order to complete this project within a realistic timeframe, the algorithms in the PyGMO library had to be made to operate in parallel. Although PyGMO was selected partly due to its support for parallel operation modes, it turns out that the most important method of parallelism for surrogate model work is algorithmic parallelism. This is not possible with every type of optimisation algorithm; algorithms with a concept of 'neighbourhood' usually require the fitness of every individual to be calculated before each offspring is generated (e.g. MOEA/D [251]). This meant that of the implemented multiobjective algorithms available in the framework, NSGA2 [50], remained the only feasible option, with other implementations such as NSPSO [132] (a Nondominated Sorting extension of Partical Swarm Optimisation by [55]) being actively developed by the community of PyGMO developers.

4.3.1 NSGA-2 Algorithm Modified for PaGMO2/PyGMO2

In order to use algorithmic parallelism, modifications to the PyGMO library were required to the implementation of the NSGA2 algorithm and supporting interfaces. This was initially carried out as a branch of the original PyGMO library. After a satisfactory solution was agreed with the community, a 'pull request' was created for this implementation. The parallel implementation that was developed met the vision of the PaGMO/PyGMO developers.

Description of changes made for parallel operation

In order to optimise efficiently in parallel, an additional wrapper to the 'fitness' function has been developed by the author. Although this allows parallel fitness evaluations, the python3, BOOST interface and C++ library had to be modified.

The initial implementation split fitness evaluations into two functions: a `'start fitness'` and a `'wait_collect_fitness'`; the `'start_fitness'` function returns an instance hash that the (blocking) `'wait_collect_fitness'` uses to find the results. In the *MO*, all of the individuals in a generation can have their evaluation initiated before any of the results are required. The net gain of this parallelism is close to $1/population\_size$ due to the high cost of evaluation of nuclear simulations, when compared to *MO*.

The second implementation was based on suggestions from the PaGMO/PyGMO community as well as parallel development from them. This method adds an optional 'batch fitness' implementation and other flags to the algorithm. A properly developed parallel algorithm can then call the 'batch fitness' function. This is superior to the simple implementation above as it allows many function evaluations to be made, and a single function can distribute the calls

across a cluster or across another distributed computing environment. After the first parallel implementation was completed, the community had implemented the preferred interface, but no algorithm in the PyGMO library actually used the parallel functionality. So, the author then implemented a parallel NSGA2 algorithm, including testing and debugging and provided a pull request to the community maintainers.

Quantitative evaluation of equivalence over a range of problems

The PaGMO/PyGMO library has unit-testing functions as well as continuous integration implementations. In order for the code to be accepted, unit test compliance must be observed in the integration test shown in Table 4.3.

The pull request for the parallel branch developed by the author was merged into the main branch of Pagmo in March 2019.

4.4 A 6×6 Microcore with Rotational Symmetry

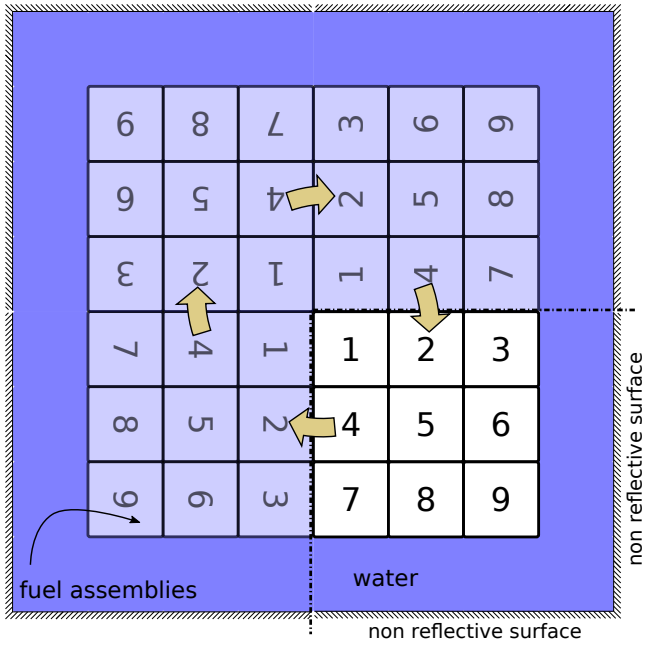


Fig. 4.3 A small microcore, created as a Serpent simulation for the purposes of investigating the optimisation of small cores.

The initial experiments with *SMO* will use a tractable problem: a microcore composed of 6×6 standard PWR assemblies. The layout described here is shown diagrammatically in Figure 4.3. The core is formed of 36 PWR 'standard' fuel assemblies [229] arranged in a square. A fourth-order rotational symmetry is used, to simplify the design. This allows limited checker-boarding of the design while keeping the number of variables relatively low. In initial experiments, the assembly enrichment is varied on a per- assembly basis. So, there are only nine input variables (the U235 enrichment of each assembly), and each variable is able to take 22 values of enrichment between 0.8 and 5 w.t.%. This creates a discrete input space with 1,207,269,217,792 possible solutions.

Simulation in Serpent

The *DSO* simulations in Serpent use parameters as shown in Table 4.4. The detectors used in the simulations to estimate pin powers are collecting the 'total fission energy deposition' in MeV.

Table 4.4 Serpent Simulation Parameters for BOL and Burnup simulations and for training the fission matrix

	BOC	Burnup	Fission matrix
Source neutrons per cycle	20000	20000	800000
Active cycles per timestep	500	500	500
Inactive cycles per timestep	100	100	100
Fuel Temperature (K)	900	900	900
Coolant Temperature (K)	600	600	600

Burnup simulation in Serpent

The experiments in this section use Serpent's stand-alone burnup calculation. The power density is set to 40 kW/kgU, the fuel is depleted using 20 cumulative depletion steps, based on the example burnup code for a PWR assembly [127]. The burnup steps are at 0.1, 0.5, increments of 1 from 1-10, increments of 2.5 from 12.5-20, and increments of 5 from 25 to 40 MWd/kgU. From the Serpent simulation, the cycle length is estimated by taking a linear interpolation in terms of k_{eff} and burnup at the step where k_{eff} goes below unity.

Multiobjective Objectives

Although in-core reactor analysis groups at a utility company will optimise designs over many objectives simultaneously, it is instructive to consider two variables at a time. This is since the $(N-1)D$ 'surfaces' in N dimensional spaces take up relatively more space as N increases [115], so the '*NDF* to hypervolume' becomes larger making realistic consideration of the search space more difficult.

The startup experiments look at *PPF* vs the position of the hottest pin, while burnup experiments investigate the variance of the pin powers (an analogue for *PPF*) and *cycle length*, a selection of objective variables are presented in Table 4.5.

Table 4.5 Objective variables considered

Objective /units	**Description**	**Attributes**
PPF	Power Peaking Factor is a common measurement of the evenness of the temperature of the coolant as it leaves the reactor. By lowering *PPF*, more power is delivered to the turbines for a fixed maximum temperature inside the reactor.	Error is a function of the hottest pin
Radial distance of hottest pin	Moving the hottest pin to the outside, while reducing the *PPF*, the core designer can create a design that has a flatter overall power profile.	
Mean *enrichment* /w/o U235	Enrichment of the isotope U235 from natural uranium, is a major cost centre in fuel manufacture. The cost of enrichment is exponential with w/o U235, so modest reduction of enrichment gives significantly improved fuel cost.	Trivial to calculate from the inputs
EOC Burnup $.(-1)$ $/MWd/kgU$	The burnup achieved before $k_{eff} < 1.0$, A reactor that delivers a controlled amount of power for longer, delivers more total power per refuel and is offline less of the time relative to a shorter cycle. Optimisation considers minimisation problems by convention, so the number is multiplied by -1.	Reactors are refuelled at planned outages, so a longer possible cycle may not benefit the utility company as much as expected.
Pin Power Variance (PPV)	An alternative measure of the evenness of the temperature of the outlet coolant would be the variance of the pin powers	Less sensitive to a stochastic error on pins, for example with Monte Carlo simulations.

Selection of some pairs of objective variables from the table was found to generate single optimal results, for example lowering PPF and reducing mean *enrichment* unexpectedly created

a unique solution. On analysis, this turns out to be because the parameters of the simulation allowed enrichment that generates $k_{eff} < 1$, so at near zero powers, a single lowest solution for *PPF* can be seen.

Pairs of variables that generate interesting *NDFs* have been the focus of this study, as it is of interest to know how the surrogate models perform across the *NDF*.

Criticisms of the Problem

The author acknowledges that the limitations of the design will lead to reduced optimality in the found solutions. Full *checkerboarding* is not possible at the chosen fourth-order rotational symmetry. However, the aim of this exercise was to create a viable minimal core design that allows for optimisation experiments without becoming overly complex. This microcore design and optimisation scheme has the advantage that it has not, to the best of the author's knowledge, been investigated in the literature. One problem when comparing *SMO* (or any other automatically generated) results with expert human designers is that there is a large corpus of literature that already describes near-optimal design solutions for existing cores, which are known to most human experts (e.g. examples include [174, 13, 180, 175]).

4.5 Summary

A wide variety of cutting-edge computing libraries and techniques have been applied in this thesis. Where the tools were unable to deliver a suitable solution, for example, in the case of parallel operation of the optimisation algorithm, modifications to the tools were developed. This could be achieved by developing code around APIs, as was possible for the *SMO*, or by developing solutions within the APIs reviewed by peers. Modifications to open-source tools made by the author have been accepted by the community after following software development best practices.

Table 4.6 shows a summary of the APIs used to generate this thesis. Although it is vital for the author to share the methodology used, it is also possible to show the working of software by sharing the source, as has been adopted in this thesis.

The next three chapters will describe, in a roughly linear fashion, experiments using a number of different surrogate model techniques developed using the software framework described in this chapter in order to show when and how *SMO* can be applied to problems associated with in-core fuel management in a PWR.

Table 4.2 Comparison of approaches to parallelism in optimisation frameworks

Type of Parallelism	Implementation Level[a]			Comments
	Algorithmic	Island	Archipelago	
PISA	–	–	–	File based interfaces allows user-defined parallelism
JMetal	P	×	–	Parallel implementation of problem/algorithm on multiple independent populations
PyGMO/PaGMO	I	×	×	Support for high performance computers and clusters running multiple islands

[a] × = Implemented, P = Partially Implemented, I = Implementation by the author, – = Not Implemented

Table 4.3 Continuous integration testing frameworks

Testing Provider	Result	Website
Circle CI	success	`https://circleci.com`
Travis CI	passing	`https://travis-ci.org`
AppVeyor	passing	`https://ci.appveyor.com`

Table 4.6 Summary of software component choices

Component	Chosen	Alternatives	Comments
Optimisation	PyGMO	JMetal, PISA, scipy.optimization, matlab	Chose python API due to fit with overall language, better multiobjective support than scipy. [47]
MO Algorithm	NSGA2	Simulated Annealing, evolutionary algorithms, heuristic methods ... *and many more*	NSGA2 is a well researched algorithm, which is extensively used to benchmark algorithms [120, 130, 50].
Quantum Annealing	dimod / Ocean	Qiskit, forest SDK (IBM)	The dimod interface is an open source hardware neutral environment. However, the Ocean inc. API is the only implemented interface for quantum annealing at this time. [42]
Deep Learning	Keras / Tensorflow	Cafe, Theano, PyTorch, effective quadratures	Keras operating on Tensorflow gives good abstraction for deep learning architectures with low level access deep learning architecture. [37, 6]
Neutronics	Serpent[1], PANTHER[2]	WIMS, OpenMOC, OpenMC	Serpent is a Monte Carlo neutronic solver capable of solving for arbitrary geometries [128]. PANTHER is a nodal code that is used for full core PWR evaluation and used due to the existing BEAVRS benchmark [82].

[1] Used in Chapters 5 and 6 [2] Used in Chapter 7

Chapter 5

Deep Learning Surrogate Models

5.1 Introduction

This chapter examines two deep learning regression models used with iterative optimisation to create an *SMO* process. These deep learning surrogate models are evaluated for a 'toy' problem of optimising the design of a 'microcore' simulation, which is constructed from 36 'standard' *PWR* fuel assemblies as described in Section 4.4. The design is considered with order four rotational symmetry, reducing the problem to nine assemblies (Fig. 4.3). Although there are legitimate limitations of this core design for applications in the real world, it is ideal for investigation and optimisation strategy evaluation. The designs are relatively easy to simulate, the design space is bounded, and results can be interpreted by human experts.[1] The final experiment shows how this technique can be easily extended to a geometry equivalent to the Nuscale *SMR*.

The software design philosophy adopted is to use standardised state-of-the-art implementations of recognised software techniques, and open source libraries where possible. In particular, recent years have seen the release of the advanced machine learning libraries tensorflow [6] and Keras [37] and the optimisation library PyGMO [92]. Therefore this philosophy allows for cumulative development of new knowledge that uses the state-of-the-art, and can be replicated as researchers have access to the underlying source code.

The surrogate models used to predict the core parameters are deep Multi Layer Perceptrons (MLPs), and Convolutional Neural Networks (CNNs). The objective parameters considered are *PPF* and the position of the hottest pin at the Beginning of Life (BOL), *PPF* and the average

[1] Results in Section 5.4 have been peer reviewed and accepted for the PHYSOR 2020 conference [235]. They are presented here in Section 5.4, and text is used in this chapter and Section 4.4. It is reproduced here with some modifications to make the subject more tractable in the context of the thesis and to extend work to incorporate the use of Sobol sequences as training data.

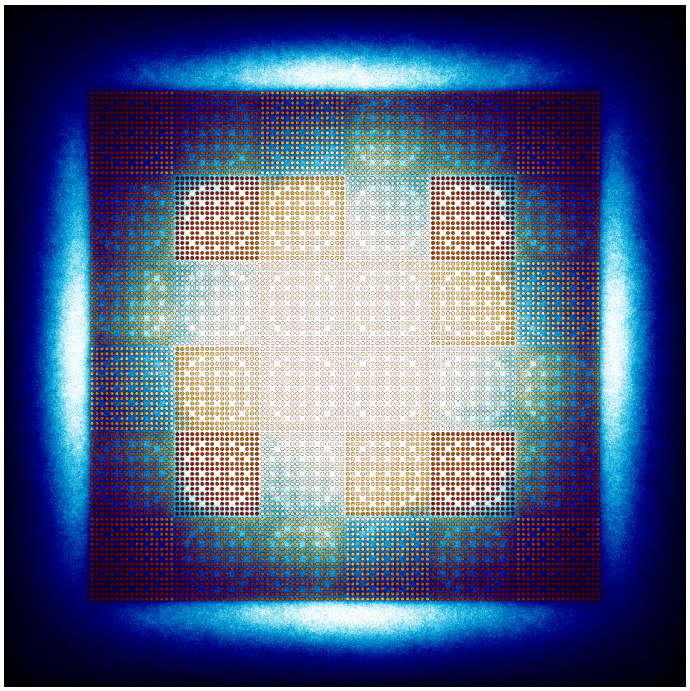


Fig. 5.1 An example flux map for the microcore as described in Section 4.4

enrichment, as well as *PPF* and cycle length; these are outputs in the *MLP* model and derived from pin power predictions in the *CNN*. The surrogate models are developed using a number of tests to establish the best design parameters.

The derived deep learning models are then used as surrogate models with a benchmark iterative optimisation algorithm: the Non-dominated Sorting Genetic Algorithm (NSGA2) [50]. The speed of the deep learning models enables them to evaluate thousands of solutions per second. This means that the optimisation algorithm could be given larger populations with minimal impact on the computational budget or that more exploratory settings can be chosen, while using only a fraction of the computational time that a more traditional *DSO* approach would require. However, in this study, the performance of most results are compared using like-for-like settings of the optimisation algorithm.

5.2 Surrogate Model Optimisation

SMO is becoming popular in the field of nuclear engineering [242]. The method is advised for problems where the objective function is computationally expensive [67, 148] and works by applying a standard optimisation algorithm on a ‘surrogate function’, which is usually

regressed from data points obtained by sampling the original objective function. The significant components used in this chapter are described below.

5.2.1 Optimisation Tasks

The *DSO* and training simulations use the Serpent Monte Carlo code [128], to simulate a 'toy' example of a fuel arrangement problem. This enables an exploration of the *SMO* approach to fuel management problems. As described in Section 4.4, nine assemblies are arranged into a square lattice, quadrant rotational symmetry is applied on two sides, and they are surrounded by water at an equal thickness to the assemblies followed by non-reflective (black) boundaries (Fig. 4.3). The assemblies, labelled 1–9, are standard *PWR* type assemblies separated by a small water gap. Each assembly has an enrichment value that can be varied independently, with quantised enrichment values at increments of 0.2 between 0.8 and 5.0 w/o U235. Monte Carlo simulations were run for uniformly random uranium enrichments in each of the nine assemblies; these form an initial set of training simulations used to train surrogate models.

A number of objectives are investigated. *PPF* measures the uniformity of the power generated [208, p73] and is an indicator of safe power that can be evolved within the allowed *Departure from Nucleate Boiling Ratio (DNBR)*. The position of the hottest pin is an indicator relevant to the shape of the power profile, controlling and monitoring the axial position of the hottest pin is of interest operators ensuring that *DNBR* is not exceeded [158]. Neural networks have been used previously to predict similar scenarios [158, 108]. Average enrichment of the pins is also used as an objective since this has a significant impact on the cost of the fuel, while extending burnup can be used to directly extend the time that the reactor operates, or to deliver a greater degree of controllability if a plant is operating on a fixed batch length.

5.2.2 Optimisation Algorithm

NSGA2 is considered a 'solid multi-objective algorithm'. It is widely used in many real-world applications and is easily parallelised [47]. The parameter values used in this chapter are shown in Table 5.1 and are used for both *SMO* and *DSO* unless otherwise stated. Although the number of generations, N, and population size, P, are generally smaller than used in other studies, such as [50] ($P = 100$), [175] ($N = 51, P = 204$) and [130] ($P = 300$). These parameters were found to perform well and are chosen to represent the *DSO in the best possible light*, despite its very high computational cost. The choice of parameters is based on the study in Appendix B, where the parameters are considered in detail. NSGA2 is widely used as a benchmark algorithm when comparing novel algorithms or techniques. The implementation used in these experiments is from the software library PyGMO [92], with modifications as detailed in Section 4.3.1.

Table 5.1 Parameters used in NSGA2

Parameter	Value
Population size, P	50
Generations, N	60
Crossover probability	0.95
Mutation probability	0.01

5.2.3 Training Sets

Initially, the correct size of training set was unknown, so two large training sets were generated. Firstly, a uniform random training set was generated, using the Mersenne twister algorithm [149]. In the random training set, although each input (the enrichment of an assembly in w/o U235) is uniformly distributed in discrete values of 0.2 from 0.8w/o U235 to 5w/o U235, the mean value of the ensemble tends to be normally distributed around tid-range value by *central limit theorem*.

The second training set uses the Sobol sequence for nine dimensions [207] to create a 'space filling' set. Sobol sequences are known to *gracefully* fill many dimensional spaces with a limited number of samples. That is to say, that for a given number of sample points, the space will be divided by points in a reasonably equal manner, and each subsequent point will be placed into the largest undivided subspace. Unlike pseudo random sets of data, the Sobol sequence is deterministic – each subsequent value fills a gap in the previous set. So the training set is selected from the first part of the set and tested with the same test set as other experiments.

The size of the initial training sets are comparable to the number of simulations required by the *DSO*, which is considered to be the limit of a useful *SMO*. In Section 5.3, the size of the training sets is investigated by reducing the size and observing the error. Subsequent experiments either use new smaller training sets, based on this evidence, or use the full, previously generated, training sets.

5.2.4 *MLP* based Surrogate Model

A simple deep *MLP* is used to predict the output variables. For example, *PPF* and the horizontal x, y positions of the hottest pin. Using the w/o U235 on a per-assembly level as the inputs, this means that the input dimension for a quarter core is nine, while the output dimension of the surrogate model is three (the hottest pin coordinates and *PPF*). For an experiment where the optimisation objectives are the *PPF* and to maximise the distance of the hottest pin from the centre of the core (modelled by $1 - |x, y|$ in relative coordinates). In order to select an appropriate topology for the *MLP*, a short study of network topology was carried out. The

numbers of hidden layers and neurons per hidden layer were varied. Figure 5.2 shows the results. The choice to keep the number of neurons per layer constant was made in order to reduce complexity and keep the design comprehensible on a two-dimensional heatmap. The performance seen in Figure 5.2 confirms that *MLP* networks are robust to a wide variety of topological values. This has been discussed in the literature by many authors (see [24, p130] and more recent deep learning discussion starts with [123]).

Based on the study in Section 5.3.2 a fairly large network with nine layers of 80 neurons was chosen for subsequent experiments. The weights are modified using backpropagation and the 'Adam' loss algorithm ([109] *see also* Section 2.4.5). The network was trained for 500 epochs. In each training epoch, the system trained on a set of 50 randomly selected samples from the training set, N. This helps to reduce over-fitting of the network to the training set compared to training on the whole set, while the number of epochs helps to ensure that all of the training set is tested.

5.2.5 *CNN* based Surrogate Model

The *CNNs* for the microcore are trained on the *per pin* input image. It is significant to note that the possibility exists for CNN type neural networks to predict solutions for arbitrary pin problems. In this study the training set and the inputs are limited to uniform assemblies; however, this limitation is from the training data and the optimisation problem, not of the *CNN*.

The topology of the *CNN* used is shown in Table 5.2. Three convolution and pooling layers are followed by six, fully-connected, feedforward layers. As with the *MLP*, 'Adam' is used to modify weights based on a *MAE*, loss function [109]. The training was carried out for 100 epochs with a sample size of 50. This number of epochs ensures that all training samples are used, while the sample size helps to avoid over specialisation of the network on the training set. Unlike in the *MLP* surrogate model, the *CNN* is used to predict pin-by-pin power of the system, the objective variables are then calculated from this output.

5.3 Design of Deep Learning Surrogate Models

5.3.1 Method

In order to establish the parameterisation of surrogate models for use in the following experiments, a series of short studies have been carried out.

In this section, the design of two types of versatile surrogate models is investigated. The models are *MLP* and *CNN* types. The architecture of the *CNN* is based on designs from literature [122, 124, 117], but the *MLP* design is established by testing the depth and size of

Table 5.2 Keras summary data for the CNN showing topology and modifiable parameters

Layer name (type)	Output dimensions	No. parameters
conv one (Conv2D)	(53, 53, 3)	84
pooling 1 (Average Pooling)	(26, 26, 3)	0
conv two (Conv2D)	(24, 24, 6)	168
pooling 2 (MaxPooling2D)	(12, 12, 6)	0
conv three (Conv2D)	(10, 10, 3)	165
pooling 3 (MaxPooling2D)	(5, 5, 3)	0
dense 1 (Dense)	(3000)	228000
dense 2-5 (Dense)	(250)	1528601
reshape 1 (Reshape)	(51, 51)	0

rectangular networks exhaustively. The size of the training set and the type of loss function are considered by testing the design and observing the error on the training set.

Three loss functions are investigated for types of error over the training run. These are *MAE*, mean squared error, and mean squared logarithmic error. The *MAE* is much more computationally efficient than the other two loss functions, since the operation of setting the sign bit is a single 'bit' operation, rather than numerous floating point operations for mean squared or mean logarithmic error. The mathematical significance of mean squared error is that large error values are amplified making them more significant to the loss algorithm than *MAE*. Larger error values are more easily divided into weights when carrying out backpropagation. The significance of mean logarithmic error is that it selectively punishes outlier error values, encouraging the algorithm to be more generalising than *MAE*.

The size of the training set is critical to the performance of deep learning surrogate models. However, if the training set becomes too large it competes with the computational cost of the *DSO*. The performance of *MLPs* and *CNNs* training on random and Sobol training sets of different sizes is considered based on the *MAE* of the network on the same random test set.

5.3.2 Results

Figure 5.2 shows the error heatmap for different geometries of rectangular *MLP* type neural networks. The heatmap shows the error for networks with one to one hundred and fifty neurons per layer and one to twenty hidden layers. Although it is possible to iterate this heatmap many times to generate a smoothed version, it is instructive to inspect results for a single seed value. When the author produced averaged results, interesting features of the heatmap related to the reliability of results were lost.

Unsurprisingly, the heatmap shows that networks with few neurons per hidden layer generate poor results. These results are explained by a network with a few neurons being functionally unable to generate a function with the complexity of the objective function. When a network is too deep it also fails to train, this is likely due to the problem of backpropagation of small error values through many layers of network. At each layer the error differential must be distributed to all the weights. This value can quickly become too small to be represented in a floating point representation (see p. 39 for a description of the backpropagation algorithm). Both of these areas fail catastrophically and generate error values in the region of 20%.

When the network becomes too large (in the top right hand quadrant), the results become unstable. This is likely due to the CPU and memory being overwhelmed. Anecdotal results here have found different behaviour on different computers, although the general behaviour is a rolling increase in likelihood of catastrophic failure to contain the error magnitude. Very large and *Not A Number* (*NAN*) results can be returned in this area and are shown in white on the heatmap.

A mild gradient of decreasing error can be observed in networks with a depth from one to five layers and above 50 neurons per layer. This is the increase in performance developed in 'deep learning' designs (see p. 43 for more discussion of this).

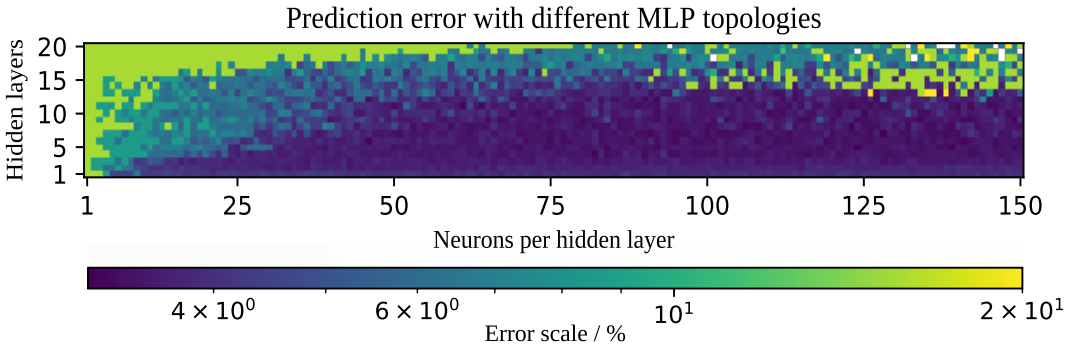


Fig. 5.2 Heatmap of *MLP* topologies (seed=3454321) [233]

The *MLP* networks used in this study are chosen to have a topology of 80 neurons per layer and nine hidden layers. This was determined to be suitably far from all of the regions of high error described above.

Figure 5.3, shows a graph of the *MAE* when an *MLP* network is trained with three different loss functions. The logarithmic loss function fails to converge. This is because the error differential becomes disproportionately small at low values, making it impossible for the backpropagation algorithm to work. The square error and the *MAE* perform similarly, although the square error appears to be less stable. Based on the improved average performance, the *MAE* is used as the loss function in all subsequent experiments.

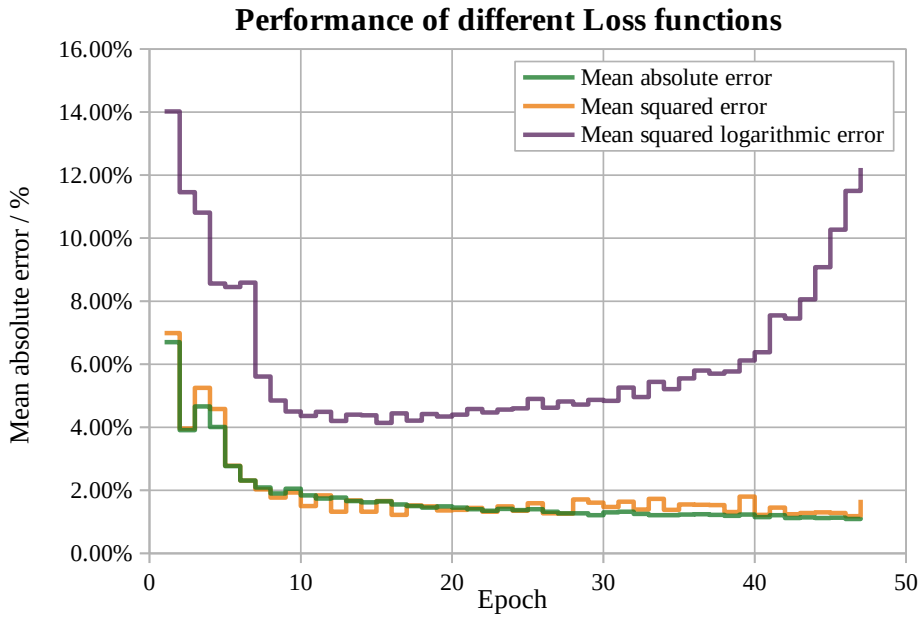
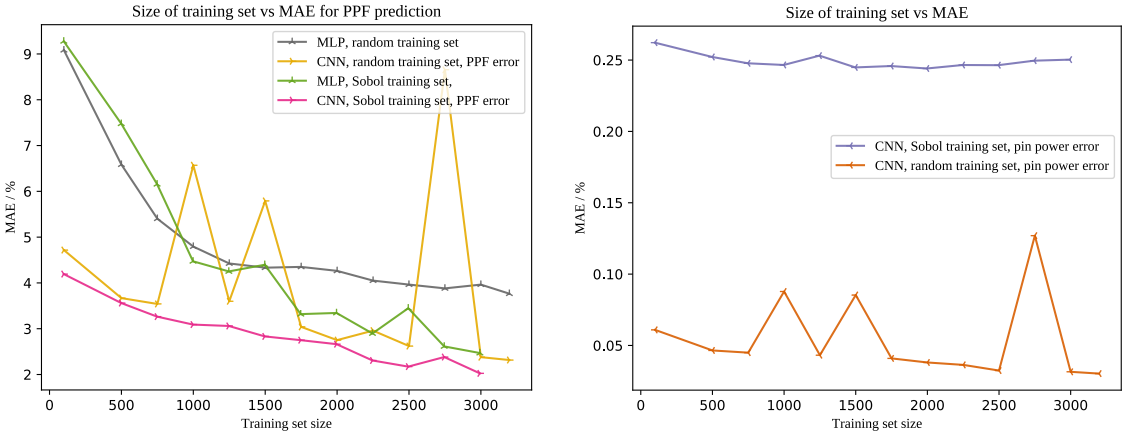


Fig. 5.3 Comparison of mean error performance evolution over training for three loss functions, applied to the *CNN* surrogate model.

In Figure 5.4, it can be seen that surrogate models trained on Sobol sequences outperform random sets for both *MLP* and *CNN* surrogate models for predicting PPF. This result was surprising, given the expected convergence of the mean of multivariate inputs through the *central limit theorem* to bias a random training set *towards* a mid-range mean.

The error of the *CNN* model for random data showed much more highly erratic performance than either the Sobol *CNN* or the *MLP* tests. The results were calculated 5 times per training set size and exhibit a fairly high variance. The pin-for-pin results in Figure 5.4b for the *CNN*, show that the random surrogate model predicts the pin values up to five times more accurately than the Sobol *CNN*. This might be explained by the Sobol algorithm having been trained on a space-filling training set, so it operates less effectively on average, but is more efficient at predicting significant pin powers as observed in Figure 5.4a. In order to use a *CNN* that has trained to predict per-pin powers, it is important to ensure that the objectives actually correlate. The *CNN* predicts pin powers with a mean accuracy of 0.64%, but the *PPF* is predicted with much larger errors. The increase in error is because the *PPF* is particularly susceptible to the error on outlier values, whereas the loss function of the neural network rewards prediction on average. Despite quite high actual error between the predicted value and the objective parameter, the correlation of the variables is good, as shown in Figure 5.5. This makes using it as an optimisation objective acceptable, at least in the domain of the test set used in this instance.



(a) PPF (a derived parameter for the CNN)

(b) CNN, pin powers

Fig. 5.4 MAE of *MLP* and *CNN* models, trained on diffferent sized training sets and tested on the same random test set (mean results for 5 iterations, seed = 221179).

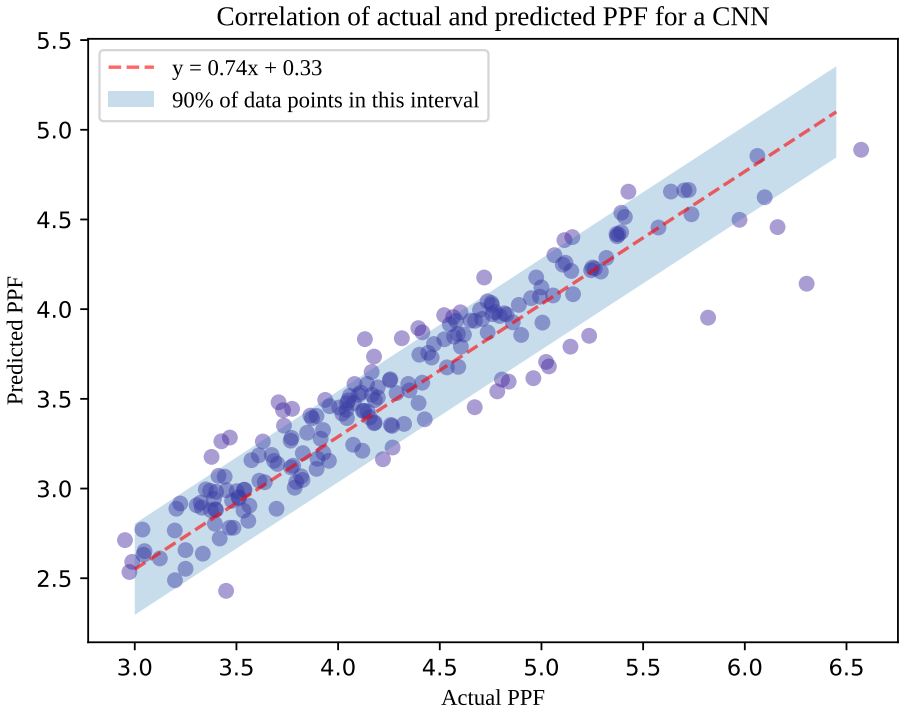


Fig. 5.5 CNN parameter prediction error for random *LPs*

Thus far in the chapter, short studies have been used to establish the design parameters of the two deep learning surrogate models. In the following four experiments, these deep learning

surrogate models are used to demonstrate *SMO*. The first two experiments (Sections 5.4 and 5.5) prove the concept of *SMO* applied to the startup conditions of the microcore, firstly by optimising for the radial distance of the hottest pin vs the *PPF*, then by considering *PPF* vs mean enrichment. The third experiment (Section 5.6) shows that the surrogate model can be applied to more computationally expensive problems by optimising for *PPF* vs cycle length in a study that requires burnup simulations. Finally, Experiment 4 (Section 5.7) shows the scalability of the work beyond the 'toy' example of the 6×6 microcore, by applying the optimisation to an initial core loading problem in an *SMR* core, similar to the Nuscale design described in Section 1.1.5 and Figure 1.2d.

5.4 Experiment 1: BOL PPF vs Position of Hot Pin

5.4.1 Method

NSGA2 was used with the *MLP* and *CNN* surrogate models and in a *DSO* using Serpent for solution evaluation. The *NDF* for the populations of *SMO* processes were then re-evaluated using Serpent. The *SMO* outputs are compared with the *NDF* from populations of different generations in the *DSO*. When NSGA2 runs using an existing surrogate model objective function, the computational resource usage is less than one hundred thousandth that of the direct optimisation process.

Based on the trends seen in Figure 5.4, the networks were trained on a set of up to 3200 uniform randomly generated core designs and predictive performance was evaluated using another 800 similarly random designs, while models trained on Sobol training sets were trained on the first 3000 core designs and tested on 800 random core designs. The size of the training set is comparable to the *DSO* requirements, the aim being to create the best possible surrogate model and compare the *NDF* results of surrogate and direct optimisation approaches.

The loss function used is *MAE* and the surrogate models are trained for 50 epochs on 100 samples. These numbers are chosen to maximise the chance that all training samples are used while training, and the epoch approach avoids overspecialisation of the network.

5.4.2 Results

Simulations show that the surrogate models achieve moderate errors for the objective variables on a test set of random data and are able to generate results using hundreds of thousands of times less computational resource. The *CNN* predicted pin powers with a *MAE* around 1%; however, this translated to *MAE* values of 3.805% and 2.421% for the actual objective variables over the test set. This is likely due to the training of the *CNN* to predict pin powers *on average*,

whereas the *PPF* depends upon *the highest outlier* value. Execution time measurements in Table 5.3 are shown relative to *MLP* training time in order to compensate for effects specific to the hardware and software platform. For reference, the mean *MLP* training time was 172.4 *CPU* seconds over 30 iterations on the low-end laptop hardware used.

It can be seen from Figure 5.6 that the *MLP SMO* process yields an *NDF* that has similar performance to the *NDF* of the direct optimisation for generation five, while the *CNN SMO* process has a *NDF* that has similar performance to the twelfth generation.

Table 5.3 MAE for *MLP*, *CNN* and Monte Carlo on training data (mean values for 30 runs)

	MLP	**CNN**	**Monte Carlo**
Training set, N	3200	3200	
Test set	800	800	
MAE per-pin /%	-	0.9949	29.31E-5
error for PPF/%	2.374	3.805	29.31E-5
error for hot pin /%	4.051	2.421	-
relative CPU time (evaluation)	10^{-7}	10^{-5}	$\sim 18^{a}$
relative CPU time (training)	1.000	15.26	-
relative CPU time (NSGA2)	0.03	0.4	$\sim 10^{7}$ [a]

[a] HPC nodes used for Monte Carlo (see Table D.1 for hardware specs.)

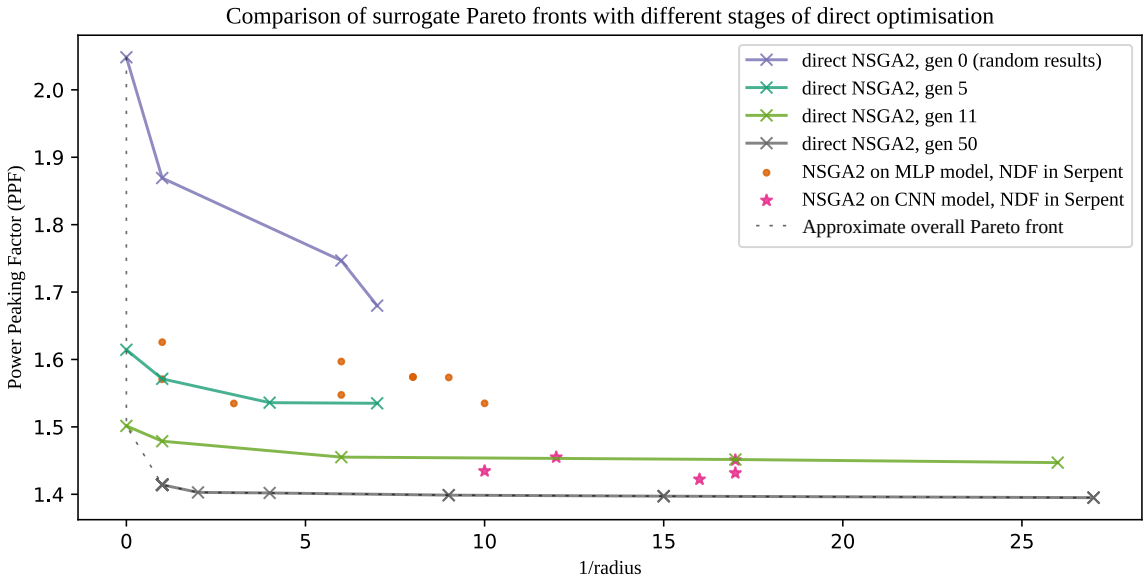


Fig. 5.6 Sample *NDF* for populations in a DSO vs MLP and CNN surrogate models trained on random input data ($N = 3200$)[233]

The *MLP* outperforms the *CNN* at predicting *PPF* but underperforms at predicting the hot pin location. However, the CNN performs better when predicting non-random (optimisation) input data. This is believed to be because the *CNN* is predicting pin power while the *MLP* is predicting *PPF*, which is derived from pin powers. This may cause the the *CNN*'s results to be more robust to the extrapolation that occurs when the optimisation search moves the input space away from the random training set.

When the Sobol trained surrogate models are applied, the *CNN* shows an improved performance in terms of the position of the hottest pin but does not dominate for the PPF, as shown in Figure 5.7.

If the surrogate models investigated here could be used to seed the *DSO* population, they have the potential to reduce computational resource usage by up to 25%, assuming that the training set already exists. However, the training set (at 3200 samples) is not justified in terms of computational expense for this experiment. If data created during a design study already exists, such as in previous work [236], or if an optimisation is going to be carried out on a regular basis, then creation of a surrogate can be justified. Random training data is equivalent to the random initial populations used by many iterative optimisation algorithms (including NSGA2) and might be reused to create a surrogate model. The *CNN* surrogate model significantly outperforms the *MLP* model for *SMO* at generating lower *PPFs*. If the optimisation algorithm used a population of 300, as per [130], and the CNN surrogate still performed comparably for twelve generations, then a net computational saving would be seen. This is not inconceivable since population size primarily discourages premature convergence to local optima rather than changing the rate of *NDF* progression [78].

This initial study has shown that a deep learning surrogate model can generate a final population that has migrated away from the random range of results and towards the *NDF*, and that the calculation of results from the surrogate model is extremely computationally efficient compared to direct Monte Carlo simulation of the system neutronics. Further work should establish the investment versus benefit of SMO with regards to direct optimisation where population size is similar to that in other studies using NSGA2 (see Section 5.2.2).

By training the surrogate model on a more space-filling training set, the surrogate model appears to compete more effectively during optimisation. Smaller training sets can also be considered as an approach to justify the surrogate model training set. Although the parameter error decreases as size of training set increases, as seen in Figure 5.4, the highest improvements are when the training set goes from 100 to 1500, justifying smaller training sets. Finally, it should be noted that an assumption has been made that low *MAE* on a random test set correlates to *SMO* performance. Although this follows logically, it may be possible to use surrogate models that have high *MAE* but good correlation.

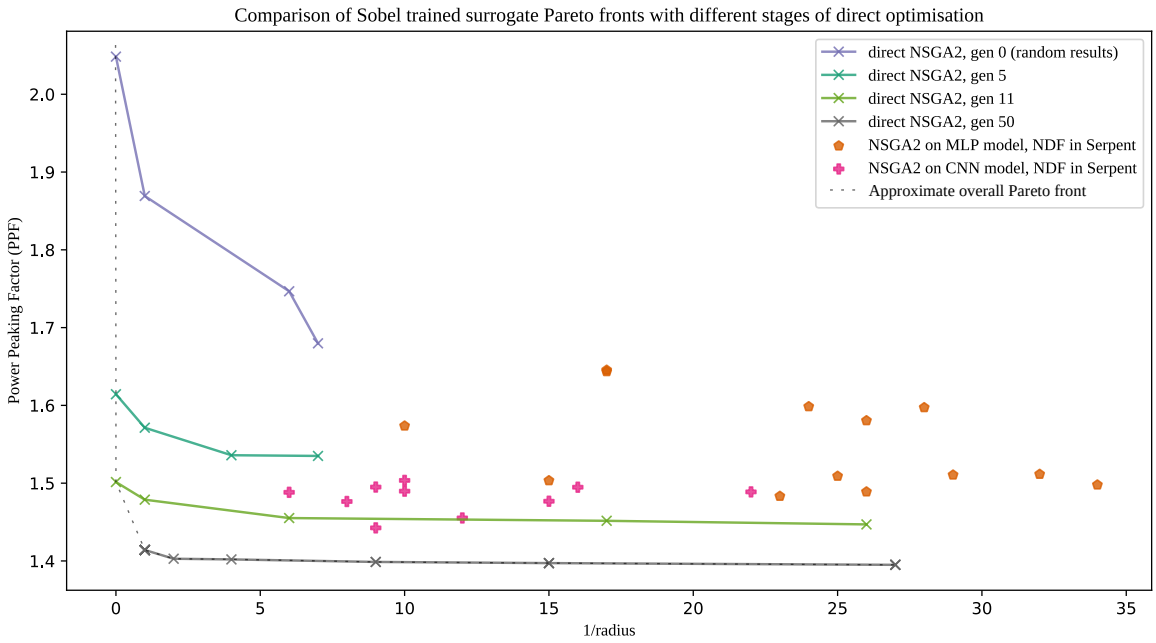


Fig. 5.7 Sample *NDF* for populations in a *DSO* vs *MLP* and *CNN* surrogate models trained on Sobol set input data ($N = 3000$)[233]

5.5 Experiment 2: BOL PPF vs Mean Enrichment

5.5.1 Method

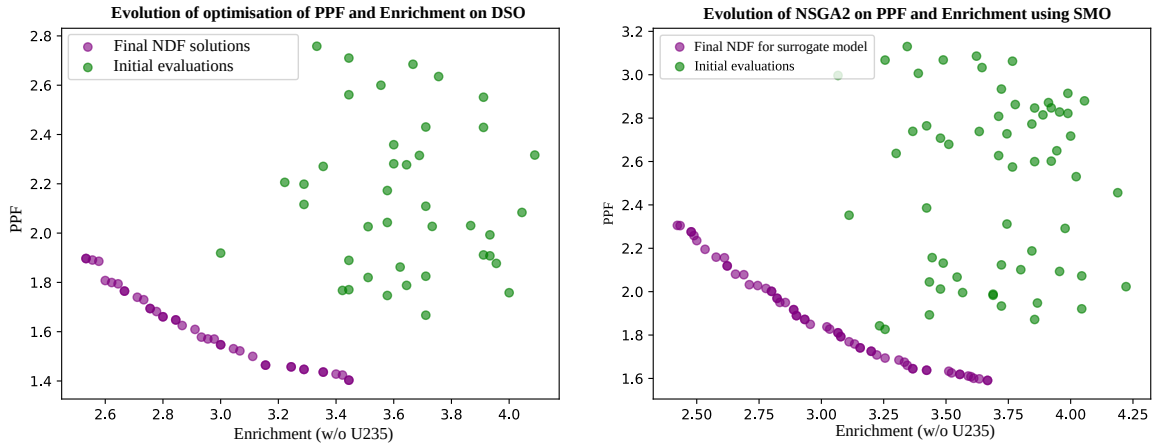
An advantage of *SMO* is that once a training set has been created, surrogate models can be created or re-used for subsequent studies at minimal computational cost. In this experiment, the *CNN* model from the previous experiment is repurposed to study other objective variables.

A lower *PPF* allows a higher average coolant temperature at the outlet of the reactor without violating the *DNBR*, which is advantageous in terms of generator efficiency and the overall efficiency. Lower enrichment enables cheaper fuel, since the isotopic enrichment of uranium is expensive, due to the similarity of atomic weights of the naturally most common isotopes U238 and U235. In this experiment the enrichment and *PPF* are minimised.

A *DSO* is carried out using NSGA2 and the parameters shown in Table 5.1. The same optimisation is then carried out with an *SMO*. The final *NDF* of the *SMO* is then evaluated in Serpent and the solutions that are achieved are compared.

The aim of this experiment is to establish whether the *SMO* could be used to find similar results to the *DSO* without the computational expense. For this reason the *DSO* and *SMO* are run a number of times with different seeds. This allows the comparison of *NDFs* generated.

5.5.2 Results



(a) DSO, $\sim$ 200 cpu hrs, (HPC) (b) CNN SMO, $<$ 0.012 cpu hrs, (laptop)

Fig. 5.8 Examples for *DSO* and deep learning *SMO* initial and final population results

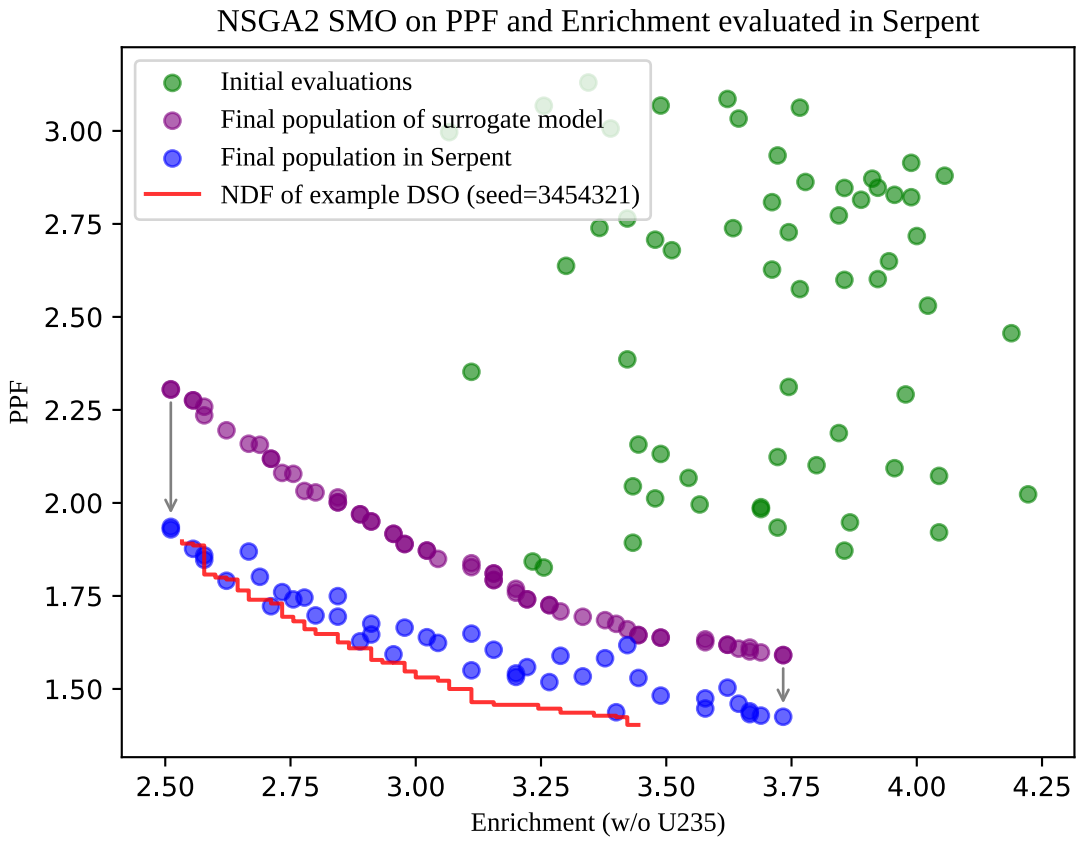


Fig. 5.9 *NDF LPs* for the *CNN SMO* simulated in Serpent.

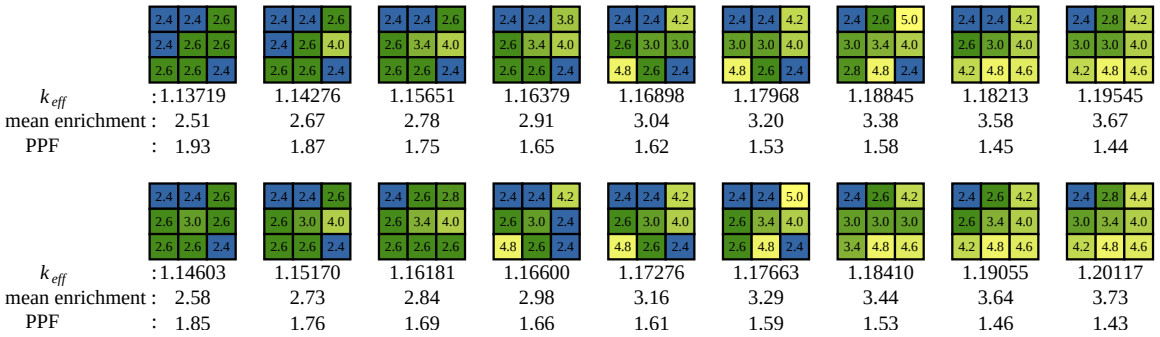
Table 5.4 Execution time for *MLP*, *CNN* and Monte Carlo (mean values for 30 runs)

	population	generations	**CPU time/s**	**machine**[a]
DSO using Serpent[b]	50	60	$\sim 3.6 \times 10^6$	HPC
DSO using WIMS[b]	50	60	$\sim 9.0 \times 10^5$	Lise
CNN SMO	50	60	40.0	laptop
CNN SMO($p = 100, N = 1000$)	100	1000	1300.0	laptop
50×Serpent evaluations	-	-	63600	Lise
50×MOC evaluations	-	-	15000	Lise

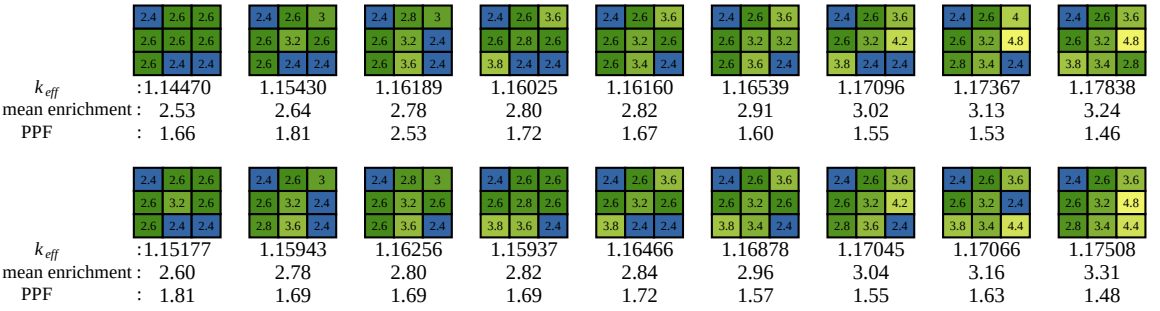
[a] for machine descriptions see Table D.1, [b] Calculated from direct simulation times and NSGA2 parameters

Figure 5.8 shows the results of NSGA2 running a *DSO* and an *SMO* using a *CNN* model. For the purposes of this experiment, the surrogate model is assumed to have been pre-trained. Figure 5.8a shows the initial population and final population for NSGA2 optimising for mean enrichment and *PPF* using a *DSO*. The nine assemblies of the microcore (Figure 4.3) and the *PPF* evolved. Initial tests in this experiment were found to achieve a low power peaking factor by having a low k_{eff} (< 1.0). This is because subcritical reactors appear to exhibit extremely low *PPFs* in the Monte Carlo simulation (as the initial population of simulated neutron chain reactions rapidly die out). To ensure that the system did not use this strategy, the minimum enrichment allowed was 2.4 w/o U235. These figures show that the algorithm is capable of optimising both the *DSO* and the *SMO* problems, although the optimised results for the *SMO* give higher estimated values for *PPF*. A visual inspection of the *NDF* results for both *SMO* and *DSO* shows structural similarities. The results, shown in Figure 5.10 are arranged in ascending order by mean enrichment in vertical columns from left to right. By inspection, we can see that the *SMO* framework generates solutions that look reasonable at least at a superficial level. It is also apparent that at least one edge of the *NDF* does not achieve the global minimum solution (the lowest possible enrichment is uniform 2.4w/o U235). This is unsurprising for two reasons. Firstly, the search is conservative in terms of computational expense in order to compare with the *DSO*. Secondly, the 'edge cases' of the *NDF* in a converged population may be difficult to obtain, because crossover operations will create offspring between the parents in the search space and, therefore, away from the edges of the *NDF*.

The computational budget for the *DSO*, given in Figure 5.8, is based on the actual optimisation, which was carried out in Serpent. Serpent is a Monte Carlo type simulation, which achieves high accuracy over arbitrary geometries but at a high computational cost. It is instructive to compare the same cost with a more economical simulation. Although the author has attempted to generate the same geometry in the WIMS code, it is not possible due to the rotational boundary condition used. This feature does not exist in version 10 of WIMS, the



(a) Selected (18/44) *NDF LP* arrangements generated by a *CNN SMO* for the bottom righthand quadrant of the microcore (seed=3453412, `pygmo_micro.py`)



(b) Selected (18/32) *DSO LP* arrangements generated for the bottom righthand quadrant of the microcore (seed=3453421)

Fig. 5.10 Examples for *DSO* and *SMO LPs*,

stable release at the time of writing. However, similar experiments, using a reflected boundary conditions take around 3 minutes per burnup step on the 'Lise' blade computer. This means that the equivalent *DSO* optimisation would take $\sim$ 120 CPU hours. Although it is dangerous to compare simulations from different computers, the aim of these numbers is to show the reader that the surrogate model in this case is running ***four orders of magnitude faster*** than direct simulations, even on entry level hardware, compared to high end hardware used for the *DSO* simulations. Figure 5.10 shows side-by-side results from the *DSO* and *SMO*. The solutions found by both strategies compare favourably to a recognised strategy, called an *in–out loading pattern* [208, p. 210]. This is where the high reactivity fuel is loaded at the edge of the reactor and lower reactivity fuel is placed at the centre.

It is known from the training sets of the *CNN* that the prediction performance of the surrogate model is reliably a few percent (see Table 5.3) *on a random test set*. However, it is also known that the designs of interest have structure and are not random. Furthermore, due to the *central limit theorem*, a set of nine random uniform variables will have a distribution whose

macro parameters (such as mean enrichment) are no longer evenly distributed. This means that it is possible that the surrogate model is operating in a region of the search space that it has not been trained upon and is likely to be performing poorly compared to the *DSO*. For this reason the *NDF* is simulated in Serpent to establish the actual performance of the *SMO*.

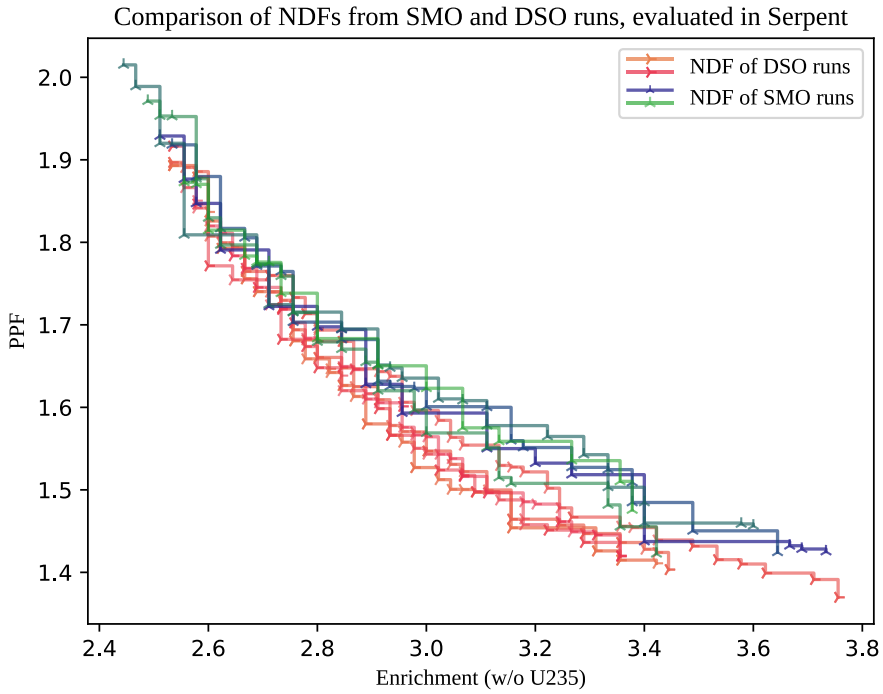


Fig. 5.11 Repeated runs show competitive results between the *SMO* and *DSO*.

Figure 5.9 shows the results of the *SMO* from Figure 5.8b with actual direct simulation results for the final *NDF*. As expected, the results are no longer accurate. It can be seen that the surrogate model over estimates the *PPF*, leading to an improved result from the solutions than the surrogate model predicts. The calculation of enrichment is a linear problem in terms of input variables (it is the mean of them); however, *PPF* is a non-trivial product of the inputs and a product of the particular geometry, neutronics and inputs. This makes finding all *NDF* results apart from the limit case (where each input enrichment is minimised), more difficult.

In order to confirm that the *SMO* reliably competes with the *DSO*, the experiment has been run for a number of seed values as shown in Figure 5.11. Here the *NDFs* are plotted to show that the *SMO* reliably generates competitive solutions. In a few cases, the *SMO* finds solutions that dominate the *DSO*.

This experiment has shown that *SMO* can generate competitive results for *LPs* over two objectives compared to a single run of the *DSO*. Based on the execution times shown in Table 5.4, the *SMO* is seen to be many times more efficient than an equivalent *DSO*. If the training

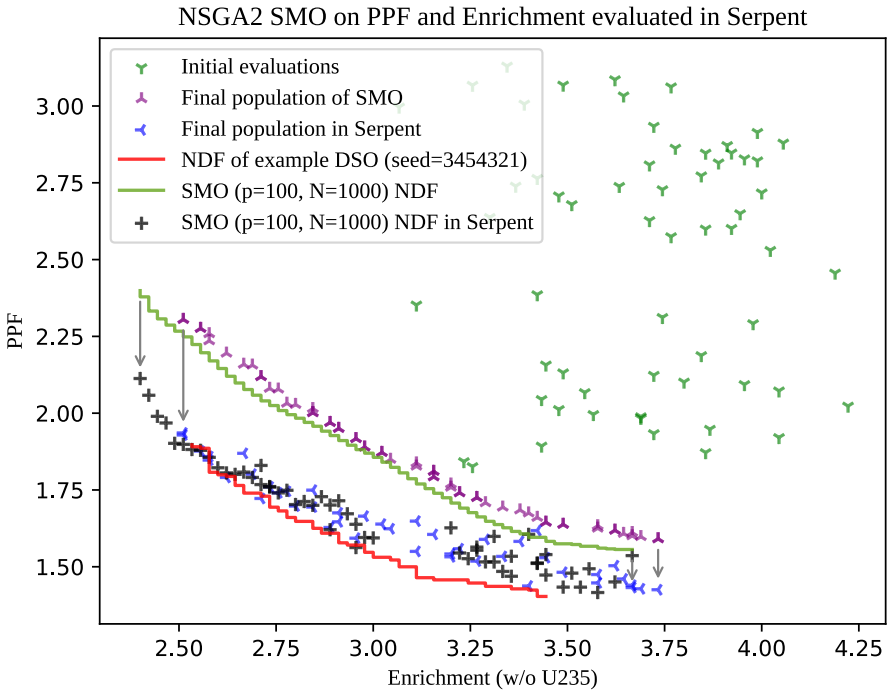


Fig. 5.12 Comparing previous results with a more exploratory *SMO* ($p = 100$, $N = 1000$).

set exisits, a computational cost improvement of ~ 690 times can be observed between running *SMO* with a much higher population and generation count. Including testing the *NDF* in the *worst case*,the most exploratory *SMO* is still seven times more computationally efficient, compared to a *DSO* running at a much lower population$\times$generation count. This result, does not consider the further actual gains that can be made by parallel evaluation of direct simulations, but is, in itself, a significant result of the thesis.

Table 5.5 compares the hypervolumes of the *DSO* and *SMO* from the runs shown in Figure 5.11. From this data it can be seen that the *SMO* appears to performs slightly better than the *DSO* for this metric on these seed values. However, the difference in hypervolumes is only two percent – an amount that can not be relied upon with only a few iterations. The *DSO* could not be repeated further within the allotted HPC budget without limiting the work of other students. However, it is very clear that the *SMO* results are competitive in terms of objectives and certainly thousands of times faster.

Further analysis using the two-dimensional extensions of the K-S Test by Peacock [178] does not conclusively show that the *DSO* and *SMO* distributions are significantly similar or different (found in Appendix B.3).

The fact that the *DSO* cannot be run more than five times shows the value of the *SMO* technique. Using it, it is possible to carry out many more exploratory experiments than would be possible with *DSO* alone.

It is important to recognise that the hypervolume indicator does not capture all aspects of the performance of a *MO*. Inspection of Figure 5.11, shows that the *DSO* appears to reliably dominate the search in the bottom right quadrant, while the *SMO* is more competitive in the top left. It follows that the *SMO* increases its hypervolume by extending the *NDF* further.

Table 5.5 Hypervolumes for *DSO* and *SMO*, calculated with a reference point at $(5,5)$

	DSO	**SMO**
Hypervolumes	8.695312	8.689278
	8.694719	8.605199
	8.145597	8.831682
	8.681574	8.771061
	8.528866	8.464937
Mean value:	**8.549214**	**8.672431**

Figure 5.12 shows the results presented previously, with an overlaid line showing the maximum *NDF* achieved with the *SMO* ($p = 100$, $N = 1000$) simulation, where the NSGA2 parameters are chosen without the constraints of computational cost incurred by conforming to the limitations of the *DSO*. In this case the p value is chosen to be 100 (the limiting factor on population is checking the *NDF* values by direct simulation) and N has a value of 1000. With these parameters the *SMO* takes around thirty minutes on the laptop hardware (see Table D.1), which would be equivalent to 10% of the first generation on of the *DSO*. In particular, the larger *SMO* is able to find the edge of the *NDF* (this is the trivial, uniform 2.4 w/o U235) but this is not found by either the *DSO* or the *SMO* with less exploratory parameters.

5.6 Experiment 3: EOC Burnup vs PPF

5.6.1 Method

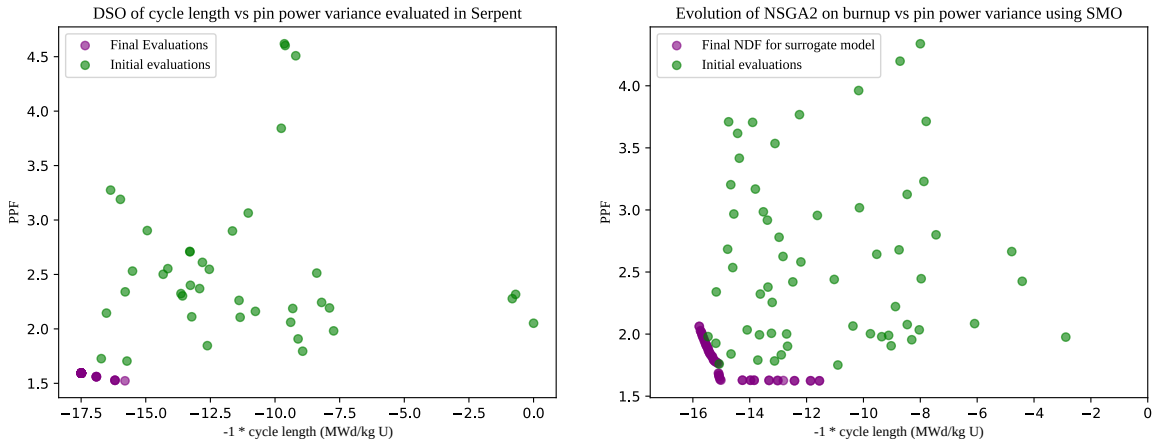
In this experiment the lifetime operation parameters of a single batch microcore is considered. The objectives are to extend the cycle length of the design (minimising negative cycle length), while minimising the *PPF*. The input parameters are the enrichments each of nine uniform uranium assemblies (w/o U235 from 0.8 to 5.0).

A training set of enrichments that conform to the first thousand entries of the Sobol set was generated and simulated in Serpent. The aim was to establish if the objective parameters could

be reliably predicted by a suitable surrogate model, and if that surrogate model could be used for *SMO*. The test set used was the first generation of the *DSO*.

The surrogate model used is an *MLP* type deep learning model. Due to the extremely high cost of running the *DSO* on the HPC, a single run of the *DSO* is provided for comparison. As with all experiments, the code required to validate these results is provided, and it is hoped that this will allow validation and reproduction of the claims.

An initial version of this experiment that used pin power variance as an objective of the experiment is presented in Appendix B.4.



(a) DSO, $\sim$ 4000 cpu hrs, (HPC)

(b) MLP SMO, $<$ 0.0019 cpu hrs, (laptop)

Fig. 5.13 Examples for *DSO* and *SMO* initial and final population results.

5.6.2 Results

In Section 5.5, the training set had already been generated for the investigation of training set size, (Figure 5.4). Based on the trends of this graph, it made sense to use a large training set, since error continues to reduce. In this experiment, the training set must be generated, and a balance between model accuracy and computational resource must be found. Based on Figure 5.4, one thousand training samples is selected, as this gets around 70% of the performance observed from the largest training sets.

The *DSO* and *SMO* runs for the *PPF* and cycle length show that both simulations reach similar *NDFs*, and that the values found are in similar areas, as shown in Figures 5.13a and 5.15b. It is of interest to note that the *MLP* surrogate model used for this burnup experiment does not perform well on the random initial population.

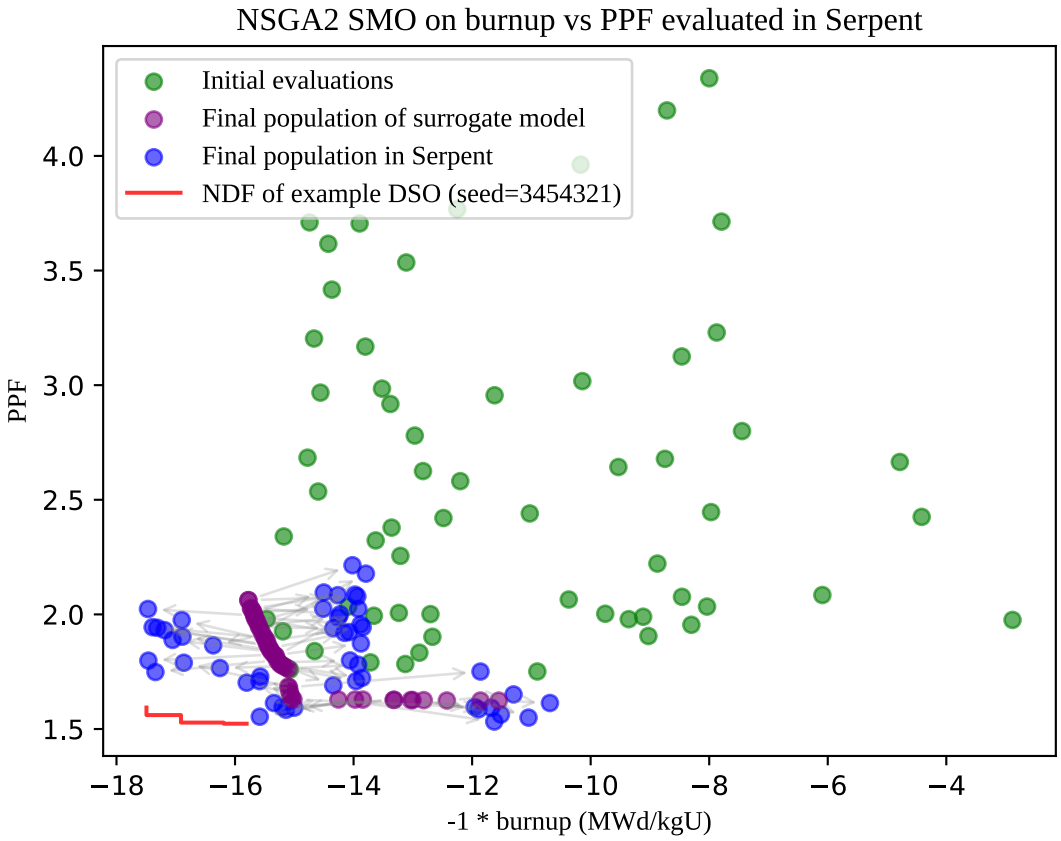


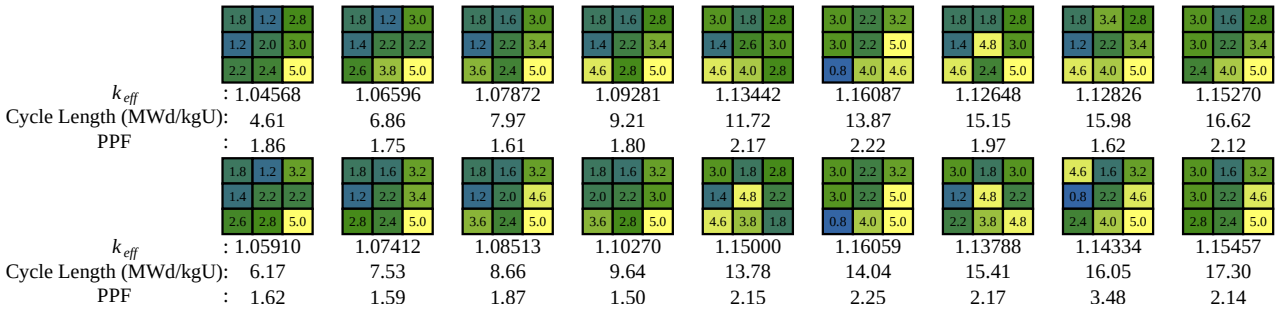
Fig. 5.14 *NDF LP* for the *CNN SMO* simulated in Serpent.

Figure 5.14 shows the surrogate *NDF* results simulated in Serpent. The *SMO NDF* does not generate performant results in this case when compared to the *NDF* of the *DSO*.

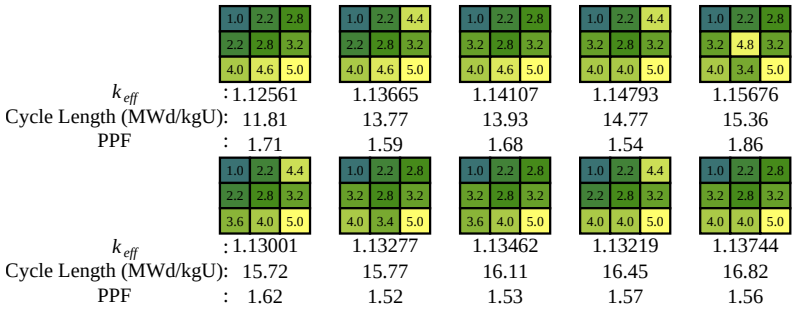
The *SMO NDF LPs* generates 52 solutions, whereas the *DSO LPs* in Figure 5.15 generates only 10. The greater diversity of solutions generated by the *SMO* is interesting as it allows an efficient use of the computational resource to directly simulate results that are close to the *NDF*. This experiment shows that a surrogate model can be used to explore parts of the *NDF* that have not been found by an equivalent run of the *DSO*. In the next, and final, experiment of this chapter the scalability of the approach is examined, by considering an *SMR* core.

5.7 Experiment 4: SMR Core

In the previous experiments, the author intentionally used a hypothetical, square reactor. This ‘toy’ problem is used to provide a first examination of the applicability of these techniques



(a) Selected (18/52), *LP* arrangements generated by a *CNN SMO* (seed=3454312, `pygmo_micro.py`)



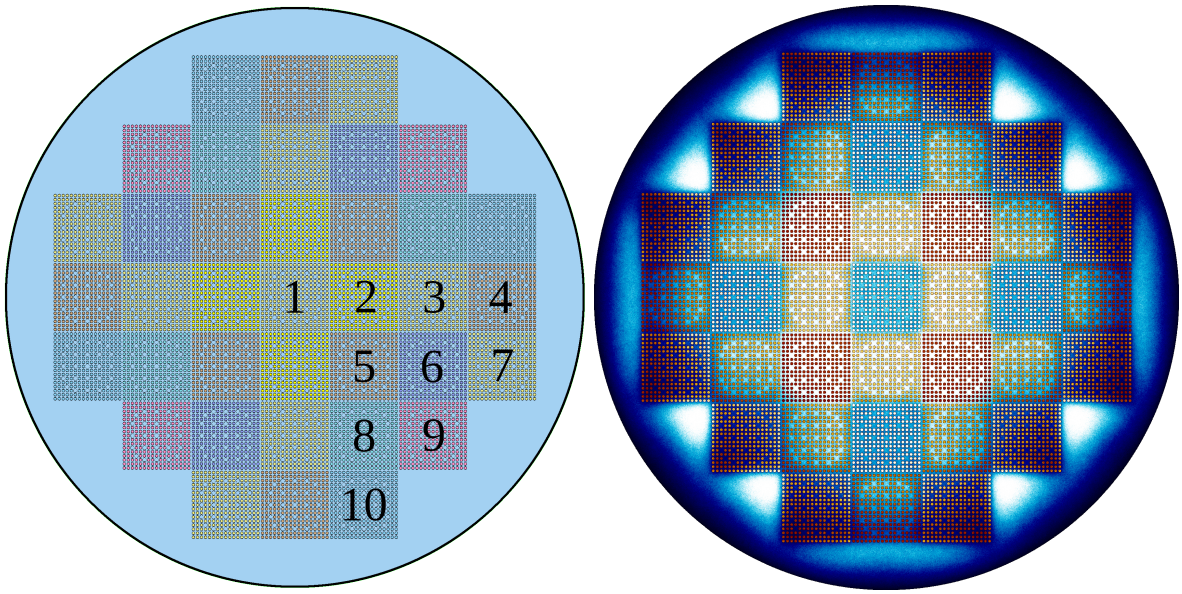
(b) Entire populatin of *DSO LP* arrangements generated for the bottom righthand quadrant of the microcore (seed=3454321)

Fig. 5.15 Examples for *DSO* and *SMO LPs*, maximising burnup and minimising *PPF*

and focus the investigation on deep learning *SMO*. For the final experiment with deep learning surrogate models, a small, single batch, *SMR* based on the proportions of the Nuscale core is optimised using the same principles as the other experiments, to show how easily the benefits of this technique can be applied to more realistic problems for nuclear engineers.

5.7.1 Method

An initial single batch core loading pattern is evaluated for *PPF* and enrichment to establish the *NDF*. Due to the high computational cost of quarter core evaluation in Serpent, data from previously collected training sets can be used as a starting point for the convolutional portions of deep learning *CNNs*, creating a useful contribution for subsequent work. The convolution layers are imported from the experiment in Section 5.5. A small Sobol training set of 1000 simulations is used to train the feedforward layers of the model, and the initial population of 50 simulations from the *DSO* is used as the test set. The setup is shown in Figure 5.16a, with the flux map shown for reference in Figure 5.16b. In order to extract the fission rate of the



(a) SMR mesh (b) Example SMR flux map (seed = 1586624087)

Fig. 5.16 A small SMR with order 4 rotational symmetry.

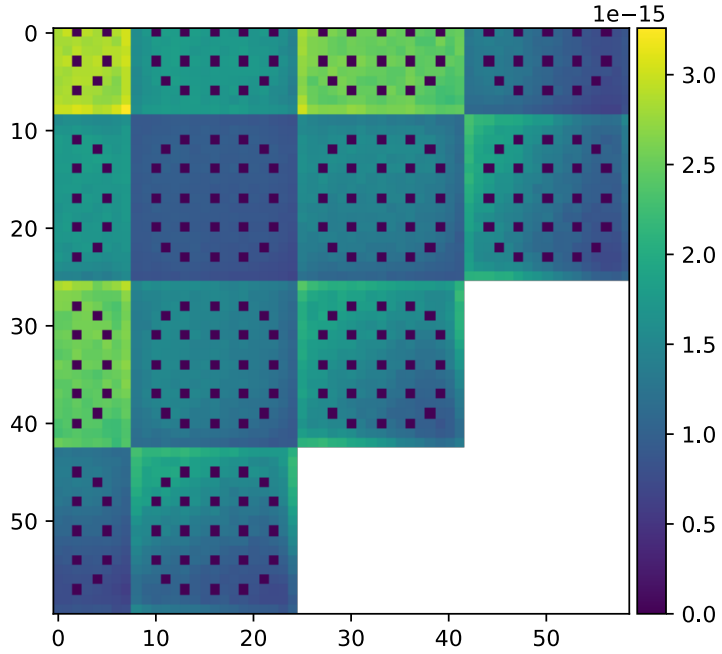
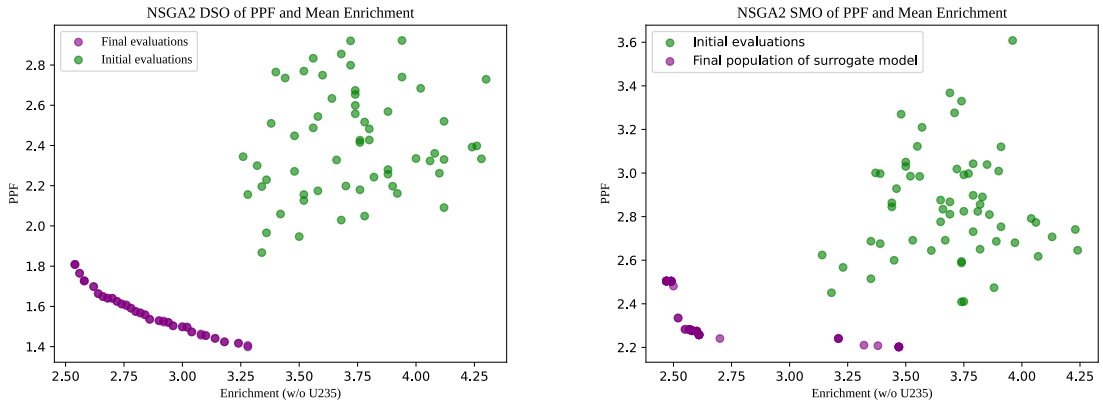


Fig. 5.17 Total fission energy per pin for the SMR in Serpent. The detector data must be reconstructed around lines of symmetry to be analysed (*seed* = 1586624087).

pins a 'detector mesh' must be created and parsed from the output files. The results of the

reconstructed mesh is shown in Figure 5.17; the detector data is padded with zeros for the purposes of prediction in the *CNN*.



(a) DSO, $\sim$ 1500 cpu hrs, (HPC)

(b) CNN SMO, $<$ 0.08 cpu hrs, (laptop)

Fig. 5.18 Examples for *DSO* and *SMO* initial and final population results for experiment 4.

5.7.2 Results

The results for this section are presented in the same format as other experiments. Firstly the results of the *DSO* and *SMO* runs are shown in Figure 5.18a and Figure 5.18b respectively. The results are then presented with the direct simulation of the *SMO NDF* in Figure 5.19. In this case the surrogate model does not compete well with the *DSO*, generating results that have a much higher estimated and actual *PPF*. The low accuracy of predictions of the deep learning surrogate models in this experiment may be due to a requirement for a larger training set on a larger problem. Interestingly the *SMO* still generates results that have a lower mean enrichment. The *SMO* extends the total discovered *NDF* by 13.6%.

5.8 Summary

The contribution of this chapter has been to design and develop two deep learning surrogate models, train them and apply them to four experiments in optimisation of simple, once-through loading pattern problems. At the time of development convolutional neural networks had not been applied to the problems of fuel loading patterns.

Significant advantages have been demonstrated to using surrogate model techniques to augment and aid the application of *DSO*, when solving problems in nuclear engineering. While the technique is not expected to replace *DSO*, it has been shown to beneficially augment it in a

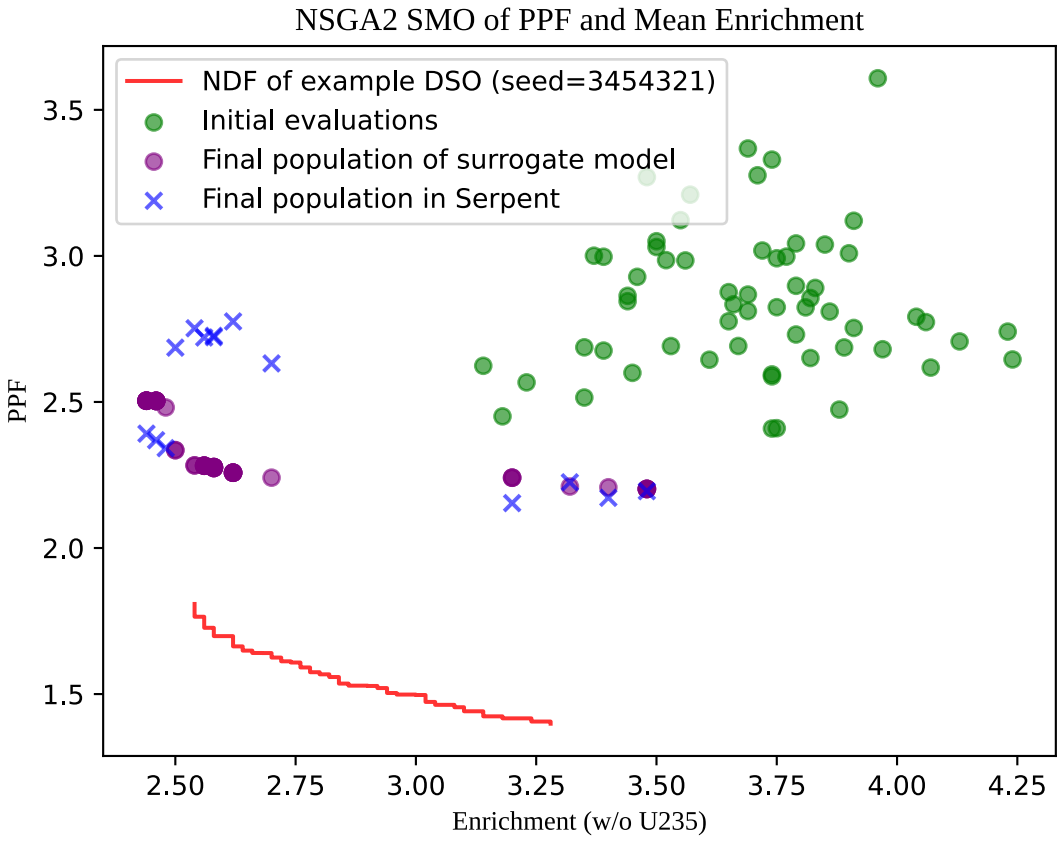
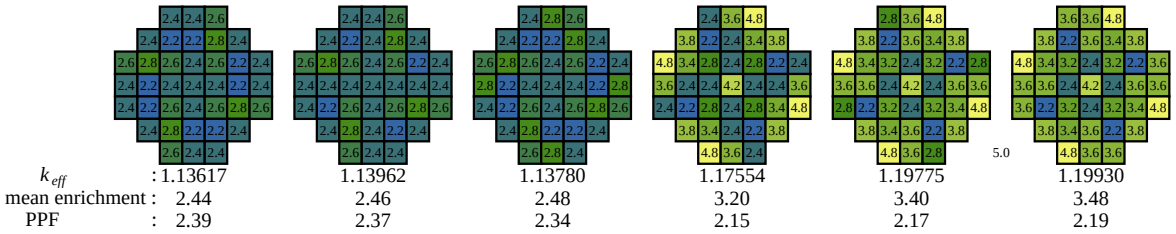


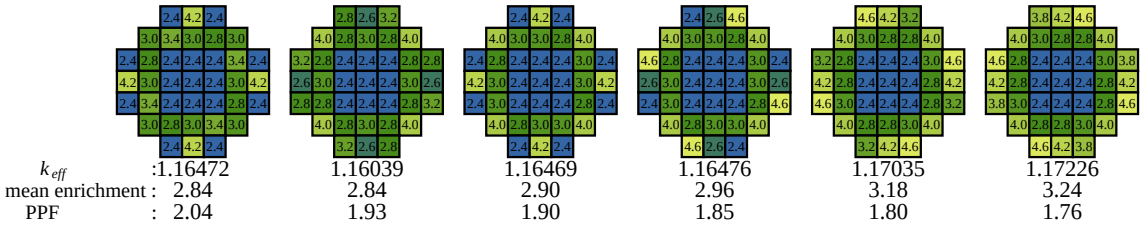
Fig. 5.19 *NDF LPs* for the *CNN SMO SMR* simulated in Serpent.

number of scenarios, especially where computational expense is a limiting factor, *SMO* can generate non-dominated solutions, reducing the computational burden of repeating the same *DSO*, and augmenting the results with low and in some situations negligible computational cost. Many *DSO* results in this chapter were achieved using the Cambridge HPC, which is a significant computational resource that is not available to every researcher or nuclear engineer.

The results of Experiment 1 (p. 84) show that surrogate models can generate results closer to the *NDF* than early stages of optimisation, potentially enabling acceleration of a *DSO* at the start of the process. This has immediate applications when a corpus of training data has already been produced; data created during a design study or as a by-product of traditional iterative optimisations could be repurposed for this. The requirement of a large training set offsets the gains of the surrogate model at times. However, it is possible for advantages such as the continuity of the search space and the low computational cost of search to permit interesting solutions. Furthermore the *CNN* generates a pin-for-pin solution, meaning that a suitably trained



(a) Selected (6/58), *NDF LPs* generated by a *CNN SMO* for an SMR (seed=1454).



(b) Selected (6/14), *NDF LPs* generated by the *DSO* for an SMR (seed=3454321).

Fig. 5.20 Examples for *DSO* and *SMO LPs*, minimising *PPF* and enrichment.

CNN could potentially break the in-core out-of-core fuel management separation, allowing pin-for-pin full core optimisation.

Experiment 2 (p. 87) describes an example where the *SMO* is able to perform similarly and in some cases outperform the *DSO*. Furthermore, due to the low computational cost, it is trivial to increase the population, and run the *SMO* for more generations. This creates a *NDF* that is more space-filling and generates solutions that are comparable to the solutions found by *DSO* runs. A significant results of this experiment is that *SMO* can enable more exploratory investigation of a search space.

Experiment 3 (p. 93) demonstrates a model delivering non-dominated solutions for burnup simulations. In this case the computational advantages of the surrogate model become even more clear. The computational cost of the burnup *DSO* is 4000 cpu hours, while the *SMO* is achieved two million times faster on inferior hardware.

Experiment 4 (p. 95) generates a surrogate model of a Nuscale type SMR. This demonstrates the technique in a reactor that is being designed today. Although the *SMO* performs markedly less well than the *DSO*, the *SMO* still weakly dominates, generating non-dominated results that were not found by the *DSO*.

The value of Experiments 2 and 4 is that it is shown that, for the seed values investigated, the *SMO* is able to generate non-dominated solutions many tens of thousands of times faster than by direct simulation, whereas Experiments 1 and 3 demonstrate that the *SMO* can generate direct

simulation results that outperform the initial population of the *DSO* while being significantly more exploratory that the *DSO*. It should be noted that across the experiments most of the deep learning *SMO* results generate *NDFs* with more solutions, the cause of this is not known. It may imply that the surrogate search space is more smoothly graduated, making space filling results more common in the surrogate *NDF*. However, it could also be a product of the stochastic noise found in the *DSO*, further research on this may generate more insights.

A deep learning surrogate model improves in performance with the number of training examples. There is, therefore, an argument that the development of the training set is too expensive to justify the development of a surrogate model and that the benefits are too small. In this chapter it has been shown that prediction error improves dramatically as the training set increase to 1000 (p. 83). This is around thirty percent of the cost of the optimised *DSO* used. Furthermore, the results seen in this chapter show that the surrogate models are routinely able to produce results that weakly dominate the *DSO* (that some points on the *SMO NDF* outperform the *DSO NDF*). This means that using an *SMO* provides insights that could otherwise be missed by repeating the *DSO* for a given computational budget.

NSGA2 can operate with a high degree of parallelisation. It is possible to evaluate every member of the population independently for a given generation before the results are used to generate the next generation. However, the training set is *trivially parallelisable*; that is to say that the entire set can be submitted to independent processors and evaluated without data from other simulations being required. This means that if a suitably powerful HPC is available, then generating the training set will be N times faster in terms of *wall-clock time* than running the *DSO* at maximum parallelisation. This is for a *DSO* such as NSGA2, that can be fully parallelised per generation. However, many algoirthms such as MOEA/D involve a concept of neighbourhood that further reduces the possible parallelism. Moving forwards, it would be interesting to establish if the results of the *SMO* can be used as the initial population of the *DSO*, effectively reducing the number of generations that the *DSO* has to run for, potentially significantly reducing the number of *DSO* generations required. If the final population of the *SMO* can be used as as the initial population of a shorter *DSO*, then an augmented simulation that benefits both from the computational acceleration of the *SMO* process and the reliability of the *DSO* may be achieved.

Figure 5.21 shows the relative cost of running a *DSO* vs a *SMO*. Although there are more decisions for an *SMO*, the worst case scenario is that the *SMO* costs around 34% of the cost of executing the *DSO*, while subsequent runs of the *SMO* will cost less than 0.85% of the cost of executing the *DSO*.

Furthermore, as discussed in Section 1.1.1 in a real *LWR*, such a process must be carried out many times during the operational life, and it is highly unlikely that only one optimisation will

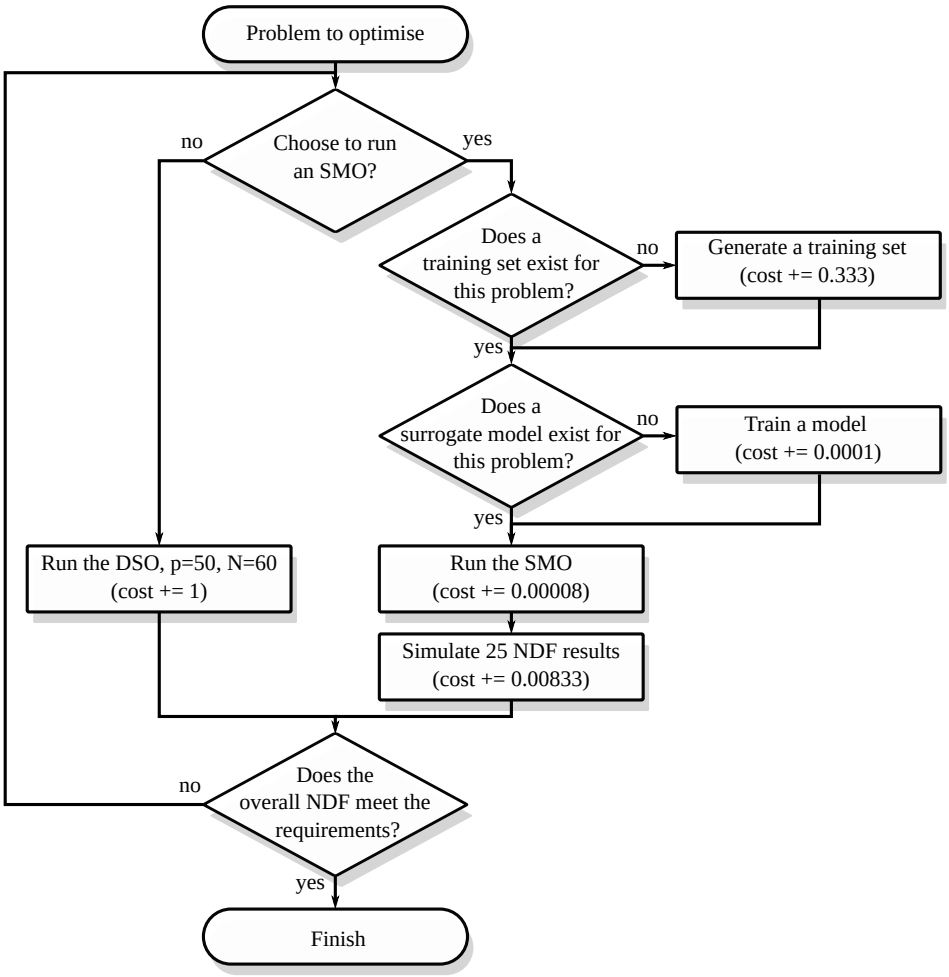


Fig. 5.21 A flow diagram demonstrating the relative computational costs of *SMO* and *DSO*.

be carried out at each refuelling, since multiple trade-off scenarios will need to be considered. These factors create a stronger justification for the generation of a training set to carry out *SMO*. If the optimisation is going to be carried out on a regular basis, then *DSO* becomes analogous to renting a house – in that a solution is found but the cost of the next optimisation is the same, whereas building a *SMO* is analogous to buying the house since, after an initial investment, computational cost can be spent operating a *DSO* closer to the *NDF*. In this chapter it has been shown that the 'house' can be built for around the same cost as one instalment of 'rent'.

The initial investment of generating a training set is justified by the very fast investigation of novel objective functions – to generate alternative solutions that may be missed by the *DSO* or to seed a *DSO* with promising solutions. As the 'open data' trend (see p. 4) becomes more common in the nuclear industry; larger, curated, training sets than those generated as part of

this work will inevitably become available to industry and researchers. These can then be used to beneficially augment the process of fuel loading pattern optimisation with *SMO* techniques.

In the next two chapters, other metaphorical, surrogate methods to accelerate optimisation strategies are investigated. Firstly the Fission matrix, then a matrix encoding of Galperin's heuristic rules are evaluated using a quantum annealer. [2]

[2] Code used in this section is available for audit, reproducibility and derived works. A copy can be obtained from the repository under the permissive two-clause Berkeley Standard License [3]:

https://bitbucket.org/ajw287/chapter5-deep-learning.git

Chapter 6

Fission Matrix Loading Pattern Model

6.1 Introduction

In this chapter, a surrogate model is derived from the fission matrix. The rationale for this model is to create a surrogate model that competes in computational efficiency with the deep learning models used in Chapter 5, but that has a basis in the simulations. In order to do this a fairly radical approach is taken: the fission matrix for a loading pattern is reconstructed from component matrices for uniform loading patterns. Although the method was developed independently to it, the method bears some similarities to the hybrid method proposed by Raunand and Williams [187], which applied a similar method to changes in temperature in reactors.

The fission matrix is, in essence, a statistical model of neutrons. Each entry in the matrix represents the probability that a neutron moves from its origin to a fission event in a region of interest. Like other *SMO* methods, the fission matrix model is generated from 'training' data. Running a Monte Carlo simulation, while recording the frequency with which neutrons are born in one region and die in a fission reaction in another, makes it possible to estimate the probability of these events. The first eigenvector of the fission matrix is proportional to the amount of fission energy generated in each region [31].

Relatively recent studies by Carney *et al.* [31, 30] have sparked renewed interest in fission matrix approaches, with applications to 'source convergence' (the pre-estimation of k_{eff} for the purposes of reducing the number of inactive cycles in Monte Carlo simulations). In their 2014 paper, Carney *et al.* comment that the fission matrix form of the neutron transport equation has been known since the first days of the Monte Carlo method and has been applied many times.

The suitability of the fission matrix as a surrogate model was evaluated using three experiments. The first experiment evaluates predictions of the microcore fission rate on a quarter assembly resolution for a number of fixed *LPs*, the second experiment uses the model developed

in the first experiment as a surrogate model for optimisation of *PPF* and enrichment, and the third experiment looks at *PPF* and cycle length using a simple fission-matrix based surrogate model to predict cycle length.

6.2 Fission matrix for the 6×6 Microcore

For the purposes of the experiments, the nine-assembly microcore introduced in Section 4.4 is used. In this chapter, each quadrant of each assembly is considered independently. This is common when considering the burnup of assemblies as rotation is limited to four angles due to the symmetry of square fuel assemblies used in most *PWR* designs.

Figure 6.1 shows an example of a fission matrix for the bottom right-hand quadrant of the microcore, with four regions per assembly, and thirty six entries per region (corresponding to the fission caused in the output region). In this case, each row of the fission matrix covers half of a row of assemblies, so the top quadrants of assemblies 1-3 are considered first with respect to the fission rate in every other region, then the lower quadrants of assemblies 1-3 and so on. The fission matrix shows strong diagonality as neutrons are most likely to cause fissions close to where they are born. In this case, the rotational symmetry creates a connectivity from assembly 2 to 4, and 3 to 7; this can be seen as off-diagonal artefacts in the matrix.

The matrix also has good diagonal symmetry since it is reasonable to assume isotropic scattering in a *PWR*, that the neutrons have no mean angular neutron flux [249, p. 51]. In other words, a neutron is equally likely to travel from *A* to *B* as from *B* to *A*.

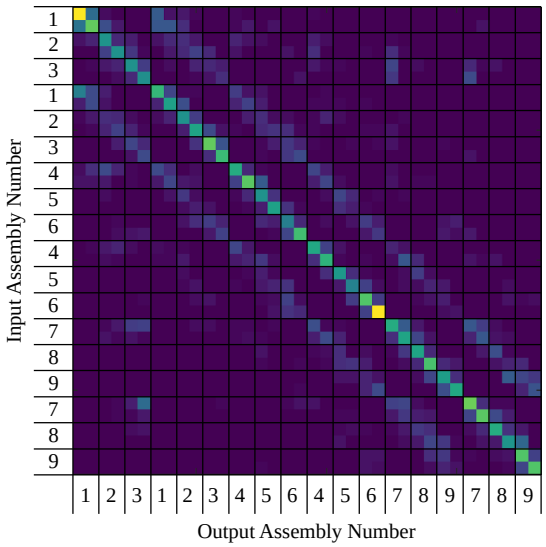


Fig. 6.1 The fission matrix for a quadrant of the 36 assembly microcore.

6.3 A Fission Matrix Surrogate Model

In the Serpent Monte Carlo code [128], the fission matrix can be generated by defining the correct mesh for the tallying of neutron events. In order to generate a high-integrity fission matrix from a Monte Carlo simulation, a large number of neutrons are required. $800,000$ neutrons were used in these experiments, which was an amount that was found to yield a fission matrix that was reasonably diagonally symmetric, while retaining computational benefits for three enrichments. The noise level of the fission matrix was established using visual inspection and confirmed with a short study in Appendix B, Section B.5.

The eigenvalue form of the neutron transport equation, called the forward fission matrix equation by Brown et al. [28] is shown in Equation 6.1, where $\vec{S}_t$ represents the neutron source at some time t, K is the effective multiplication factor and F is the *fission matrix*.

$$\vec{S}_{t_1} = \frac{1}{K} F \vec{S}_{t_0} \tag{6.1}$$

If the system is in steady state then $\vec{S}_{t_1} = \vec{S}_{t_0}$ and $\vec{S}$ can be seen to be an eigenvector and K an eigenvalue of F, so the per-region fission rate is then approximately the first eigenvector of F.

$$S_i = \frac{1}{K} \sum_{j=1}^{N} F_{i,j} S_j \tag{6.2}$$

Each entry in the fission matrix is described by Carney et al. [31] as the "fission rate in spatial region i due to average neutron born in region j" .

$$F_{i,j} = \frac{F_N s(\vec{r})}{F_D s(\vec{r})} = \frac{\text{fission rate in region } i \text{ due to fission source in region } j}{\text{fission source in region } j} \tag{6.3}$$

The surrogate model hinges on generating F_N and F_D, then separating them and reconstituting them on a per-assembly basis.

In this chapter, the model is generated by separating the F matrix from a number of generated fission matrices, then separating the F_D vector for each enrichment by taking the sum of the rows. The F_N matrix is then generated from the mean of the training matrices. This assumes that there is the same neutron flux density shape and *coupling* over the range of U235 enrichment in question. Although this is acknowledged to be a large approximation, the surrogate model does not need to be an accurate model of the system. Instead, the surrogate model informs the optimisation algorithm which direction to move in the search space. Thus, *it is sufficient for the surrogate model to correctly estimate the sign of the gradient of output variables in the region of the search space that the search trajectory crosses.*

It is known that the absorption cross-sections of U235 and U238 differ, which will have knock-on effects to the connectivity of the regions in the fission matrix. However, since the isotopic concentration of U235 is only changing from 1.6w/o to 3.2w/o, the change of U238 (from 98.4w/o to 96.8w/o) is small, the coupling is assumed not to be affected. This assumption allows the creation of a single F_N matrix of the connectivity of the regions. The effects of discretisation are discussed in [31]; a discretisation level of a quarter assembly was chosen based on this study and to permit further work with fuel shuffling. The method described in this section is clarified by way of a simplified example.

Example: The fission matrix surrogate model

The fission matrices used in this chapter are calculated using 6×6 regions, which generates a 36×36 fission matrix. It is impractical to write this out numerically, but for the purposes of a worked example, the same core is considered with 2×2 regions, to demonstrate the process used to generate the surrogate model for novel loading patterns.
First, fission matrices for three enrichments are generated from a Monte Carlo simulation for the geometry; these are 4×4 matrices, $F^{(enrichment)}$. The F_D vector is generated by taking the row-sum of the fission matrix, as shown for the example enrichments below:

Uniform 1.6 w/o U235:

$$F^{(1.6)} = \begin{pmatrix} 0.157 & 0.042 & 0.016 & 0.005 \\ 0.041 & 0.133 & 0.034 & 0.005 \\ 0.016 & 0.032 & 0.133 & 0.036 \\ 0.005 & 0.005 & 0.033 & 0.136 \end{pmatrix} \quad , \quad F_D^{(1.6)} = \begin{pmatrix} 0.219 \\ 0.212 \\ 0.217 \\ 0.182 \end{pmatrix}$$

Uniform 2.4 w/o U235:

$$F^{(2.4)} = \begin{pmatrix} 0.177 & 0.048 & 0.017 & 0.005 \\ 0.047 & 0.151 & 0.038 & 0.005 \\ 0.017 & 0.038 & 0.149 & 0.039 \\ 0.006 & 0.005 & 0.038 & 0.156 \end{pmatrix} \quad , \quad F_D^{(2.4)} = \begin{pmatrix} 0.247 \\ 0.242 \\ 0.243 \\ 0.205 \end{pmatrix}$$

Uniform 3.2 w/o U235:

$$F^{(3.2)} = \begin{pmatrix} 0.191 & 0.051 & 0.019 & 0.006 \\ 0.050 & 0.163 & 0.042 & 0.006 \\ 0.019 & 0.040 & 0.161 & 0.043 \\ 0.006 & 0.005 & 0.041 & 0.171 \end{pmatrix} \quad , \quad F_D^{(3.2)} = \begin{pmatrix} 0.266 \\ 0.259 \\ 0.262 \\ 0.227 \end{pmatrix}$$

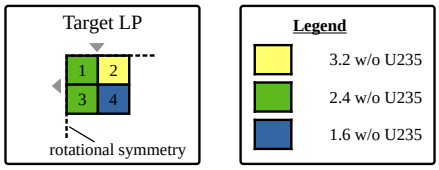
Equation 6.3 is then used to create connectivity matrix F_N, by dividing each element by the corresponding row of F_D.

$$F_N^{(1.6)} = \begin{pmatrix} 0.157/0.219 & 0.042/0.219 & 0.016/0.219 & 0.005/0.219 \\ 0.041/0.212 & 0.133/0.212 & 0.034/0.212 & 0.005/0.212 \\ 0.016/0.217 & 0.032/0.217 & 0.133/0.217 & 0.036/0.217 \\ 0.005/0.182 & 0.005/0.182 & 0.033/0.182 & 0.136/0.182 \end{pmatrix}$$

Based on an assumption that the connectivity is not changed by changing the enrichment, the element-wise mean of the F_N is used for novel *LPs*.

$$F_N^{(mean)} = \begin{pmatrix} 0.043 & 0.011 & 0.004 & 0.001 \\ 0.011 & 0.036 & 0.009 & 0.001 \\ 0.004 & 0.009 & 0.036 & 0.008 \\ 0.001 & 0.001 & 0.009 & 0.032 \end{pmatrix}$$

Using this data, a novel *LP* can be approximated by the following steps:



The approximated fission matrix $\hat{F}$ for the target *LP* is generated as follows:

$$\hat{F}^{(target)} = \begin{pmatrix} F_{N\ 1,1} \cdot F_{D,1}^{(2.4)} & F_{N\ 1,2} \cdot F_{D,1}^{(2.4)} & F_{N\ 1,3} \cdot F_{D,1}^{(2.4)} & F_{N\ 1,4} \cdot F_{D,1}^{(2.4)} \\ F_{N\ 2,1} \cdot F_{D,2}^{(3.2)} & F_{N\ 2,2} \cdot F_{D,2}^{(3.2)} & F_{N\ 2,3} \cdot F_{D,2}^{(3.2)} & F_{N\ 2,4} \cdot F_{D,2}^{(3.2)} \\ F_{N\ 3,1} \cdot F_{D,3}^{(2.4)} & F_{N\ 3,2} \cdot F_{D,3}^{(2.4)} & F_{N\ 3,3} \cdot F_{D,3}^{(2.4)} & F_{N\ 3,4} \cdot F_{D,3}^{(2.4)} \\ F_{N\ 4,1} \cdot F_{D,4}^{(1.6)} & F_{N\ 4,2} \cdot F_{D,4}^{(1.6)} & F_{N\ 4,3} \cdot F_{D,4}^{(1.6)} & F_{N\ 4,4} \cdot F_{D,4}^{(1.6)} \end{pmatrix}$$

$$\hat{F}^{(target)} = \begin{pmatrix} 0.043 \cdot 0.266 & 0.011 \cdot 0.266 & 0.004 \cdot 0.266 & 0.001 \cdot 0.266 \\ 0.011 \cdot 0.242 & 0.036 \cdot 0.242 & 0.009 \cdot 0.242 & 0.001 \cdot 0.242 \\ 0.004 \cdot 0.243 & 0.009 \cdot 0.243 & 0.036 \cdot 0.243 & 0.008 \cdot 0.243 \\ 0.001 \cdot 0.182 & 0.001 \cdot 0.182 & 0.009 \cdot 0.182 & 0.032 \cdot 0.182 \end{pmatrix}$$

$$\hat{F}^{(target)} = \begin{pmatrix} 0.01062 & 0.00271 & 0.00111 & 0.00026 \\ 0.00279 & 0.00862 & 0.00241 & 0.00025 \\ 0.00105 & 0.00211 & 0.00774 & 0.00147 \\ 0.00035 & 0.00028 & 0.00198 & 0.00578 \end{pmatrix}$$

The primary eigenvector (corresponding to the largest eigenvalue) $\vec{S}$, is proportional to the fission rate per region. In this case:

$$\vec{S} = \begin{pmatrix} 0.700 \\ 0.588 \\ 0.377 \\ 0.149 \end{pmatrix}$$

Rearranging the eigenvector according to the regions of the discretisation of the fission matrix gives the estimated relative map of fission rates.

$$Target\ relative\ fission\ map,\ \hat{T} \propto \begin{pmatrix} 0.700 & 0.588 \\ 0.377 & 0.149 \end{pmatrix}$$

Experiment 1 (Section 6.4) investigates the effectiveness of this model for a number of arbitrary 6×6 region fuel *LPs*. The source code is also made available for investigation of the surrogate modelling process (instructions for obtaining the code are found in Section 6.8).

6.4 Experiment 1: Powermap Prediction

A simple experiment with the fission matrix model, which aims to confirm that the fission rate of regions in arbitrary loading patterns can be predicted by the method developed in Section 6.3.

6.4.1 Method

The first experiment aims to establish a model that is able to predict the power maps of a number of *LPs*, in order to evaluate whether this approach is feasible as the basis of a surrogate model for optimisation. A fission matrix is generated for each of the 'A:Training' *LPs* of Figure 6.2, then the normalised power map is compared with the first eigenvector of the generated fission matrix. Due to the very high granularity required of the fission matrix, very large populations were required in order for the resultant fission matrix to exhibit a low level of noise. The populations used in these simulations are described in Table 4.4.

By generating fission matrices for three training enrichments it becomes possible to generate an approximation to a fission matrix $\hat{F}$ for a previously unseen arbitrary *LP* made up of these three enrichment values. For example, the *LPs* 'B:Test' are loading patterns chosen for the experiment. The process for generating the surrogate model fission matrices has been carried out as described in the worked example on p. 108.

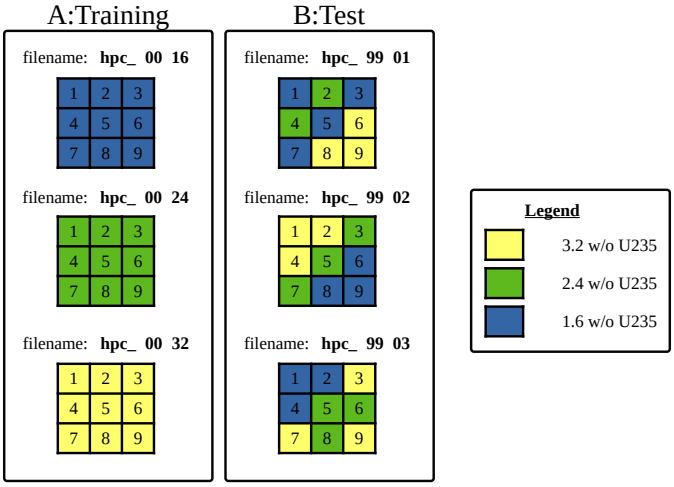


Fig. 6.2 Cores used for training and validating the fission matrix surrogate model

This method of generating the 'model fission matrix' is very crude. Assumptions include: the linkage of the F_D term depending only on the geometry. In reality, the value of F_D is dependent on the cross-section of the target cell as well as the geometry of the system. Furthermore, the source term is assumed to be completely linked to enrichment, which is untrue, since the source is a product of the fissile material (which is only the enrichment at BOL) and the neutron flux spectrum. This model implicitly assumes that the flux spectrum is the same as the constituent fission matrices. Therefore, as the target core power map diverges from the power maps of uniform cores used as 'training' data, it can be expected to perform less well. Despite these caveats, this experiment investigates a number of *LPs* to test if the model may be a *good enough* model for the optimisation to proceed.

6.4.2 Results

In the first two columns of Figure 6.3, the first fission matrix eigenvector is shown on a relative colour-map. Then the actual power map for each uniform core is then depicted. The first eigenvector of the fission matrix is known to be proportional to the magnitude of fission reactions occurring in each region. The second column shows the actual total fission power calculated by the Monte Carlo simulation. The next three columns show the eigenvector,

a ‘generated’ result and the actual power map. The generated result was created using the decomposed fission matrix from the three uniform simulations to create a source and geometry term for each position and enrichment. It is then possible to generate a ‘model matrix’ whose eigenvectors are shown in the ‘generated’ column. Visual inspection shows that they follow the fission matrix and power map to some extent. The results in Table 6.1 were generated using the SciPy [225] implementation of the Pearson’s product-moment correlation coefficient (described in [212, p. 53]). From the table it can be seen that there is a statistically significant result for all of the tests, with two of the three tests showing a high ($> 95\%$) likelihood of the ‘null hypothesis’ being true – that the model is correlated with the results. These results do not prove the null hypothesis, but do provide evidence to support it.

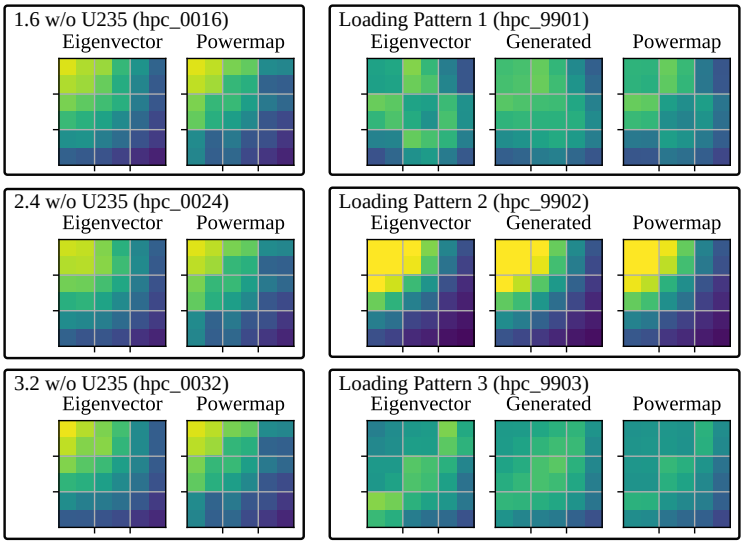


Fig. 6.3 The fission matrices from uniform distributions are split into geometry and source terms and are then used to reconstitute a ‘surrogate fission matrix’ eigenvector.

Table 6.1 Correlation tests for power map to fission matrix and to surrogate model (to *6 s.f.*)

	Fission Matrix		Surrogate Model	
	Pearson’s R	p-value	Pearson’s R	p-value
hpc_0016	0.959097	0.00247531	–	–
hpc_0024	0.972104	0.00115643	–	–
hpc_0032	0.979600	0.000620010	–	–
hpc_9901	0.972368	0.00113472	0.984934	0.000338787
hpc_9902	0.999314	7.04828E-07	0.998811	2.12050E-06
hpc_9903	0.810218	0.0506083	0.725750	0.102506

6.4.3 Limitations of a Fission Matrix Model

The fission matrix model created in this experiment is most accurate when it is predicting a system with a flux shape that is similar to the original flux shape. This is because the F_D of a given assembly is based both on the amount of fissile isotopes in the assembly (proportional to enrichment in this experiment) and the flux in that assembly. The model used here varies the source of neutrons caused by enrichment, but not the multiplicative aspects of the resultant increase. However, a highly simplified approximation could still have the right sign of gradient, even after neglecting the multiplicative effects of neutron chain reactions. Thus, it may still be of use, since the surrogate model is being used only for navigating the search space rather than for collecting meaningful results.

The results of the correlation calculation in Table 6.1 are promising. Two of the three test patterns have correlation p-values less than 0.05%, where 5% is a typical value for accepting the null hypothesis [95, p.832]. However, the last model (where it is expected that there is a significant neutron flux at the outside) yields a p-value greater than 10.0%. Whether this is sufficient similarity for a surrogate model to *point in the right direction* is the subject of the next experiment.

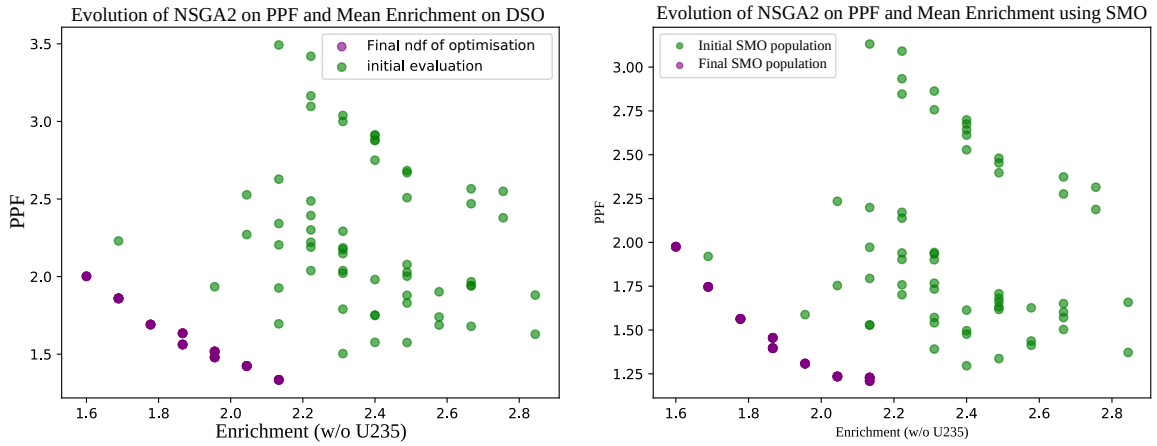
6.5 Experiment 2: BOL PPF and Mean Enrichment

6.5.1 Method

This experiment proceeded similarly to the one in Section 5.5 of Chapter 5: a *DSO* was run and compared to a run of the *SMO*. An important difference is that, due to the creation of fission matrices for only three enrichments, the optimisation is limited to these enrichment values. In order to make the comparison fair, the *DSO* has the same requirement. The optimisation algorithm used for both the *SMO* and *DSO* is NSGA2 [50], implemented in Pygmo [92]. The hyperparameters of the algorithm are unchanged and found in Table 5.1.

6.5.2 Results

The results depicted in Figure 6.4 show the initial and final populations of *DSO* and *SMO* runs. Visual inspection of the (identical) initial populations shows that the *SMO* generates numerically inaccurate results, but that the overall shape of the results displays some similar features to the direct simulation evaluation. Side-by-side comparison of the *NDFs* (Figure 6.4) shows that the *DSO* generates a result that appears to compete with the *SMO* evaluation. However, when the *SMO* results are evaluated in Serpent, the results for the *SMO* are strictly dominated in this case.



(a) DSO, $\sim$ 200 CPU hrs, (HPC) (b) Fission matrix SMO, $<$ 0.045 CPU hrs, (laptop)

Fig. 6.4 Examples of initial populations and final *NDFs* for *DSO* and *SMO*.

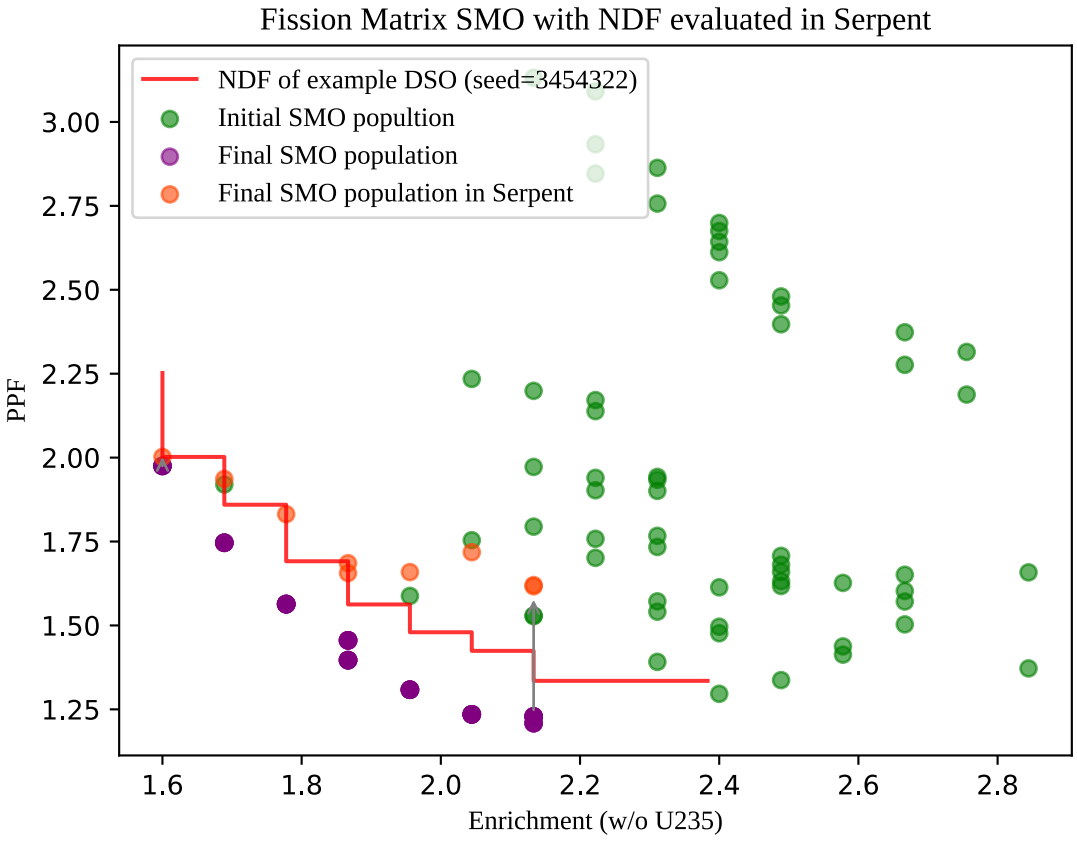


Fig. 6.5 *NDF LP* for the fission matrix *SMO* evaluated in Serpent.

The surrogate model underestimates the *PPF* in every case at the *NDF*. Analysis of the actual

Table 6.2 Execution time for fission matrix model and *DSO*

	CPU time (s)	**machine**[a]
Fission matrix generation[b]	5.06×10^5	Lise
Fission matrix SMO	132.7	laptop
Fission matrix burnup SMO	495.9	laptop
DSO	$\sim 3.6 \times 10^6$	HPC

[a] for machine descriptions see Table D.1, [b]per value of enrichment investigated.

LPs generated by these runs (Figure 6.6) shows some structural similarities and differences in the evolved solutions. Both optimisations find the trivial, lowest enrichment solution and there appears to be a preference towards higher enrichments at the edge for both of the other *NDF* sets of solutions. The central four enrichments are 1.6 w/o U235 in all solutions in both *NDFs*. Figure 6.6a shows the *LPs* that result from the *SMO*, while Figure 6.6b shows the *LP* results for the example run of the *DSO*. It should be noted at this stage that the results here are for single runs of the algorithms. In order to make less anecdotal observations of the trends in *LPs* from each method, more runs would need to be carried out or the experiment could be repeated for a geometry that is possible in a deterministic solver such as WIMS [135]. Due to the extreme computational cost of the *DSO*, and the limited access to resources that could run WIMS (see Section 4.2.3), this has not been possible. The computational cost of generating parts of the fission matrix based *SMO* are shown in Table 6.2. It can be seen that in order to achieve a high fidelity fission matrix, a large computational cost must be expended. The fission matrix *SMO* does not contribute a significant amount to the computational budget, so it is considered acceptable that this has been implemented in a high level language and not been optimised for efficiency. For more complex problems with variables such as burnable poisons or more values of enrichment, more fission matrices may be required. Although generation of a number of fission matrices is more parallelisable than a *DSO*, it is likely to become uncompetitive in terms of total computational cost if many combinations of fission matrices were required.

6.6 Experiment 3: PPF vs Cycle Length

In this experiment the fission matrix model is used to generate BOC fission rates per region. These are used to estimate the cycle length in a simple burnup model that considers U235 and conversion of U238 to Pu239.

k_{eff} :	1.00361	1.02122	1.02793	1.01790	1.02270	1.02510	1.03893	1.04176	1.03702
mean enrichment :	1.60	1.69	1.78	1.87	1.87	1.96	2.04	2.13	2.13
PPF :	2.10	1.94	1.83	1.69	1.66	1.66	1.72	1.62	1.62

(a) *NDF LPs* generated by a fission matrix *SMO* evaluated in Serpent for the bottom right-hand quadrant of the microcore (seed=3453412).

k_{eff} :	1.00361	1.01297	1.01828	1.01794	1.02802	1.03097	1.02827	1.03488	1.03097
mean enrichment :	1.60	1.69	1.87	1.78	1.96	2.04	1.87	1.96	2.13
PPF :	2.00	1.86	1.64	1.69	1.52	1.42	1.56	1.48	1.33

(b) *DSO LPs* generated for the bottom right-hand quadrant of the microcore (seed=3453423).

Fig. 6.6 Examples for *DSO* and fission matrix *SMO LPs*.

6.6.1 Method

In this experiment the *DSO* is compared to a simple burnup model using the fission matrix eigenvalue solution as the basis for the calculation. Details of the direct simulation and surrogate model are described below.

Burnup Simulation in Serpent

The experiments in this section use Serpent's stand-alone burnup calculation. The power density is set to 40 kW/kgU and the fuel is depleted using 20 cumulative depletion steps, based on the example burnup code for a *PWR* assembly [127]. The burnup steps are at 0.1, 0.5, increments of 1 from 1-10, increments of 2.5 from 12.5-20, and increments of 5 from 25-40 MWd/kgU.

From the Serpent simulation, the cycle length is estimated by taking a linear interpolation in terms of k_{eff} and burnup steps at the step where k_{eff} goes below unity.

A Simple Surrogate Burnup Model

The simplest model of cycle length considers only the number of fissile nuclei in the reactor at BOC and the rate of utilisation, such as in the description by Stacey [208, pp. 219–221]. However, fertile isotopes are converted to fissile isotopes as neutrons interact with them. These experiments use only Low Enriched Uranium (LEU) – uranium with an isotopic *enrichment* below 5% U235. The major fissile isotope produced in LEU through neutron capture in U238 and two successive β-decays is Pu239 [18, p. 60]. This conversion creates the possibility of

extending the fuel cycle beyond the amount of U235 originally loaded into the core. Considering only U235 and Pu239 as fissile isotopes, and U238 as a fertile isotope, Lewis [129, pp. 252-255] generates the following burnup model:

$$\frac{d}{dt}N^{U235}(t) = -N^{U235}(t)\sigma_a^{U235}\phi(t) \tag{6.4}$$

$$\frac{d}{dt}N^{U238}(t) = -N^{U238}(t)\sigma_a^{U238}\phi(t) \tag{6.5}$$

$$\frac{d}{dt}N^{Pu239}(t) = \sigma_\gamma^{U238}\phi(t)N^{U238}(t) - \sigma_a^{Pu239}N^{Pu239}(t) \tag{6.6}$$

Due to the high proportion of U238 compared to the other isotopes, the N^{U238} is considered constant.

$$N^{U238}(t) \approx N^{U238}(0) \tag{6.7}$$

Equation 6.6 then simplifies to

$$\frac{d}{dt}N^{Pu239}(t) = \sigma_c^{238}\phi(t)N^{238}(0) - \sigma_a^{Pu239}N^{Pu239}(t) \tag{6.8}$$

Integrating Equation 6.4 and Equation 6.8 gives:

$$N^{U235}(t) = N^{U235}(0)e^{-\sigma_a^{U235}\Phi(t)} \tag{6.9}$$

$$N^{Pu239}(t) = \frac{\sigma_c^{238}}{\sigma_a^{Pu239}}N^{238}(0)\left(1 - e^{-\sigma_a^{Pu239}\Phi(t)}\right) \tag{6.10}$$

The fluence is defined by Equation 6.11.

$$\Phi(t) = \int_0^t \phi(t')dt' \tag{6.11}$$

A number of simplifications are now required for the fission matrix model to be used. It is assumed that the relative BOC fission rate obtained from the surrogate model $\hat{F}$ in region j is related to the flux ϕ according to Equation 6.12. In fact, the flux must increase as the cycle proceeds due to the reduced inventory of fissile material and the accumulation of fission products that absorb neutrons.

For simplicity, the fission rate per region is assumed to stay constant. Which permits the fixed case flux and fluence to be calculated using Equations 6.12 and 6.13, respectively. Here P_D is the power density and BU_{core} is the total burnup for a 2D (equivalent to 1cm^3 deep)

simulation of the core at the time of interest:

$$\phi_j = \frac{P_D \, \hat{T}_j}{E_f \Sigma_{f,j}} \tag{6.12}$$

$$\Phi_j(t) = \frac{BU_{core}(t) \, \hat{T}_j}{E_f \Sigma_{f,j}} \tag{6.13}$$

This allows the objective function for the surrogate model to be to minimise Equation 6.14, which is proportional to the number of fissile atoms present. It is known from Section 5.6 that a cycle length of 10 MWd/kU is a possible for the microcore. So, by maximising the total inventory of fissile material for this amount of burnup, a surrogate objective might be found for increasing the cycle length.

$$-N^{fissile} \simeq -1 \times \left(\sum_{j=1}^{36} N_j^{U235}(0) e^{-\sigma_a^{U235}\Phi_j} + \frac{\sigma_\gamma^{238}}{\sigma_a^{Pu239}} N_j^{238}(0) \left(1 - e^{-\sigma_a^{Pu239}\Phi_j}\right) \right) \tag{6.14}$$

This model ignores the position of the fissile inventory. In reality, the core may be subcritical if all of the fissile inventory is too far apart for a neutron chain reaction to be sustained. This model does not consider the position of the fissile material in the core. The poisoning effects of fission products are also ignored.

Table 6.3 Fission matrix burnup model parameters for the four region-example, based on data obtained from Stacey [208, pp. 13,669-674] and calculated values

Nuclear Data			
σ_f^{U235} (barns)	577		
σ_a^{U235} (barns)	$101+577=678$		
σ_c^{U238} (barns)	2.73		
σ_f^{Pu239} (barns)	741		
σ_a^{Pu239} (barns)	$274+741=1015$		
Energy per fission (MeV)	200		
Calculated Values	**Enrichment (w/o U235)**		
	1.6	2.4	3.2
N^{U235} (cm^{-3})	3.79×10^{20}	5.69×10^{20}	7.58×10^{20}
N^{U238} (cm^{-3})	2.61×10^{22}	2.59×10^{22}	2.57×10^{22}
fuel volume per region (cm^3)	317.68		
$N(0)^{U235}$	1.2053×10^{23}	1.8080×10^{23}	2.4107×10^{23}
$N(0)^{U238}$	8.3034×10^{24}	8.2359×10^{24}	8.1684×10^{24}
Σ_f^{U235} (cm^{-1})	0.219	0.328	0.438
Σ_c^{U238} (cm^{-1})	0.0714	0.0708	0.0702
Total core fissions to 10MWd/kgU	3.17120×10^{23}		

Example: The fission matrix to surrogate burnup model

To illustrate the burnup method, the result from the example calculation on p. 108 is used as the basis for the calculation of the $N^{fissile}$ inventory after 10MWd/kgU, compared to the initial fissile inventory (the amount of U235 in the four regions). This example uses values from Table 6.3.

The initial fissile inventory is assumed to be equal to the number of U235 atoms in the fuel:

$$\begin{aligned} N^{fissile}(0) &= 1.81\times10^{23}+2.41\times10^{23}+1.81\times10^{23}+11.21\times10^{23} \\ &= 7.23\times10^{23} \end{aligned}$$

To calculate the fissile inventory at the end after 10MWd/kgU, the following steps are carried out. First, the relative eigenvector result, which has a vector magnitude of 1, should be scaled to

have an element sum of 1.

$$\hat{T} \propto \begin{pmatrix} 0.700 & 0.588 \\ 0.377 & 0.149 \end{pmatrix} \Rightarrow \begin{pmatrix} {}^{0.700}/_{1.814} & {}^{0.588}/_{1.814} \\ {}^{0.377}/_{1.814} & {}^{0.149}/_{1.814} \end{pmatrix} = \begin{pmatrix} 0.386 & 0.324 \\ 0.208 & 0.082 \end{pmatrix}$$

The fluence that corresponds to the fission map can be estimated according to Equation 6.13. Although the recoverable energy differs slightly between U235 and Pu239, a fixed value of of 200MeV per fission is used to estimate the fluence.

$$\Phi_{i,j} = \frac{3.17120 \times 10^{23} \times \hat{T}_{i,j}}{577 \times 10^{-24} \times N(0)_{i,j}^{U235}}$$

$$\Phi = \begin{pmatrix} 1.54 \times 10^{22} & 1.16 \times 10^{22} \\ 1.05 \times 10^{22} & 7.05 \times 10^{21} \end{pmatrix}$$

Then for a burnup of 10MWd/kgU, Equation 6.14 is used. The objective of a high fissile inventory is considered equivalent to a longer cycle length for the experiment:

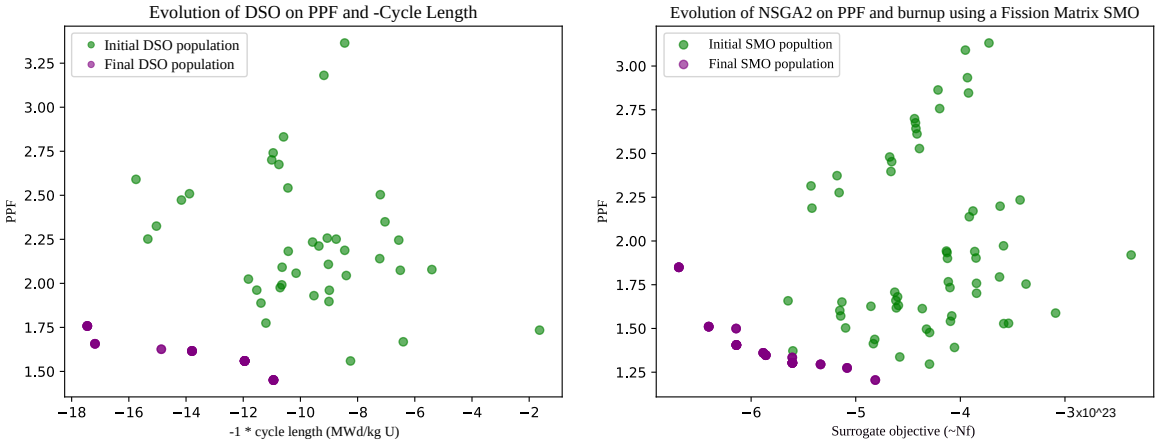
$$-N_{fissile}(BU_{core} = 10) = 4.81207 \times 10^{+23}$$

This model does not use the value of k_{eff} (the parameter that actually controls cycle length), which means that the fissile inventory might not be physically realisable. However, the aim is to optimise an objective that correlates with the cycle length. The geometry is not changing and the enrichment is only varied over a small range. Therefore, it assumed to be unlikely that large changes in the physical location of the fissile inventory due to conversion of U238 to Pu239 will occur.

6.7 Results

In a procedure that is now familiar, Figure 6.7 shows the side-by-side *DSO* and *SMO* results, and Figure 6.8 compares the *NDF* results of the *SMO* and the *DSO*. Figure 6.8, shows the results together, with the red line representing the *NDF* of the *DSO* and the orange circles representing the Serpent results of the *SMO*'s final *NDF*. Note that the surrogate model results are shown on the scales of PPF and $-N_{fissile}$, an estimate of the total fissile inventory at 10 MWd/kgU. The *DSO* results and the Serpent simulation of the *NDF* compares PPF and cycle length.

Figure 6.8 shows that the *NDF* of the *SMO* does not exhibit good correlation between predicted and directly simulated values for this seed value. Although one solution is shared between the *DSO* and *SMO NDFs*, the results do not readily form an interpretable trend. The *SMO* results appear to perform fairly closely to the *DSO* results, although the *DSO* strictly dominates the results. However, examination of the actual *LPs* generated by both the *DSO* and the *SMO* shown in Figure 6.9, shows that while many of the results appear be rational *LP* designs, a small portion of solutions have unexpectedly high, or low cycle lengths. This may be due to stochastic noise causing instability in the burnup simulation, if this is the case, then the *DSO* will actively select for outliers.



(a) DSO, $\sim$ 1500 cpu hrs, (HPC) (b) Fission matrix SMO, $<$ 0.083 cpu hrs, (laptop)

Fig. 6.7 Examples for *DSO* and *SMO* initial and final population results.

Stochastic Noise and Multiobjective Optimisation

As previously considered in Section 4.2.3, stochastic noise causes problems for iterative multiobjective optimisation algorithms. This is due to the fact that, if the *NDF* is not clearly defined, the population cannot easily converge on any one set of solutions. In this thesis, the approach has been to fix the seed value of the Monte Carlo simulation and to use appropriate simulation parameters to establish a low level of stochastic noise. However, it is possible for the *DSO* to find solutions that benefit artificially by having outlier stochastic noise for the seed value used, whereas the surrogate models do not have stochastic noise.

In July 2020, Rojas Gonzalez et al. [190] proposed the application of *SMO* as a significant technique for solving this problem. By optimising a stochastic-noise-free surrogate model, solutions closer to the true *NDF* can be established. Stochastic noise dependence is likely to be larger in the burnup calculations and this phenomenon might explain the increased performance

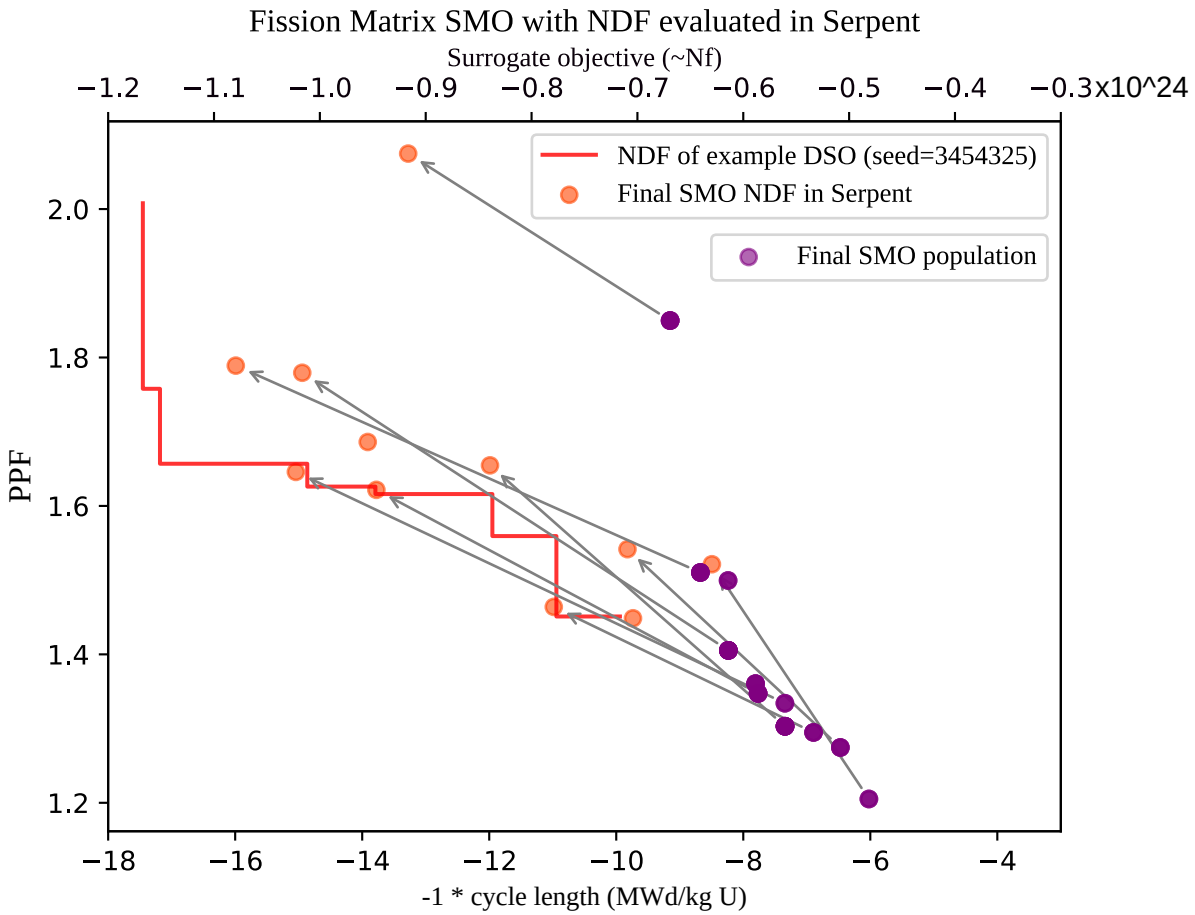


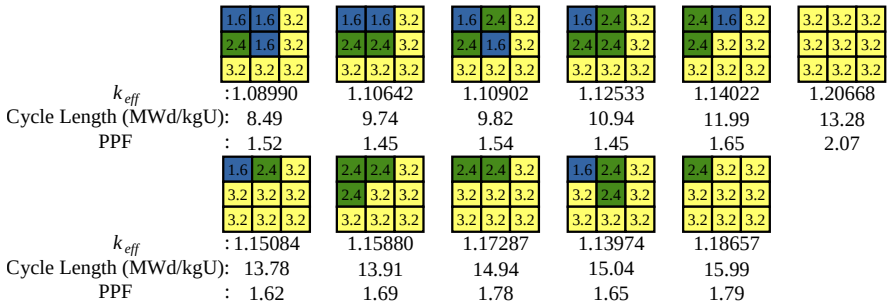
Fig. 6.8 *NDF LP* for the fission matrix *SMO* evaluated in Serpent.

of the *NDF* for the *DSO* in this experiment and Experiment 3 of Chapter 5 (p. 93). If this is correct, this results do not invalidate these experiments *for this seed*. However, the opportunity exists for further work to validate the true *NDF* using a deterministic code such as PANTHER [156].

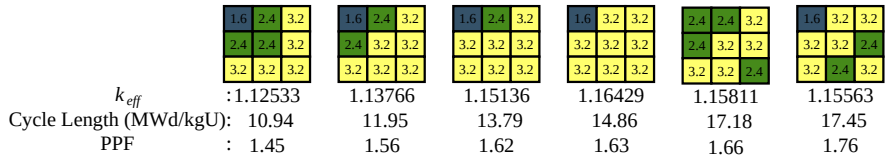
6.8 Discussion

The fission matrix has a number of attractive features as a surrogate model. Conceptually, it has the advantage of being an extremely simple model of neutron transport and is easily adaptable to the inputs of the simple problems of fuel loading in this chapter. It has a physical basis in the neutron transport equation, unlike the deep learning or Ising models[1], which allays the fear that the surrogate model may not reflect reality.

[1]The Ising model is the basis for Chapter 7 and is introduced in Section 7.2.1



(a) *NDF LPs* generated by a fission matrix *SMO* for the bottom right-hand quadrant of the microcore (seed=3453412).



(b) *NDF LPs* generated by *DSO* for the bottom right-hand quadrant of the microcore (seed=3453425).

Fig. 6.9 Examples of *DSO* and fission matrix *SMO LPs*,

However, the implemented model has a number of remaining problems. In order to develop a number of fission matrices, a high-fidelity simulation is required for each enrichment used. This has been feasible for three enrichments of U235 only, but would quickly become prohibitive under the combinatorial explosion of more realistic considerations, such as burnable poisons and fuel histories in a multi-batch setup. The computational cost of these experiments is summarised in Table 6.2.

The models used in these experiments appeared promising for the simple test patterns evaluated in Experiment 1, but delivered unsuccessful results when applied to optimisation problems for this implementation on these particular seeds. Significant simplifications have been made, as described in Section 6.4.3. However, it appears that even this simplified model allows for some level of transfer between the *SMO* and the *DSO*. The results presented here show that, for the seed values used, the *SMO* results are generally less performant than the example *DSO*, though it is not certain that the evolved *SMO NDFs* do not represent potentially useful solutions.

A significant advantage of this model is that it is possible to express the fission matrix equation in a form that can be shown to be equivalent to a Hamiltonian operator similar to the Ising model. This could allow the model to be optimised on experimental and theoretical quantum computers. Particularly, it has been proposed to develop this model for optimisation

on machines using ‘polaritons’ to locate the global optimum of an objective, as introduced by Berloff et al. [21]. The connection between *LP* optimisation and quantum computing will be investigated further in the next chapter. However, at this time the fission matrix was not chosen as a model of the system. [2]

[2] Code used in this section is available for audit, reproducibility and derived works. A copy can be obtained from the repository under the permissive two-clause Berkeley Standard License [3]:

https://bitbucket.org/ajw287/chapter6-fission-matrix.git

Chapter 7

Extended Work: Quantum Annealing Optimisation of a Heuristic Surrogate Model

In this chapter, as an extension of the work with surrogate models, a heuristic surrogate model is developed and executed, first on a simulated quantum annealer (SQA). Then, the model is exported to an actual DW2000Q machine, which its manufacturers claim operates as a true *QA* [2]. This demonstrates the application of present day quantum computers to solve nuclear fuel loading problems using a simplified metaphorical surrogate model.[1]

7.1 Introduction

From 1965 to the early 2010s, computing power increased exponentially, with the number of transistors per chip doubling every two years, a phenomenon predicted by Moore [155]. Enormous increases in computing power were achieved. Iterative optimisation as applied to the fuel loading problem coincides with this change. Research such as that by Parks [173, 174] and Ahn [7] coincides with, and utilises, this novel resource. However, the number of transistors on a chip has stopped following this trend despite increasing investment by the semiconductor industry in supporting it. In 2016, Waldrop commented:

> "Moore's law, the principle that has powered the information technology revolution since the 1960s, is nearing its end." [226]

[1] Work in Section 7.2 has been previously peer reviewed and accepted for the PHYSOR 2020 conference. [234] it is reproduced here with some modifications to make the subject more tractable in the context of the thesis.

Interest is now returning to alternative methods of computation. Some, such as the digital logic that forms the basis of *Field Programmable Gate Arrays (FPGAs)*, have existed since before modern digital computer architectures, while others, such as neural accelerator hardware, *Graphical Processing Units (GPUs)* and quantum processors, are contemporary inventions. The question now arises as to how to express nuclear engineering problems in a way that they can be solved by these novel computing architectures.

One vision of a future HPC architecture is shown in Figure 7.1. Here, three different alternative computing coprocessors – hardware neural accelerator such as a 'Tensor Processing Unit' (TPU) [98], *GPU* and quantum computer architectures are available to support a classic *CPU* node [23]. This chapter describes a method for expressing simple heuristic rules for loading pattern generation into a type of processing device called a quantum annealer. The encoding that is generated can be loaded in the simulated *QA* (that runs on a host *CPU*) or submitted to an actual *QA* device [2].

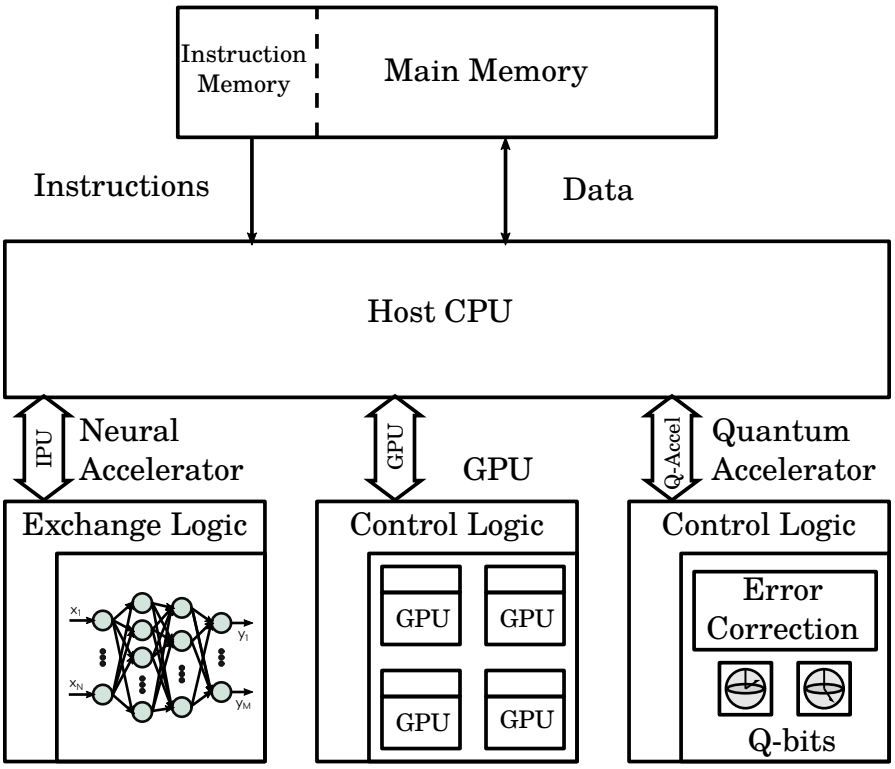


Fig. 7.1 A vision of a future HPC architecture, adapted from [23].

7.2 Adiabatic Optimisation using a Simple, Rule-based Surrogate

In 1982, Feynman [62] initiated the field of quantum computing by suggesting that the inherent complexity of quantum mechanics might be an efficient way to solve complex computational problems. Despite significant research effort, it is only in recent years that quantum computers have become available commercially.

In this section, heuristic rules, similar to those used in [71], will be encoded into a surrogate model suitable for use by a *QA* [64], a present-day quantum computer, to generate *PWR LPs*. *QAs* use a trick from quantum adiabatic theory, which allows a simple Hamiltonian to be converted into a complex one, while keeping the system in the minimum energy state. The following subsection describes how adiabatic theory permits the discovery of the global optimum of a system. This has been utilised in adiabatic quantum computing and implemented to some extent in *QAs*.

7.2.1 Adiabatic Quantum Computers and Quantum Annealing

Quantum adiabatic theory describes a technique for global optimisation. The theory applies to a Hamiltonian that evolves in time t, from a simple system to a complex one over a time period T, as described by:

$$H(t) = (1 - t/T)H_0 + (t/T)H_1 \tag{7.1}$$

H_0 describes a simple Hamiltonian shown in Eq. (7.2), and H_1 is an implementation of the problem that we are looking to solve. In order for our problem to be solvable by an adiabatic system, the problem must be expressed in terms of the classical Ising model, of the form shown in Eq. (7.3).

$$H_0 = -\sum_{i,j} \sigma_i^X \sigma_j^X \tag{7.2}$$

$$H_1 = -\sum_{\langle i,j \rangle} \mathscr{J}_{ij} \sigma_i^Z \sigma_j^Z - \mu \sum_i h_i \sigma_i^Z \tag{7.3}$$

where: H is a Hamiltonian, a function representing the total energy of the system; $\langle i, j \rangle$ signifies connected qubits i and j; $\mathscr{J}$ is an exchange constant, a matrix of strengths of connections between qubits attracting them to similar or opposite states; σ signifies logical qubits, which can have values -1 or $+1$ upon measurement; and h is a transverse field, an external field strength applied to each qubit that biases its state.

Quantum adiabatic theory states that, if a system like the one above is in its ground state and is evolved over time, then the system remains in its ground state *unless the rate of energy*

put into the system is enough to push it into a new state. While this may sound trivial, the implications are important. So long as a 'speed limit' for t is not exceeded, then the system will evolve from H_0 to H_1 while remaining in the global minimum energy state. Thus it is possible to find the global optimum of a problem. However, for an adiabatic quantum computer to optimise an objective function, it must be expressed in terms of the Ising model Hamiltonian, Eq. (7.3).

Furthermore, existing quantum computers operate at around 10 mK above absolute zero, which means that, although the system is expected to be in its ground state initially, this is not guaranteed. Finally, the 'speed limit' is given by Eq. (7.4), where Δ is the difference of energy between the ground state and other states. Unfortunately, it may be more difficult to find the difference between the global minimum and the second best minimum than to find the global minimum in the first place [240].

$$t_{limit} \approx \frac{1}{(min(\Delta) \times t)^2} \tag{7.4}$$

This leads to the implemented solution: quantum annealing. The system is the same, but the system evolves from H_0 to H_1 at a practical speed, ignoring the speed limit. The process is executed many times and the minimum solution found is taken to be the optimum. This allows the problem to be solved in a realistic amount of time, while still identifying the global optimum, assuming that at least one execution has evolved from the ground state of H_0 to the ground state of H_1.

7.2.2 Methodology

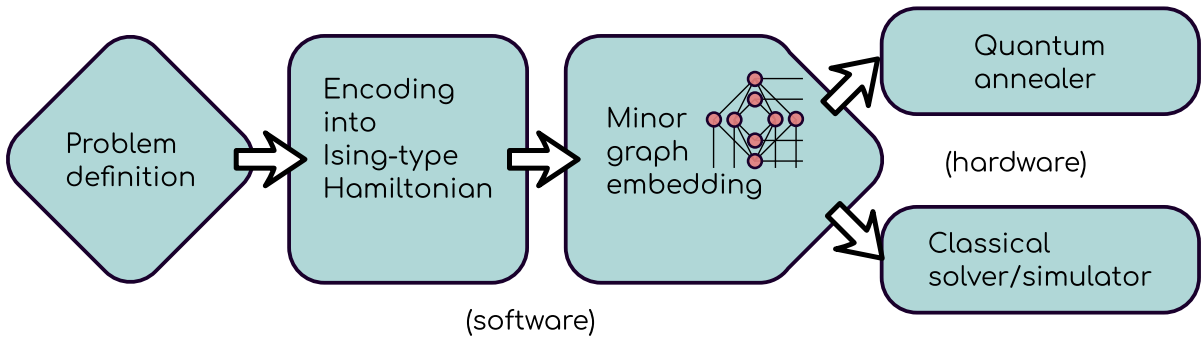
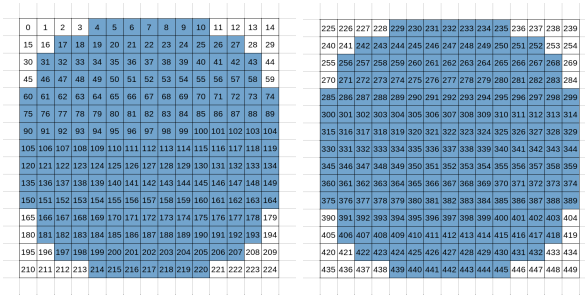


Fig. 7.2 The operation stack of a quantum annealer (based on [64]).

A *PWR* reload core with three different fuel enrichments is used as an example problem. The following assumptions have been made to simplify the problem:

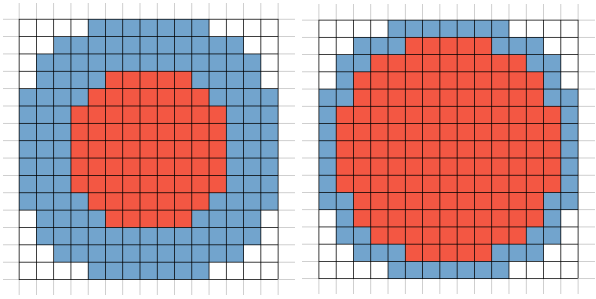
1. Burnable poisons (BPs) are neglected.
2. Control rods and instrumentation are considered to be withdrawn.
3. Only the start-up core made up of fresh fuel is considered.
4. Scenarios with two fuel-type and three fuel-type reload cores will be considered.

Based on the software stack in Figure 7.2, the problem of fuel loading must be encoded into a suitable Ising model. First, a two fuel-type reload core is considered. Each assembly in the core is assigned a logical qubit as shown in Figure 7.3a. The qubits are named numerically according to the image for regularity in the code, but only qubits that represent an actual assembly (blue in the figure) are assigned. This means that only 193 logical cubits are required to describe the system. Second, a three fuel-type reload core is considered: two qubits are assigned for each assembly, as shown in Figure 7.3b, which enables the assignment of three meaningful states in an Ising-type model.



(a) Two fuel-type core (b) Three fuel-type core

Fig. 7.3 Core representation in qubits



(a) (b)

Fig. 7.4 Transverse fields, h

In order to encode the kind of knowledge-based rules used by Galperin [71], the connectivity of the qubits must be set up and interactions are defined by setting values of the transverse field, h, and the exchange constant, $\mathscr{J}$, for every connected i and j. Galperin's example heuristic rules are used as the basis for the rule set used in this study. They are as follows:

1. Fresh fuel assemblies should not be loaded into peripheral core positions.
2. Twice-burnt fuel assemblies should not be loaded into innermost core positions.
3. Two fresh fuel assemblies should not be loaded into adjacent positions unless one of them is adjacent to a peripheral one.
4. Two twice-burnt assemblies with high accumulated burnup values should not be loaded into adjacent positions.

In order to translate this rule set to the core design tasks in this chapter, the assumption is made that the cycle number (burnup state) of the fuel is equated to enrichment. Thus, the highest enrichment fuel is equated to fresh fuel in a batch system, and the lowest enrichment fuel is equated to twice-burnt fuel. Rule 3 is then applied as is, while Rules 1 and 2 are inverted compared to Galperin's rules [71], due to a desire to be compliant with the reload core design used in the BEAVRS benchmark [86].

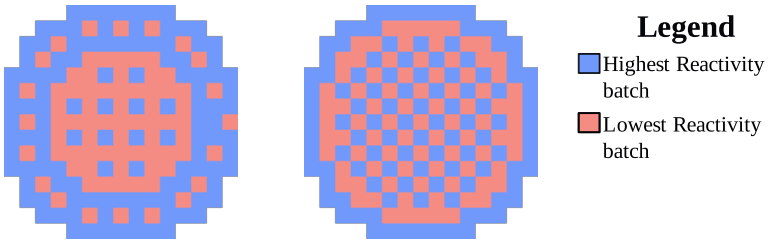
In the Ising model, the transverse field, h, is a blocking force that overrides the connectivity of qubits. In order to equate it to the heuristic rules, it is defined using one of the patterns shown in Figure 7.4a and Figure 7.4b. This allows qubits to have a particular affinity for one type of spin based on position in a particular region (defined radially here). This is used to enforce Rule 1. The exchange constant, $\mathscr{J}$, is then designed to ensure that the system obeys Rule 3. This is achieved by creating a connection between each qubit and adjacent qubits. The extension to 3 batches also connects qubits that are adjacent to the edges of the regions defined in the transverse field.

In the next step, the connections defined in the logical qubits of the Ising model description of the problem are translated to the actual architecture of a target *QA*. The Ising description can connect any number of qubits together in any arrangement, and actual *QAs* must emulate this functionality. The solution of this process is called the *minor embedding*. A common arrangement of connections for physical qubits is called a *chimera graph* [239, p. 148]. The box labelled 'Minor graph embedding' in Figure 7.2 shows a single group of eight qubits from a chimera graph: the groups are locally interconnected and each qubit is connected to two adjacent groups. In order to emulate a logical qubit's connections, multiple actual qubits are strongly coupled together until the correct connectivity is achieved.

7.2.3 Results

Patterns from Simulated Quantum Annealers

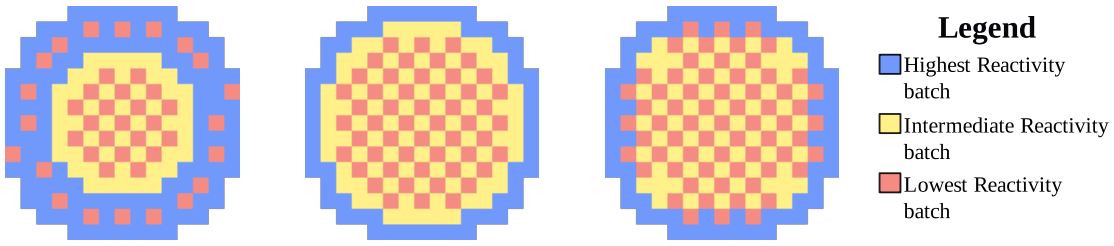
The results in this section were created using SQA [42] on conventional hardware. Nevertheless, the algorithm used attempts to model the quantum annealing process accurately and the software used is easily converted for use on an existing *QA*. The first result is that it is possible to transcribe a set of rules for *PWR LP* optimisation into a real-world quantum computer. The two fuel-type problem has a minor embedding that fits within a 16×16 chimera graph, as this is the largest quantum computer at the time of writing. The three fuel-type problem has a minor embedding that fits onto a 25×25 chimera graph, which is expected to be the largest implementation at the time of submission [2].



(a) LP 1 (b) LP 2

Fig. 7.5 *LP* designs using SQA for a two fuel-type PWR.

Two fuel-type *LPs* optimised using an SQA are shown in Figure 7.5. The Figure 7.5a *LP* was found when there is a bias for a large radial region with adjacent assemblies experiencing an inverting force. The radial region is smaller for the Figure 7.5b *LP*, and an even split of the fuel types is achieved. *LPs* found in the three fuel-type case are shown in Figure 7.6. Rules 1 and 2 create the outer 'blanket' region, while Rule 3 generates the 'checker-boarding' effect. A start-up core *LP* from the BEAVRS benchmark [86] is shown in Figure 7.6c, for comparison; in this, the highest w/o U235 correlates well with fresh fuel, the next w/o U235 with the once-burnt batch and the lowest w/o U235 with the twice-burnt batch. Note that the BEAVRS start-up core does not exactly conform to Galperin's rules.



(a) LP 1 (b) LP 2 (c) BEAVRS initial LP

Fig. 7.6 *LP* designs using SQA for a three fuel-type PWR compared and c) the initial core *LP* in the BEAVRS benchmark [86].

Since the problems posed in this chapter are all symmetrical, the global optimum must also be symmetrical. A simple metric for the effectiveness of the algorithm is needed to establish proximity to the global optimum. The metric used was to calculate the degree of quadrant difference, by comparing each quadrant rotated appropriately as follows:

$$\mathrm{R} = \frac{\Sigma_{N=2}^{4} \Sigma_{x=0}^{w/2} \Sigma_{y=0}^{h/2} Q_{x,y}^{1} - Q_{x,y}^{N}}{w \times h} \tag{7.5}$$

where: R is a measurement of quadrant rotational differences; Q^{1-4} are the rotated quadrants of the output 'core' *LP*; x is the horizontal position coordinate; y is the vertical position coordinate; w is the model width; and h is the model height.

This calculation produces a value in the range 0–1, with 0.03125 per non-similar output for 15×15 arrays such as here. Although this does not effectively rank the solutions and has other limitations, it gives an easy-to-calculate measurement of proximity to the global optimum of the heuristic model, based on the assumption that all solutions are within a few pixels of the global optimum and that optimum has order 4 rotational symmetry. This can be checked by visual inspection of the results (examples shown in Figures 7.5 and 7.6). Table 7.1 shows the results of this calculation for 30 runs for each of the configurations. Readers should note that this code is only running on a single core and has not been optimised (see p. 201 for a description of the laptop hardware used here).

Table 7.1 SQA results (mean of 30 runs), values to 4 s.f.

Cases	**Batches**	**Quadrant Differences** (R)		**Run Time (s)**	
		mean	σ	**mean**	σ
Figure 7.5a	2	0.1875	0.08860	547.1	9.510
Figure 7.5b	2	0.3521	0.07603	553.7	22.35
Figure 7.6a	3	0.09861	0.03414	787.2	20.23
Figure 7.6b	3	0.0	0.0	853.36	65.138

Validating Results in a Nodal Code Simulation

Although visual inspection of the results from the SQA process shows that the designs have a regularity that would be expected of good results, the system is only optimizing heuristic models developed according to a set of rules. To check whether these models have optimal solutions with favourable characteristics in a simulated *PWR*, they were analysed using a neutronics simulation. *LPs* from Figures 7.5 and 7.6 were selected and simulated using a PANTHER/WIMS model of the BEAVRS benchmark [86, 82]. In order to evaluate the *LPs*, octant core symmetry is applied, which is considered acceptable in this case since the solutions exhibit a high degree symmetry and it is common in fuel loading pattern analysis (e.g. [175, 71, 139, 60] and [75, p. 599]).

Some indicative statistics for a number of simulated cores are shown in Table 7.2. The cores are labelled by the Figures 7.5–7.6 for two and three fuel-types, with Figure 7.6c representing a simplified version of the BEAVRS initial *LP* and the core labelled "2.6 w/o U235" being a uniformly loaded core with an intermediate enrichment. Although the latter is not a performant *LP* in terms of EOC boron or *PPF*, it is included for comparison with the optimised *LPs*.

The WIMS/PANTHER BEAVRS model used in this study has shown good agreement with the benchmark data [82]. Since the heuristic surrogate model used here does not attempt to consider BPs, a simplified version of the initial *LP* was generated (Figure 7.6c) for comparison. Although the PANTHER results for Figure 7.6c show significant divergence from actual BEAVRS results, the differences are most pronounced at beginning of cycle (BOC). This implies that these discrepancies are due to the removal of BPs, which act to reduce the initial reactivity and the PPF.

The results for three fuel-type *LPs* are promising. The SQA solution for three fuel-types and a smaller radial region (Figure 7.6b) has improved PPF values compared to the Figure 7.6c design, albeit at a shorter cycle duration, as indicated by the end of cycle (EOC) boron concentration. While the two fuel-type *LPs* are less optimal than the three fuel-type ones, they are superior to the single fuel-type core (2.6 w/o U235) in all aspects except final PPF.

Table 7.2 Results for *LPs* burned up to 13593 MWd/tonne in PANTHER

Core	**BOC PPF**	**EOC PPF**	**BOC boron (ppm)**	**EOC boron (ppm)**
Figure 7.5a	1.854	1.299	1816	100
Figure 7.5b	2.849	1.247	1682	132
Figure 7.6a	1.760	1.127	1345	−41
Figure 7.6b	1.866	1.188	1518	71
BEAVRS - model (Figure 7.6c)	1.462	1.173	961	−32
BEAVRS - actual	1.438	–	975	0
2.6 w/o U235	2.158	1.163	1888	213

7.3 Extension: LPs from the DW2000Q Quantum Annealer

In light of these promising results, minor embeddings, such as those shown in Figure 7.7, were submitted to a D-Wave *QA*, the DW2000Q_5 machine, through the web interface [2]. Access time to these machines is limited, so care is taken when selecting the embedding. Although minor embeddings were attempted for the three fuel-type case, no solutions were found that embed the problem of 449 logical qubits within the 2038 real qubits, so experiments concentrated on two fuel-type cases, which only require 224.

Initial results from the default values of the *QA* yielded highly degraded results. So a study was carried out, varying ‘chain strength’, ‘anneal time’, and ‘number of reads’. However, results were inconclusive. Eventually, this was found to be due to the algorithm used to recreate the minor embedding for each test. The embedding used is optimised by a separate iterative

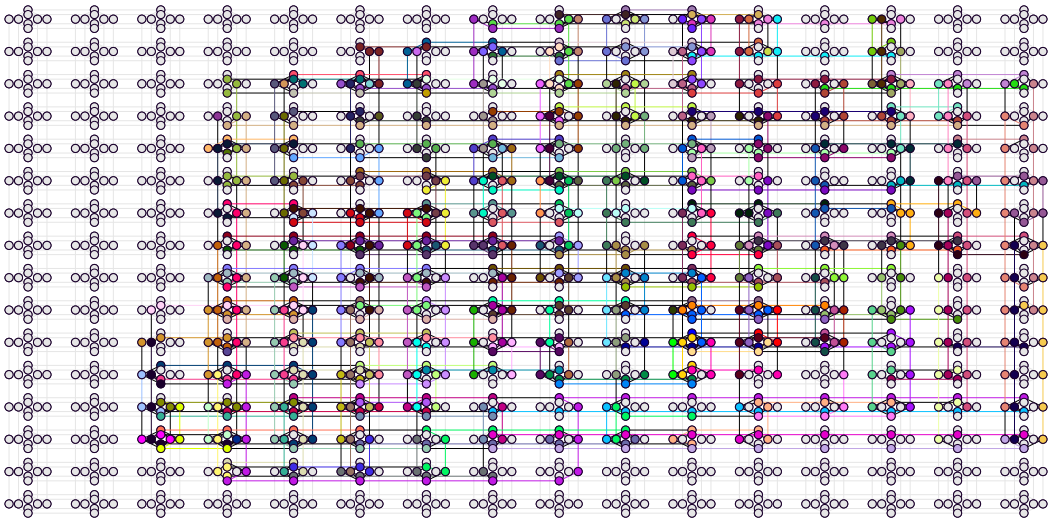


Fig. 7.7 An example of a minor embedding for the algorithm for two fuel-types in an idealised 'chimera graph'

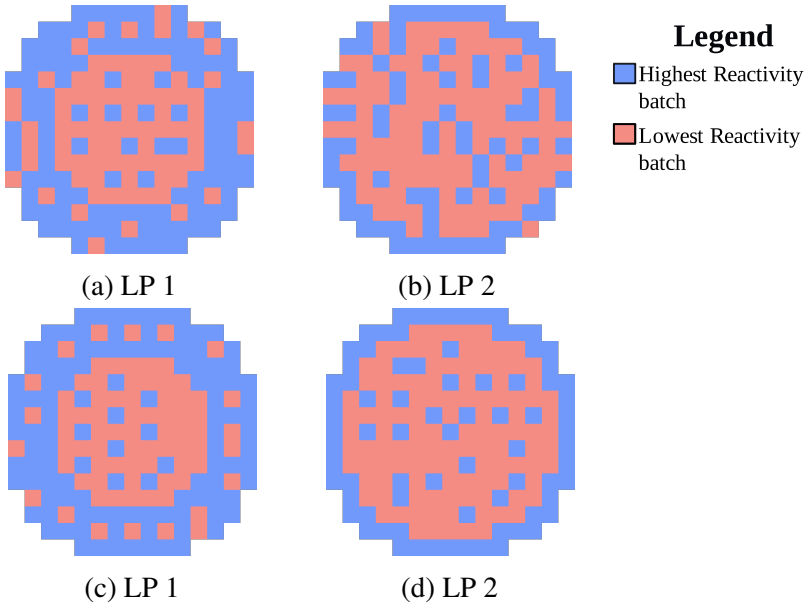
optimiser, and the resulting variation has a larger effect on the output than the other variables investigated.

The algorithm is pushing the boundaries of present-day DWave *QAs*. This means that the 'minor embedding' forms very long chains in order to simulate 'virtual qubits'. These chains can become uncoupled during the simulation, so the algorithm needs to be carefully tuned. In the end, over 2500 runs were made and the minor embedding with the shortest maximum chain length was recorded. This yielded a minor embedding with a maximum chain length of 8.

Examples of the results achieved in August 2019 with the DWave DW2000Q_5 *QA* are shown in Figures 7.8a and 7.8b. Visual inspection shows that the results are nowhere near as good as the SQA results. Although the time taken to find these results is significantly faster (Table 7.3) than for the SQA process (Table 7.1), no increase in speed justifies incorrect results However the technology is not yet mature and improvements are ongoing. Figure 7.8a and 7.8b were generated in August 2019, when this study was carried out. Upon review of the the code for release, a possible improvement in accuracy. This can be seen in the same results generated in July 2020 on the DW2000Q_6, as shown in Figures 7.8c and 7.8d and estimated quantitatively in R in Table 7.3.

Table 7.3 Results on the DW2000Q_5 in August 2019 and DW2000_6 in July 2020 *QAs* (mean of 5 runs), values to 4 s.f.

Outer Region	Machine	Batches	Quadrant Differences (R) mean	σ	Run Time (s) mean	σ
Figure 7.4a	DW2000Q_5	2	0.5792	0.05135	0.065 51	0.0000
	DW2000Q_6	2	0.5292	0.1302	0.027 16	0.000015287
Figure 7.4b	DW2000Q_5	2	0.5063	0.08979	0.065 52	0.0000
	DW2000Q_6	2	0.4479	0.04480	0.027 15	0.00001654



(a) LP 1 (b) LP 2

(c) LP 1 (d) LP 2

Fig. 7.8 Example *LPs* generated using QAs; a) and b) were generated in August 2019 on a DW2000Q_5 *QA*. c) and d) were generated in July 2020 on a DW2000Q_6 *QA*.

7.4 Discussion

This chapter presents promising results of an initial study of the suitability of present-day and near-present-day quantum computers for optimisation of fuel arrangements in nuclear power plants. The exciting conclusion is that, using this technique, a shortcut around the combinatorial explosion might be found.

The two fuel-type implementation gives results that can be seen by inspection to adhere to the simplified rules extracted from Galperin [71]. The three fuel-type solutions shown in Figure 7.6 compare well with the arrangement of different fuels in a start-up core from the BEAVRS benchmark (Figure 7.6c). Some differences can be noted as coming from Rule 3 used to guide the optimisation. Note that the annealer is not using symmetry to generate these *LPs*;

symmetry arises from the minimum energy solution to the surrogate model. The generated *LPs* were then simulated using a nodal reactor physics code to estimate their actual performance. The results show that the *LPs* found for three fuel-types are promising when compared with the start-up core designed for a commercial reactor.

Experiments on the actual D-Wave quantum computer have enabled insights into the process of generating a minor embedding for a real-world problem, and a study of the effects of the parameters involved shows that the quality of the minor embedding is the most important variable for getting good results in this case. Example results on the DW2000-Q, Figure 7.8, show that the results from the D-Wave machine are not very reliable. This is thought to be due to the large chain length of the embeddings that break with a probability proportional to the length. SQA does not suffer from chain breakages since there is no need to embed the system in real hardware.

The timing results in Tables 7.1 and 7.3 show that quantum annealing is more than 8350 times faster than SQA. Much of this speed-up is due to the architecture of the machine being specifically designed for optimisation of Ising-type problems (in the same way as FPGA designs of an algorithm can achieve results in a single clock cycle whereas standard processor architectures performing the same algorithm sequentially may take thousands of clock cycles[237, p. 3]). From of this study, it is apparent that the SQA consistently delivers better results than a *QA* at this time. One possibility that has been suggested is that SQA can outperform 'classical' simulated annealing for problems with many local optima due to the simulation of quantum tunnelling effects [45]. If true, SQA may be a good fit for developing *LPs*, since *LPs* are generated on a scale of months to years, so a slow but accurate algorithm is acceptable.

This study is by no means a complete investigation of fuel loading optimisation with quantum annealing techniques. Future work should look to encode physical aspects of the neutron transport into the problem and to generate more realistic scenarios. There is potential to express a fission matrix in the Ising model form. Thus *QA* shows significant future promise for solving complex nuclear engineering problems. While this initial study gives promising conceptual results, limitations are acknowledged. These include: the list of assumptions on p. 128; the simplicity of the heuristics–though knowledge-based, they are much simpler than the actual interactions of components in a nuclear reactor; and the lack of consideration of specific histories (as is normal for a multi-batch core). Although the encoding does not enforce that equal numbers of assemblies of each fuel-type (batch) are used, careful design of the $\mathscr{J}$ and h fields can ensure that the minimum energy Hamiltonian is not biased towards a particular fuel category. The *QAs* used in this chapter are a specialised hardware architecture that is specifically designed for optimisation of Ising-type problems. They are very fast because they

use a physical system to optimise, and they have also benefited from a great deal of investment. It follows that they operate much quicker than conventional computer hardware simulating the same physical system. However, this does not demonstrate a 'quantum speed-up', and there is an ongoing discussion as to whether there is actually a 'quantum' advantage to quantum annealing in the current implementation [197, 35]. At this time the consensus appears to be that there is some advantage [101], but it has been noted that the quantum entanglement used in the D-wave machines used in Section 7.3 is of a form that is easily simulated on conventional computers [161]. An interesting foreseeable outcome may be that although quantum annealing can be shown to be superior to simulated annealing, it does not benefit from 'quantum speed-up'. This would mean that SQA offers the same advantages as the existing generation of *QAs* [45], albeit at an architectural and time overhead due to the design of current conventional computers. Furthermore, some questions are still unanswered within the quantum computing community about quantum annealing in the current implementations, due to the finite temperatures at which real systems necessarily operate [8].

Despite the questions surrounding present day implementations, it is known that the technology is rapidly developing [182], and insights may be gained through an understanding of this new technology. For example, it has been shown that the size of the difference Δ in Eq. (7.4) of the Ising model scales linearly with the number of variables [161]. This fact might enable an understanding of the search space; if it were possible to accurately define a nuclear optimisation in Ising terms, then it would be possible to infer the size of minimum Δ for that problem.

This chapter has shown experimental evidence of the utility of alternative simulation approaches, to encourage the utilisation of the technology of quantum computers within the nuclear industry. The implementations of the algorithm in this paper apply to real and simulated and adiabatic quantum computers. Due to the concerns raised above, the data generated by the DW2000 should be considered tentative.

7.5 Conclusions

This chapter has described a method of encoding relatively simple rule sets in order to optimise a full-core *LP* at the assembly level using a simulated quantum annealer. Although there is a great deal of work still to be done to create a useful tool for *PWR* fuel management, the basic building blocks of the technique have been presented and the results show promise. Further work should look at ways to transfer real physics connectivities into the Ising simulation, as well as ways to transfer burnup histories and gradients into the fuel. Quantum computer qubit density is currently increasing at a rate comparable to Moore's Law [182], which means that it

is expected that it will be possible to achieve significantly more advanced calculations in the near future.

The technology of adiabatic quantum computing and quantum annealing have been introduced, with a start-up core loading pattern problem. A mapping of heuristic rules into an Ising model is created, and the steps to generate the actual embodied solution are carried out: the model is converted from the Ising model to a minor embedding and solved using SQA and on an existing *QA*. This represents a novel approach to *LP* generation.[2]

[2] Code used in this section is available for audit, reproducibility and derived works. A copy can be obtained from the repository under the permissive MIT license:

https://bitbucket.org/ajw287/chapter7-quantum-annealing.git

Chapter 8

Discussion and Conclusions

8.1 Insights into Surrogate Model Development

This thesis describes the development of a software framework for evaluation of *SMO* strategies, as proposed in Section 1.3. The framework has been carefully designed to maximise the likelihood of success of the surrogate models and then used to support development and systematic testing of three different surrogate models. The capacity of the framework to evaluate fuel management scenarios has been demonstrated by implementing a series of experiments for a number of different fuel *LP* scenarios and surrogate models.

The concerns of the thesis remain grounded in the real engineering problem of fuel loading optimisation in *PWRs*. However, due to the highly commercially sensitive nature of fuel loading pattern optimisation, it has been necessary to develop the method of *SMO* on hypothetical models. While it is important to acknowledge the limitations of the simplified models used in this thesis, they have proved a fertile experimental apparatus for the development of methodological experiments for loading pattern optimisation. For optimisation of computationally expensive objective functions, this thesis identifies the following situations where surrogate model optimisation should be considered:

- When the optimisation will be carried out routinely, since the investment of resource in creating the model is justified compared to the saving over many optimisations (see Figure 5.21). This is true for PWR fuel management.
- When the objectives are unclear, a model such as a *CNN*, may be used to generate basic outputs that allow investigation of a number of objectives (see pp. 84–93). Due to the different business groups involved in PWR fuel management (see Figure 1.1), this may apply.

- When the objectives are subject to stochastic noise, a surrogate model can be generated that is deterministic (see Section 6.7). This is true for simulations using the Monte Carlo method, a popular approach for neutron transport analysis.
- When there is a hardware advantage to optimise a surrogate problem. Either in terms of increased parallelism (see Section 5.8) or in terms of specialised hardware (see Section 7.3). This has been shown to be possible for *LPs* in Chapter 7.

These criteria can also be seen in a wide variety of other engineering problems.

Three forms of surrogate model have been presented: deep learning models in Chapter 5, a statistical 'black box' model based on fission matrices in Chapter 6, and the solution of a heuristic model is carried out by converting the problem to a form that works on *QAs* in Chapter 7.

In each of these exploratory studies, a model of the nuclear reactor is generated that does not solve the neutron transport equation explicitly. In the deep learning and fission matrix models, a function approximation is generated from samples made on a neutron transport simulation. In the heuristic approach, rules are generated by expert observers based on their knowledge of results from simulated and real reactors and an understanding of the underlying physics.

8.1.1 Deep Learning

The investigation of two different established topologies of deep learning models in Chapter 5 demonstrates the flexibility of the surrogate model framework. Deep *MLP* and *CNN* types of network were applied to solving simple nuclear fuel *LP* problems in what is among the first applications of deep learning to the field of nuclear engineering. The *CNN* appears to slightly outperform the *MLP*; this is probably due to the *CNN*'s specialism for spatially structured data, which two dimensional *LPs* are.

It has been demonstrated that the training set can be generated for a relatively low computational cost and with the highest level of parallelism. The use of Sobol sequences as the basis for the training set is demonstrated to outperform a uniform random training set. This can be notionally explained by considering the effects of the *central limit theorem* (demonstrated in Appendix B.1). The *central limit theorem* biases multiple random inputs towards the mean. In the case of the nine dimensional (or higher) input space used in Chapter 5, this becomes a significant effect. Thus, a surrogate model trained on a random training set is biased towards predicting results with a mid-range mean and these are also the initial conditions of the optimisation. The entire *NDF* is unlikely to be found at the initial conditions of the optimisation, so as soon as the population of solutions iterates, it is likely to move away from the specialisation of this surrogate model. In contrast, the Sobol sequence is roughly evenly spaced across the

input space. In keeping with the results in Section 5.3, Sobol trained *deep networks* are less performant at predicting values on a random test set, but they appear to be more robust for use as a surrogate model in *SMO*.

The novel application of *CNNs* to predict core parameters is a significant development. It has been shown in this work that they can be used to generate results for small cores with quadrant rotational symmetry. However, it is important to recognise that these limitations are only due to the implementation. This thesis demonstrates the value of a technique which could efficiently generate predictions for whole-core optimisation, on a pin-by-pin basis. Established techniques used with *CNNs*, such as transfer learning [250], allow training data from this thesis to be used directly as the basis for initial training of larger problems such as whole-core analysis.

Two out of four experiments in the initial study achieve weakly dominating solutions in the *SMO NDFs*, while the other two experiments generate results that are closer to the *NDF* than the initial random population in NSGA2. This points to a conclusion that deep learning surrogate models can be efficiently used to augment the solution of *DSO*. The primary approach investigated in this thesis is by running the *SMO* in parallel to a *DSO* – to search for alternative non-dominated solutions. Alternatively, the results of the *SMO* could potentially be fed into the initial population of the *DSO* in a 'hybrid' optimisation – to try to reduce the number of generations required using the direct simulation.

8.1.2 Fission Matrix

The investigation of the fission matrix further demonstrates the flexibility of the software architecture. Well-defined software modules have allowed different surrogate models to be easily changed without significant code rework. The results found in the experiments showed that the fission map generated by combining fission matrices appears to correlate statistically with test patterns. It is then successfully able to generate a number of loading patterns that are relatively close to the results generated by the *DSO* in terms of performance. The fission matrix surrogate model represents a novel use of the fission matrix and displayed performance that was competitive with deep learning methods in terms of computational resource. Unlike most mathematical modelling, models for *SMO* do not need to make extremely accurate predictions, but the correlation of the surrogate models and the objective functions must be good in order for the optimisation algorithm to find *NDF* solutions that correlate closely with solutions that are non-dominated in the objective space.

As mentioned in Section 6.8, a major potential advantage of the fission matrix form was to express a surrogate model in a form that is compliant with the Hamiltonian of a polariton condensate, a form of quantum computer proposed by Berloff et al. [21] (cf. Equation 6.1 and

Equation 7.3). Development of the fission matrix model is ongoing; however, at the time of writing, the model in Chapter 6 was not mature enough to be applied successfully in a polariton simulator. Consequently, heuristic rules were used to create the Ising model that was used as the surrogate model in Chapter 7.

8.1.3 Quantum Annealing

A simple optimisation scheme has been demonstrated for a full 193 assembly *PWR* start-up core, without reliance on symmetry. This demonstrates a major advantage of quantum annealing – that it potentially avoids the *curse of dimensionality*. The *curse of dimensionality* limits the number of degrees of freedom a problem can have, if it is to be solved on classical computers. This is because the number of permutations of inputs grows exponentially with the number of variables. The implication of this strategy is that it could be possible to generate solutions to real running reactors, where a variety of factors, such as limited stock, 'early retirement' of a fuel assembly or operational sensor observations, lead to considerations that make the assumption of eighth core symmetry sometimes impractical.

A simple set of heuristic rules were encoded into a form that is compatible with an Ising model. This enables the optimisation of the entire core of the PWR without reliance on symmetry. It has been shown that the methodology for generating the encoding creates solutions in *simulated quantum annealing* that perform comparably to a simplified version of a commercial reactor start-up loading pattern from the BEAVRS benchmark. This is a promising result for two main reasons. Firstly, it shows that the approach is well suited for the very large number of degrees of freedom that realistic problems must consider. Secondly, it shows for the first time that SQA can be used to generate loading patterns, and demonstrates (with limited performance) the same algorithm running on a DWave DW2000 hardware *QA*.

8.1.4 Trends in Computational Optimisation

Advances in computational optimisation of nuclear fuel *LPs* have coincided with the revolution in digital computers. As a result, the techniques developed in nuclear fuel optimisation utilise and are synergistic with state-of-the-art in computing at the time.

Early attempts began by creating simple rule-based strategies [81]. These solutions are guided by the human expert and create reasonable solutions but rely on the knowledge of the human designer. As digital computers with suitable power started to become available to researchers, approaches such as *SA* and evolutionary algorithms were popularised by researchers such as Parks and Ahn [173, 7]. These techniques are still computationally expensive today and represent the most advanced methods of searching nuclear fuel loading pattern optimisation

landscapes. At the same time, the alternative method of optimisation with heuristic rules has been developed by Galperin and Nissan [72]. This has the advantage of explicitly embodying the physics of the problem into the heuristic rules, but the disadvantage that the step between the physics and the heuristic rules is in the mind of the designer.

This thesis has been developed as fundamental changes in the advancement of computers occur (primarily the end of Moore's law [226]) and are again causing fundamental changes in the way that researchers and industry will solve problems using computers. A significant opportunity now exists for the nuclear research community, which led the adoption and development of numerical and simulation techniques [170, p. 2], to be at the forefront of these developments.

At the start of this work, deep learning was positioned to have a great impact on nuclear academic research and industrial development. At that time, breakthroughs in deep learning technology opened the field to new applications. Simultaneously, open APIs were breaking down the barriers to application of these technologies and significant investment in deep learning hardware was being made by the world's largest companies. Three years later there is now significant interest in applying this technology in the field of nuclear engineering. The experiments in Chapter 5 demonstrate some of the significant advantages. Today there is a great deal of hype around the developments of quantum computing. Table 8.1 references and compares the state of deep learning in 2017 and quantum computing technologies today. These technologies are fundamentally different, so it is is wrong to assume that the research trajectories will be parallel. However, it would also be a mistake to ignore these significant indicators that quantum computing is going to progress significantly, since this technology may provide opportunities to the nuclear industry.

Table 8.1 Enabling factors for the technologies of deep learning in 2017 and quantum computing in 2020.

	Deep learning in 2017	**Quantum Computing in 2020**
Fundamental research breakthroughs	*CNNs* [117], deep learning [123]	Quantum Supremacy [15], Adiabatic Quantum Annealing [100]
Free, accessible APIs	Tensorflow [6], Theano [20], Keras [37]	Quiskit, Forest, Dimod [42]
Investment from large corporations	Google, Facebook	IBM, AT&T, Alphabet
Novel hardware developments	Tensor Processors [98] and others [48]	Bristlecone [105], DWave [2]

8.2 Recommendations for Future Research

The experiments presented in this thesis are fundamental research experiments and each of them poses more questions than are answered. However, from this thesis, a scaffolding for further research can be envisaged. In keeping with the growing trend for open data in science and best practices in the field of nuclear engineering, the author makes the code and data available under the permissive MIT and Berkeley licenses. This means that other researchers can directly pick up the code of the framework or use the training datasets as the basis for further research and commercial work without financial obligations.

The deep learning models using *CNNs* are novel experiments and represent fundamental research. The experiments in this thesis have demonstrated a proof-of-concept of the application of these technologies to the generation of loading patterns. However, many opportunities exist to extend this work. Contrasting the results of this thesis made using Serpent with solutions using deterministic methods would enable a consideration of the benefit of *SMO* when working on a stochastic objective function.

The research in this thesis also enables further fundamental research avenues. For example, transfer learning [214], which is a common technique used with *CNNs*, can augment a novel training set with results from datasets in this thesis. The resulting *CNN* can outperform a model generated with only the novel data. There are a number of applied research projects which could extend the work carried out in order to prepare this technique for commercial readiness. These include the consideration of more complex scenarios, particularly multi-batch burnup operation, burnable poisons, and other fuel types which may be required. Using techniques from transfer learning, it should be possible to directly adapt models from this thesis and retrain them reusing the code to apply to more complex scenarios. The concept of extending surrogate models based on *CNNs* to full core analysis is a very exciting prospect. The prospect of a pin-for-pin optimisation of a PWR could potentially inform the out-of-core purchase of fuel for the predicted core loading pattern. This would be a solution to Turinsky and Parks' 'grand challenge' of nuclear fuel management [220]. Applying further constraints (fuel inventory, early retirement of single assemblies, fuel costing) to the code framework would be advantageous to make this a commercially ready tool for engineers operating in the field.

The fission matrix surrogate model in Chapter 6 demonstrates interesting results in terms of the simple burnup model used. However, in the experiments in this thesis, inaccuracies are seen in the estimate of *PPF*. It may be of interest to reconsider the fission matrix model with a view to improving the *PPF* prediction. Extending these experiments to contrast the *DSO* in Serpent with results from a deterministic neutron transport solver would help to ensure that the *NDFs* found by optimisation on Serpent was not unfairly advantaged by stochastic noise.

Significant further work that builds on the fission matrix model in this thesis would be to establish an improved version that could be applied in larger and multi-batch loading pattern problems. A major advantage of the fission matrix approach is that it can be expressed in a form that permits its interpretation as an Ising model (see Section 7.2.1). This would mean that further research using this model could potentially be converted to a form that can be executed on quantum computing hardware [98].

Chapter 7 considers the use of *QA* applied to established heuristic rules for nuclear loading pattern generation in the context of the changing landscape of computer hardware which has coincided with this work. Extending this work directly by expressing more complex representations of nuclear systems into forms that are amenable to alternative computational co-processors will be of increasing importance as the rate of development of CPU technology slows. Incidentally, considering the implications of quantum accelerators or deep learning hardware technology such as TPUs [98] may yield significant advantages for neutronics and *CFD* simulation methods.

This thesis has focused on the computational advantage of *SMO*. Although deep learning surrogate models cannot replace *DSO*, they can significantly augment it – by running in parallel. Further research should extend the studies in this thesis, to confirm whether *SMO* reliably delivers solutions for a wider variety of fuel management scenarios, to reinforce the advantages of running *SMO* in parallel. However, it is possible that *SMO* could bring other advantages as part of a 'hybrid' optimisation (see Section 8.1.1). It is likely that the optimisation algorithm responds differently to the surrogate model and qualitative advantages might be accomplished. For example, neural network models are usually generated from continuous activation functions when the direct simulation may have discontinuities. It would be of great interest to modify the framework to initialise the *DSO* with the results of many runs of *SMO*. Significant computational cost could be saved by starting the *DSO* closer to the *NDF*, which might be achieved by using the outputs of *SMO* runs. The danger of this approach is that, if the *SMO* somehow biases the optimisation, then significant results might be missed. Hence, interesting fundamental research might be undertaken investigating the implications of this kind of 'hybrid' approach.

A fascinating but highly complex frontier of fundamental research in optimisation is exposed by the *NFL* theorem from Wolpert and MacReady (see Section 1.2.2), which shows us that, within the set of all problems, no strategy outperforms any other. This troubling theorem appears to leave us with no handle on the efficient search of complex spaces, yet real-world problems are routinely solved efficiently. In this thesis, an explicit method for addressing the problems of the *NFL* theorem is developed. By embedding an approximation to the objective function into the optimisation algorithm, the resulting optimisation will be

biased towards the objective function. This principle has been implemented in a number of different experiments. However, the fact that iterative optimisation algorithms reliably work, hints towards some level of universal structure in real-world problems. Universality is currently an active area of mathematical endeavours [52] and can be observed in the well known *central limit theorem*, which demonstrates the universality of the Gaussian distribution. It would be a significant achievement to show that there is an optimal search approach for the subset of 'realistic problems', even if this is just the set of problems that adhere to some fairly basic universal principles. This subject is of interest to mathematicians, but it would inevitably be relevant to surrogate model implementations by giving insights on the most basic assumptions that can be applied across engineering problems.

8.3 Conclusions

The problem of in-core fuel management is known to be a complex and highly nonlinear problem [71]. However, investigating this engineering problem quickly reveals lines of inquiry that transcend the specific engineering problem. Developments in optimisation, such as the *NFL* theorem (Section 1.2.2), inspired the decision in this thesis to generate optimisation strategies that utilise aspects of the topology of the objective function and insights from theoretical physics to augment the optimisation algorithm. This thesis, therefore, has investigated the philosophical landscape of optimisation in general and *SMO* in particular, while remaining faithful to the practical implementation of *LP* optimisation schemes, albeit for a number of simplified start-up cores.

In Chapter 5, Experiment 1 (Section 5.4) demonstrates the advantage of using Sobol sequences for training deep learning models. This counter-intuitive result is interesting because it demonstrates the benefit of a global view of the objective for these example experiments and the emergence of form via the *central limit theorem* in random inputs. Experiment 2 (Section 5.5) shows that deep learning *SMOs* can generate solutions that dominate an equivalent *DSO*. Experiment 3 (Section 5.6) extends this investigation to a single batch burnup scenario and Experiment 4 applies a deep learning surrogate model to a small *SMR* core.

The fission matrix model is developed and applied in Chapter 6. Experiment 1 (Section 6.4) considers the fission matrix model for a number of loading patterns, concluding that it works most effectively for loading patterns that the resemble the flux distribution of the original 'training' loading patterns. Experiment 2 (Section 6.5) looks at the application of the developed model for optimising the enrichment of fuel and *PPF*. Experiment 3 (Section 5.6) attempts to generate a simple burnup scheme that is compatible with this model. The fission matrix model was developed with a view to implementation on a *QA*.

In Chapter 7, the conceptual advantage promised by *QAs* is introduced and the framework of code is adapted to optimise a simplified heuristic model of a 193 assembly *PWR* core. These novel technologies promise the possibility of finding global optima to some degree of certainty and are naturally adept at solving problems with a very high number of input variables. This is shown to be the case by optimising the full core without reliance on symmetry for both two-enrichment and three-enrichment *LP* scenarios. Firstly, optimal *LPs* for these heuristic rules are found using an SQA running on standard computational hardware. Then, as an extension, the code is executed on a DWave machine, which has been claimed to operate as a hardware *QA*.

SMO introduces an application for a method of optimisation where the original objective is replaced with an alternative model. This must be used to benefit the optimisation, whether it is to reduce the computational budget, counteract problems caused by stochastic noise or to allow operation on specific hardware. It has been demonstrated that significant computational benefit can be achieved. Furthermore, in some experiments, the resulting solutions of the *SMO* uncover aspects of the original objective that are not found by equivalent runs of the *DSO*, proving that the *SMO* can effectively augment results found by traditional iterative optimisation studies.

8.4 Significant Contributions to the Field

The author has worked on surrogate models of nuclear processes at a time when surrogate modelling has received a great deal of interest. This thesis has implemented and applied two techniques that have been created in other fields of research and applied them to nuclear engineering for the first time. These are deep learning *CNNs* and quantum annealing, which have been applied to fuel loading pattern design in simplified *PWRs* by the author with interesting and promising results. The fission matrix model, developed and applied in Chapter 6, demonstrates another novel contribution to the field, albeit with only limited success.

Three notable contributions to the field that were uncovered during this thesis are summarised here. Firstly, training sets based on Sobol sequences generate deep learning surrogate models that, on average, outperform random training sets for optimisation (despite relatively poor performance on random test sets). Secondly, *CNNs* can be created using relatively small training sets to optimise fuel *LPs*, generating pin-by-pin power maps that can successfully predict objective parameters during *SMO*. Thirdly, *QA* can be applied to a simplified heuristic surrogate model of a full reactor core, potentially generating globally optimal solutions to fuel loading problems without reliance on symmetry, and potentially overcoming the *curse of dimensionality*.

Appendix B includes a numerical study of the effects of NSGA2 hyperparameters on the optimisation of microcore initial conditions. This could be used to simplify the selection of parameters in other studies.

Deep *MLPs* have been applied to loading pattern optimisation in Chapter 5 and also FHR fuel design in Appendix C, demonstrating the versatility of the code developed and the computational advantages of *SMO*.

References

[1] (2018). *Proceedings of PHYSOR 2018*, Illinois. ANS.

[2] (2019). Press Release: D-Wave previews next-generation quantum computing platform. https://www.dwavesys.com/press-releases/d-wave-previews-next-generation-quantum-computing-platform. Accessed: 2020-07-30.

[3] (2020). The Berkeley Standard License. ftp://ftp.cs.berkeley.edu/pub/4bsd/README.Impt.License.Change. Accessed: 2020-07-30.

[4] (2020). NIST/SEMATECH e-handbook of statistical methods. https://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm. Accessed: 2020-07-30.

[5] (2020). *Proceedings of PHYSOR 2020*, Illinois. ANS. https://www.physor2020.com/proceedings, Accessed: 2020-10-22.

[6] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. http://tensorflow.org. Accessed: 2020-07-30.

[7] Ahn, D. and Levine, S. (1985). Automatic optimized reload and depletion method for a pressurised water reactor. *Nuclear Technology*, 71:535–547.

[8] Albash, T., Martin-Mayor, V., and Hen, I. (2017). Temperature scaling law for quantum annealing optimizers. *Physical review letters*, 119(11):110502.

[9] Aleksander, I. (1989). *Neural Computing Architectures: The Design of Brain-like Machines*. North Oxford Academic Publishing Co Ltd.

[10] Anderson, H., Davidon, W., Glicksman, M., and Kruse, U. (1955). Scattering of positive pions by hydrogen at 189 mev. *Physical Review*, 100(1):279.

[11] Antonov, I. A. and Saleev, V. (1979). An economic method of computing lp-τ-sequences. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 19(1):243–245.

[12] Areva (2013). Areva design control document rev. 5 - tier 2 chapter 04 - reactor. Technical report, United States Nuclear Regulatory Commission.

[13] Ariva and EDF (2011). UK EPR™ - Generic design assessment. Technical report, Ariva.

[14] Arnold, T. B. and Emerson, J. W. (2011). Nonparametric goodness-of-fit tests for discrete null distributions. *R Journal*, 3(2):34–39.

[15] Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., Biswas, R., Boixo, S., Brandao, F. G., Buell, D. A., et al. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510.

[16] Ayoub, A. F. A., McMahon, M. V., and Driscoll, M. J. (1996). Ultralong cycle length for pwrs: Physics and fuel aspects. *Transactions of the American Nuclear Society*, pages 347–348.

[17] Bae, I. H., Na, M. G., Lee, Y. J., and Park, G. C. (2008). Calculation of the power peaking factor in a nuclear reactor using support vector regression models. *Annals of Nuclear Energy*, 35(12):2200 – 2205.

[18] Bennet, D. J. and Thomson, J. R. (1989). *The Elements of Nuclear Power*. Longman.

[19] Berg, H. C. (1993). *Random Walks in Biology*. Princeton University Press.

[20] Bergstra, J., Bastien, F., Breuleux, O., Lamblin, P., Pascanu, R., Delalleau, O., Desjardins, G., Warde-Farley, D., Goodfellow, I., Bergeron, A., et al. (2011). Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*, volume 3, pages 1–48.

[21] Berloff, N. G., Silva, M., Kalinin, K., Askitopoulos, A., Töpfer, J. D., Cilibrizzi, P., Langbein, W., and Lagoudakis, P. G. (2017). Realizing the classical xy Hamiltonian in polariton simulators. *Nature materials*, 16(11):1120–1126.

[22] Berry, J. (2018). 2015 RECS survey data. Technical report, U.S. Energy Information Administration, 1000 Independence Ave., Washington, DC 20585.

[23] Bertels, K., DiCarlo, L., Geresdi, A., Scappucci, G., Taminiau, T., Vandersypen, L., Veldhorst, M., Wimmer, M., and Criger, B. (2019). Delftx: Qtm2x the building blocks of a quantum computer: Part 1.
https://courses.edx.org/courses/course-v1:DelftX+QTM2x+2T2018a/course/. Accessed: 2020-07-30.

[24] Bishop, C. M. (1996). *Neural Networks for Pattern Recognition*. Clarendon Press.

[25] Box, G. E. P. and Draper, N. R. (1987). *Empirical Model-building and Response Surfaces.* John Wiley & Sons, London.

[26] Box, G. E. P. and Wilson, K. B. (1951). On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(1):1–45.

[27] Bratley, P. and Fox, B. L. (1988). Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 14(1):88–100.

[28] Brown, F. B., Carney, S. E., Kiedrowski, B. C., and Martin, W. R. (2013). Fission matrix capability for MCNP, part I - Theory.

[29] Bui, L. T., Abbass, H. A., and Essam, D. (2005). Fitness inheritance for noisy evolutionary multi-objective optimization. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pages 779–785, New York. Association for Computing Machinery.

[30] Carney, S., Brown, F., Kiedrowski, B., and Martin, W. (2014). Theory and applications of the fission matrix method for continuous-energy Monte Carlo. *Annals of Nuclear Energy*, 73:423–431.

[31] Carney, S. E., Brown, F. B., Kiedrowski, B. C., and Martin, W. R. (2012). Fission matrix capability for MCNP Monte Carlo. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM, US.

[32] Chapot, J. L. C., Da Silva, F. C., and Schirru, R. (1999). A new approach to the use of genetic algorithms to solve the pressurized water reactor's fuel management optimization problem. *Annals of Nuclear Energy*, 26(7):641–655.

[33] Charles, A. J. and Parks, G. T. (2017). Mixed oxide LWR assembly design optimization using differential evolution algorithms. Number Paper 67936, New York, NY. ASME.

[34] Cheng, P., Pan, J.-S., Guoqing, L., Tang, Y., and Huang, C. (2010). A survey of performance assessment for multiobjective optimizers. In *Automation Science and Engineering (CASE)*, volume 0, pages 341–345, New York. IEEE.

[35] Cho, A. (2014). Quantum or not, controversial computer yields no speedup. *Science*, 344:1330–1331.

[36] Cho, N. Z. (2000). Benchmark problems in reactor and particle transport physics. Technical report, KAIST Department of Nuclear and Quantum Engineering Nuclear Reactor Analysis and Particle Transport Lab.

[37] Chollet, F. et al. (2015). Keras, a deep learning API. https://github.com/fchollet/keras. Accessed: 2020-07-30.

[38] Chorley, R. J. and Haggett, P. (1965). Trend-surface mapping in geographical research. *Transactions of the Institute of British Geographers*, pages 47–67.

[39] Chrisman, L. (2014). Latin hypercube vs. Monte Carlo sampling. www.lumina.com/blog/latin-hypercube-vs.-monte-carlo-sampling. Accessed: 2020-10-22.

[40] Christou, M. and Konstantinidou, M. (2012). Safety of offshore oil and gas operations: Lessons from past accident analysis. *Joint Research Centre, European Commission, Luxembourg*.

[41] Collins, K. B., Sankar, L. N., and Mavris, D. N. (2013). Application of low-and high-fidelity simulation tools to helicopter rotor blade optimization. *Journal of the American Helicopter Society*, 58(4):1–10.

[42] Condello, A. and contributors (2018). Dimod, a shared API for binary quadratic samplers. https://github.com/dwavesystems/dimod. Accessed: 2020-07-30.

[43] Corsini, A., Cervi, F., and Ronchetti, F. (2009). Weight of evidence and artificial neural networks for potential groundwater spring mapping: An application to the Mt. Modino area (Northern Apennines, Italy). *Geomorphology*, 111(1):79 – 87.

[44] Cressie, N. (1990). The origins of kriging. *Mathematical Geology*, 22(3):239–252.

[45] Crosson, E. and Harrow, A. W. (2016). Simulated quantum annealing can be exponentially faster than classical simulated annealing. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 714–723, New York. IEEE, IEEE.

[46] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

[47] Dario, I. (2018). Pygmo 2 documentation. https://esa.github.io/pagmo2/. Accessed: 2020-07-30.

[48] Davison, A. J. (2018). Futuremapping: The computational structure of spatial AI systems. *arXiv preprint arXiv:1803.11288*. Accessed: 2020-07-30.

[49] de Lima, A. M., Schirru, R., da Silva, F. C., and Medeiros, J. A. C. C. (2008). A nuclear reactor core fuel reload optimization using artificial ant colony connective networks. *Annals of Nuclear Energy*, 35(9):1606–1612.

[50] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[51] DeChaine, M. D. and Feltus, M. A. (1995). Nuclear fuel management optimization using genetic algorithms. *Nuclear Technology*, 111(1):109–114.

[52] Deift, P. (2006). Universality for mathematical and physical systems. *arXiv preprint arXiv:0603038v2*. Accessed: 2020-07-30.

[53] Delalleau, O. and Bengio, Y. (2011). Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems 24*, pages 666–674. Neural Information Processing Systems Foundation, Inc., San Diego.

[54] Dennett, D. (1997). *Kinds of minds : towards an understanding of consciousness*. Phoenix, London.

[55] Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43.

[56] Eberhart, R. C. and Dobbins, R. W. (1990). *Neural Network PC tools*. Academic Press Inc, Boston.

[57] England, T. R. and Rider, B. F. (1995). Evaluation and compilation of fission product yields 1993 (la-sub–94-170). Technical report, Los Alamos National Lab., US.

[58] Esquivel-Estrada, J., Ortiz-Servin, J. J., Castillo, J. A., and Perusquía, R. (2011). Azcaxalli: A system based on ant colony optimization algorithms, applied to fuel reloads design in a boiling water reactor. *Annals of Nuclear Energy*, 38(1):103–111.

[59] Fagerland, M. W. and Sandvik, L. (2009). The Wilcoxon–Mann–Whitney test under scrutiny. *Statistics in Medicine*, 28(10):1487–1497.

[60] Faria, E. F. and Pereira, C. (2003). Nuclear fuel loading pattern optimisation using a neural network. *Annals of Nuclear Energy*, 30(5):603–613.

[61] Fasano, G. and Franceschini, A. (1987). A multidimensional version of the Kolmogorov–Smirnov test. *Monthly Notices of the Royal Astronomical Society*, 225(1):155–170.

[62] Feynman, R. P. (1982). Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488.

[63] Fincham, J. and Friswell, M. (2015). Aerodynamic optimisation of a camber morphing aerofoil. *Aerospace Science and Technology*, 43:245–255.

[64] Fingerhuth, M., Babej, T., and Wittek, P. (2018). Open source software in quantum computing. *PLoS ONE*, 13(12)(e0208561).

[65] Fiorina, C., Scolaro, A., Siefman, D., Hursin, M., and Pautz, A. (2020). Artificial neural networks as surrogate models for UQ and data assimilation in 2-D/3-D fuel performance studies. In *Proceedings of PHYSOR 2020*, Illinois. ANS.

[66] Forrester, A. and Keane, A. J. (2007). Multi-variable geometry repair and optimization of passive vibration isolators. In *48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1923, Reston, VA. AIAA.

[67] Forrester, A., Sóbester, A., and Keane, A. J. (2008). *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, London.

[68] Forsberg, C., Richard, J., Pounders, J., Kochendarfer, R., Stein, K., Shwageraus, E., and Parks, G. (2015). Development of a fluoride-salt-cooled high-temperature reactor (FHR) using advanced gas-cooled reactor (AGR) technology. *Transactions of the American Nuclear Society*, 112:566–570.

[69] Forsberg, C., Wang, D., Shwageraus, E., Mays, B., Parks, G. T., Coyle, C., and Liu, M. (2019). Fluoride-salt-cooled high-temperature reactor (FHR) using british advanced gas-cooled reactor (AGR) refueling technology and decay heat removal systems that prevent salt freezing. *Nuclear Technology*, pages 1–16.

[70] François, J. and Lopez, H. (1999). SOPRAG: a system for boiling water reactors reload pattern optimization using genetic algorithms. *Annals of Nuclear Energy*, 26(12):1053–1063.

[71] Galperin, A. (1995). Exploration of the search space of the in-core fuel management problem by knowledge-based techniques. *Nuclear Science and Engineering*, 119(2):144–152.

[72] Galperin, A. and Nissan, E. (1988). Application of a heuristic search method for generation of fuel reload configurations. *Nuclear Science and Engineering*, 99(4):343–352.

[73] Gass, S. and Saaty, T. (1955). The computational algorithm for the parametric objective function. *Naval Research Logistics (NRL)*, 2(1-2):39–45.

[74] Giraud-Carrier, C. and Provost, F. (2005). Toward a justification of meta-learning: Is the no free lunch theorem a show-stopper. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 12–19, New York. Association for Computing Machinery.

[75] Glasstone, S. and Sesonske, A. (2004). *Nuclear Reactor Engineering: Reactor Systems Engineering: v. 2*. Litton Educational Publishing inc., New York.

[76] Glaz, B., Friedmann, P. P., and Liu, L. (2009). Helicopter vibration reduction throughout the entire flight envelope using surrogate-based optimization. *Journal of the American Helicopter Society*, 54(1):12007–12007.

[77] Gomez-Fernandez, M., Higley, K., Tokuhiro, A., Welter, K., Wong, W.-K., and Yang, H. (2020). Status of research and development of learning-based approaches in nuclear science and engineering: A review. *Nuclear Engineering and Design*, 359:110479.

[78] Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128.

[79] Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947.

[80] Hahnloser, R. H. and Seung, H. S. (2001). Permitted and forbidden sets in symmetric threshold-linear networks. In *Advances in Neural Information Processing Systems*, pages 217–223, San Diego. Neural Information Processing Systems Foundation, Inc., Neural Information Processing Systems Foundation, Inc.

[81] Haling, R. K. (1964). Operating strategy for maintaining an optimum power distribution throughout life. *UNT Libraries Government Documents Department*.

[82] Harrison, R., Startin, G., Lindley, B., Powney, D., Parks, G., and Hutt, P. (2016). Validation of WIMS/PANTHER PWR fuel reactivity depletion using the BEAVRS benchmark flux map data. In *Proceedings of Physor 2016*, volume 5, pages 2833–2846, Illinois. ANS.

[83] Herlihy, M. and Shavit, N. (2012). *The Art of Multiprocessor Programming, Revised Reprint*. Morgan Kaufmann.

[84] Hill, N. J. and Parks, G. T. (2015). Pressurized water reactor in-core nuclear fuel management by tabu search. *Annals of Nuclear Energy*, 75:64 – 71.

[85] Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507.

[86] Horelik, N., Herman, B., Forget, B., and Smith, K. (2013). Benchmark for evaluation and validation of reactor simulations (BEAVRS), v1.0.1. In *Proc. Int. Conf. Mathematics and Computational Methods Applied to Nuc. Sci. & Eng.*, Illinois. ANS.

[87] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[88] Hu, X., Chen, X., Parks, G. T., and Yao, W. (2016). Review of improved Monte Carlo methods in uncertainty-based design optimization for aerospace vehicles. *Progress in Aerospace Sciences*, 86:20–27.

[89] Hunter, J. S. (1989). Let's all beware the latin square. *Quality Engineering*, 1(4):453–465.

[90] International Organisation for Standardization (ISO) (2017). ISO/IEC 14882:2017 Programming Languages C++. Technical report, International Organization for Standardization.

[91] Intl. Atomic Energy Agency (IAEA) (2017). Power Reactor Information System (PRIS). https://www.iaea.org/PRIS/WorldStatistics/OperationalReactorsByType.aspx. Accessed: 2020-10-22.

[92] Izzo, D. (2012). Pygmo and pykep: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization). In *Proceedings of the Fifth International Conference on Astrodynamics Tools and Techniques, ICATT*, Paris. ESA.

[93] Izzo, D., Ruciński, M., and Biscani, F. (2012). The generalized island model. In *Parallel Architectures and Bioinspired Algorithms*, pages 151–169. Springer, Berlin.

[94] Jaeggi, D., Parks, G., Dawes, W., and Clarkson, J. (2008). Robust multi-fidelity aerodynamic design optimization using surrogate models. In *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 6052, Reston, VA. AIAA.

[95] James, G. (2004). *Advanced Modern Engineering Mathematics*. Pearson Prentice Hall, Harlow, third edition.

[96] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 675–678, New York. ACM.

[97] Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492.

[98] Jouppi, N. (2016). Google supercharges machine learning tasks with TPU custom chip. https://cloud.google.com/blog/products/gcp/google-supercharges-machine-learning-tasks-with-custom-chip. Accessed: 2020-07-30.

[99] Kaggle inc. (2014). Kaggle cats vs. dogs competition. https://www.kaggle.com/c/dogs-vs-cats. Accessed: 2020-07-30.

[100] Kaminsky, W. M. and Lloyd, S. (2004). *Scalable Architecture for Adiabatic Quantum Computing of Np-Hard Problems*, pages 229–236. Springer US, Boston, MA.

[101] Katzgraber, H. G., Hamze, F., Zhu, Z., Ochoa, A. J., and Munoz-Bauza, H. (2015). Seeking quantum speedup through spin glasses: The good, the bad, and the ugly. *Physical Review X*, 5(3):031026.

[102] Keane, A. J. (2003). Wing optimization using design of experiment, response surface, and data fusion methods. *Journal of Aircraft*, 40(4):741–750.

[103] Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., and Murthy, K. R. K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13(3):637–649.

[104] Keller, P. and Turinsky, P. (1996). FORMOSA-P multiple adaptive penalty function methodology. *Transactions of the American Nuclear Society*, 75:341–342.

[105] Kelly, J. (2018). A preview of Bristlecone, Google's new quantum processor. https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html. Accessed: 2020-07-30.

[106] Khurana, M., Winarto, H., and Sinha, A. (2009). Airfoil optimisation by swarm algorithm with mutation and artificial neural networks. In *47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 1278, Reston, VA. AIAA.

[107] Kim, H. G., Chang, S. H., and Lee, B. H. (1993a). Pressurized water reactor core parameter prediction using an artificial neural network. *Nuclear Science and Engineering*, 113(1):70–76.

[108] Kim, H. K., Lee, S. H., and Chang, S. H. (1993b). Neural network model for estimating departure from nucleate boiling performance of a pressurized water reactor core. *Nuclear technology*, 101(2):111–122.

[109] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv eprint arXiv:1412.6980*.

[110] Kipouros, T., Molinari, M., Dawes, W. N., Parks, G. T., Savill, M., and Jenkins, K. W. (2007). An investigation of the potential for enhancing the computational turbomachinery design cycle using surrogate models and high performance parallelisation. pages 1415–1424, New York. ASME.

[111] Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986.

[112] Knowles, J., Thiele, L., and Zitzler, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland.

[113] Kolda, T. G., Lewis, R. M., and Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM review*, 45(3):385–482.

[114] Kothe, D. B. (2010). CASL: the consortium for advanced simulation of light water reactors. *Bulletin of the American Physical Society*.

[115] Krauth, W. (2016). Statistical mechanics: Algorithms and computations. https://www.coursera.org/learn/statistical-mechanics. Accessed: 2020-10-22.

[116] Krige, D. G. (1951). *A statistical approach to some mine valuation and allied problems on the Witwatersrand*. PhD thesis, University of the Witwatersrand, Johannesburg.

[117] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, San Diego. Neural Information Processing Systems Foundation, Inc.

[118] Kropaczek, D. and Turinsky, P. (1995). In-core nuclear fuel management optimization for pressurized water reactors utilizing simulate annealing. *Nuclear Technology*, 95:9–32.

[119] Kruskal, W. H. and Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621.

[120] Lattarulo, V., Lindley, B. A., and Parks, G. T. (2014). Application of the MOAA for the optimization of CORAIL assemblies for nuclear reactors. *IEEE Congress on Evolutionary Computation (CEC)*.

[121] Laurenceau, J., Meaux, M., Montagnac, M., and Sagaut, P. (2010). Comparison of gradient-based and gradient-enhanced response-surface-based optimizers. *AIAA Journal*, 48(5):981–994.

[122] LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In Arbib, M., editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press.

[123] LeCun, Y., Bengio, Y., and Hinton, G. (2005). Deep learning. *Nature*, 521:436–444.

[124] LeCun, Y. et al. (2015). Lenet-5, convolutional neural networks. http://yann.lecun.com/exdb/lenet/. Accessed: 2020-07-30.

[125] Lehmann, E. L. (1997). *Testing statistical hypotheses*. Springer texts in statistics. Springer, New York ; London, 2nd ed. edition.

[126] Leniau, B., Mouginot, B., Thiolliere, N., Doligez, X., Bidaud, A., Courtin, F., Ernoult, M., and David, S. (2015). A neural network approach for burn-up calculation and its application to the dynamic fuel cycle code class. *Annals of Nuclear Energy*, 81:125 – 133.

[127] Leppänen, J. (2015). 2D PWR pin cell burnup example. http://serpent.vtt.fi/mediawiki/index.php/Collection_of_example_input_files. Accessed: 2020-07-30.

[128] Leppänen, J., Pusa, M., Viitanen, T., Valtavirta, V., and Kaltiaisenaho, T. (2015). The Serpent Monte Carlo code: Status, development and applications in 2013. *Annals of Nuclear Energy*, 82:142–150.

[129] Lewis, E. E. (2008). *Fundamentals of nuclear reactor physics*. Elsevier, Boston.

[130] Li, H. and Zhang, Q. (2009). Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302.

[131] Li, J., Gao, Z., Huang, J., and Zhao, K. (2013). Aerodynamic design optimization of nacelle/pylon position on an aircraft. *Chinese Journal of Aeronautics*, 26(4):850–857.

[132] Li, X. (2003). A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *Genetic and Evolutionary Computation — GECCO 2003*, pages 37–48, Berlin, Heidelberg. Springer Berlin Heidelberg.

[133] Li, Z., Jun, S., Chunlin, W., Zhe, S., and Qian, X. (2020). Research on the cross-section generating method in htgr simulator based on machine learning methods. In *Proceedings of PHYSOR 2020*, Illinois. ANS.

[134] Lian, Y. and Liou, M.-S. (2005). Multiobjective optimization using coupled response surface model and evolutionary algorithm. *AIAA Journal*, 43(6):1316–1325.

[135] Lindley, B., Newton, T., Hosking, J., Smith, P., Powney, D., Tollit, B., and Smith, P. (2015). Release of wims10: A versatile reactor physics code for thermal and fast systems. In *ICAPP 2015*, Vienna. IAEA.

[136] Lloyd, S. (2002). Computational capacity of the universe. *Nature*, Phys.Rev.Lett.88:237901.

[137] Lysenko, M. G., Wong, H.-I., and Maldonado, G. I. (1999). Neural network and perturbation theory hybrid models for eigenvalue prediction. *Nuclear science and engineering*, 132(1):78–89.

[138] MacCormack, A., Rusnak, J., and Baldwin, C. Y. (2006). Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science*, 52(7):1015–1030.

[139] Mahlers, Y. (1994). Core loading pattern optimization for pressurized water reactors. *Annals of Nuclear Energy*, 21(4):223 – 227.

[140] Maldonado, G. I. (2005). Optimizing lwr cost of margin one fuel pin at a time. *IEEE transactions on nuclear science*, 52(4):996–1003.

[141] Manring, C. A. and Hawari, A. I. (2020). Development of neural thermal scattering (nets) modules for reactor multi-physics simulations. In *Proceedings of PHYSOR 2020*, Illinois. ANS.

[142] Manteufel, R. (2000). Evaluating the convergence of Latin hypercube sampling. AIAA, Reston, VA.

[143] Marklund, P.-O. and Nilsson, L. (2001). Optimization of a car body component subjected to side impact. *Structural and Multidisciplinary Optimization*, 21(5):383–392.

[144] Martín-del Campo, C., François, J. L., Avendaño, L., and González, M. (2004). Development of a bwr loading pattern design system based on modified genetic algorithms and knowledge. *Annals of Nuclear Energy*, 31(16):1901–1911.

[145] Massaro, A., D'Andrea, A., and Benini, E. (2011). Multiobjective-multipoint rotor blade optimization in forward flight conditions using surrogate-assisted memetic algorithms. In *37th European Rotorcraft Forum*.

[146] Matheron, G. (1962). *Traité de géostatistique appliquée. 1 (1962)*, volume 1. Editions Technip, Paris.

[147] Mathur, V. K. (1991). How well do we know pareto optimality? *The Journal of Economic Education*, 22(2):172–178.

[148] Mathworks inc. (2019). Global optimisation toolbox documentation. https://uk.mathworks.com/help/gads/surrogate-optimization-algorithm.html. Accessed: 2020-07-30.

[149] Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30.

[150] McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245.

[151] Mhaskar, H., Liao, Q., and Poggio, T. (2016). Learning functions: When is deep better than shallow. *arXiv preprint arXiv:1603.00988*.

[152] Michalewicz, Z. (1998). *Genetic Algorithms plus Data Structures = Evolution Programs*. Springer-Verlag GmbH, Berlin.

[153] Modro, S. M., Fisher, J. E., Weaver, K. D., Reyes Jr., J. N., Groome, J. T., Babka, P., and Carlson, T. M. (2003). Multi-application small light water reactor final report, INEEL/EXT-04-01626. Technical report, Idaho National Engineering and Evnironmental Laboratory, Idaho National Engineering and Environmental Laboratory, Bechtel BWXT Idaho, LLC.

[154] Montgomery, A. A., Peters, T. J., and Little, P. (2003). Design, analysis and presentation of factorial randomised controlled trials. *BMC Medical Research Methodology*, 3(1):26.

[155] Moore, G. E. (1965). Cramming more components onto integrated circuits. Electronics 38 (8): 114–117. found in "Proceedings of the IEEE, vol. 86, No. 1, 1998".

[156] Morrison, E. A. (2003). *PANTHER User Guide E/REP/BB-DB/0015/GEN/03ED/PANTHER/UG/5.5*. British Energy Generation Limited, Engineering Division, Gloucester GL4 3RS.

[157] Murray-Rust, P. (2008). Open data in science. *Serials Review*, 34(1):52–64.

[158] Na, M. G., Shin, S. H., Lee, S. M., Jung, D. W., Lee, K., and Lee, Y. J. (2004). Estimation of axial dnbr distribution at the hot pin position of a reactor core using fuzzy neural networks. *Journal of nuclear science and technology*, 41(8):817–826.

[159] Naft, B. and Sesonske, A. (1971). Pressurized water reactor optimal fuel management. *Nuclear Technology*, 14:123–132.

[160] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of International Conference on Machine Learning (ICML)*.

[161] Neven, H., Denchev, V. S., Rose, G., and Macready, W. G. (2012). QBoost: Large scale classifier training with adiabatic quantum optimization. In *JMLR: Workshop and Conference Proceedings*, pages 333–348, Singapore. ACML.

[162] Nissan, E. (2019). An overview of AI methods for in-core fuel management: Tools for the automatic design of nuclear reactor core configurations for fuel reload, (re)arranging new and partly spent fuel. *Designs*, 3:1–45.

[163] Noda, H., Yamamoto, A., Nagasawa, Y., Murao, H., and Kitamura, S. (1997). Core burnup calculations using neural networks. In *Proc. Topl. Mtg. Advances in Nuclear Fuel Management II*, pages 20–39.

[164] Nonbøl, E. (1996). Description of the advanced gas cooled type of reactor (AGR). Technical report, Nordisk Kernesikkerhedsforskning.

[165] Nowacki, H. (1980). Modelling of design decisions for CAD. In *Computer Aided Design Modelling, Systems Engineering, CAD-Systems*, pages 177–223. Springer.

[166] Nowak, K., Märtens, M., and Izzo, D. (2014). Empirical performance of the approximation of the least hypervolume contributor. In *Parallel Problem Solving from Nature – PPSN XIII*, pages 662–671, Cham. Springer International Publishing.

[167] NuScale llc. (2011). NuScale power, llc. and Fluor corporation team up. https://newsroom.nuscalepower.com/press-releases/news-details/2011/NuScale-Power-LLC-and-Fluor-Corporation-Team-Up/default.aspx. Accessed: 2020-07-30.

[168] NuScale llc. (2019). NuScale design certification application, final safety analysis report (FSAR). Technical report, US Nuclear Regulatory Commission.

[169] NuScale llc. (2020). NuScale submits vendor design review. https://newsroom.nuscalepower.com/press-releases/news-details/2020/NuScale-Submits-Phase-1-and-2-Combined-Pre-Licensing-Vendor-Design-Review-to-Canadian-Nuclear-Safety-Commission/default.aspx. Accessed: 2020-07-30.

[170] Oberkampf, W. L. and Roy, C. J. (2010). *Verification and Validation in Scientific Computing*. Cambridge University Press.

[171] Ortiz, J. J. and Requena, I. (2004). An order coding genetic algorithm to optimize fuel reloads in a nuclear boiling water reactor. *Nuclear Science and Engineering*, 146(1):88–98.

[172] Palar, P. S., Tsuchiya, T., and Parks, G. T. (2016). A comparative study of local search within a surrogate-assisted multi-objective memetic algorithm framework for expensive problems. *Applied Soft Computing*, 43:1–19.

[173] Parks, G. T. (1988). *Optimal in-core nuclear fuel cycles under integral constraints*. PhD thesis, University of Cambridge, Cambridge.

[174] Parks, G. T. (1990). An intelligent stochastic optimization routine for nuclear fuel cycle design. *Nuclear Technology*, 89(2):233–246.

[175] Parks, G. T. (1996). Multiobjective pressurized water reactor reload core design by nondominated genetic algorithm search. *Nuclear Science and Engineering*, 124(1):178–187.

[176] Parks, G. T. (2017). 4M17: Practical optimisation lecture notes. University of Cambridge.

[177] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems*, San Diego. Neural Information Processing Systems Foundation, Inc.

[178] Peacock, J. A. (1983). Two-dimensional goodness-of-fit testing in astronomy. *Monthly Notices of the Royal Astronomical Society*, 202(3):615–627.

[179] Poggio, T. and Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247(4945):978–982.

[180] Poon, P. and Parks, G. (1993). Application of genetic algorithms to in-core nuclear fuel management optimization. In *Proceedings of the Joint International Conference on Mathematical Methods and Supercomputing in Nuclear Applications*, volume 1, page 777–786., Vienna. IAEA.

[181] Porwal, A., Carranza, E. J. M., and Hale, M. (2003). Artificial neural networks for mineral-potential mapping: A case study from Aravalli province, Western India. *Natural Resources Research*, 12(3):155–171.

[182] Prati, E., Rotta, D., Sebastiano, F., and Charbon, E. (2017). From the quantum Moore's law toward silicon based universal quantum computing. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–4, New York. IEEE.

[183] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2003). *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, Cambridge, second edition.

[184] Prometeus GmbH (2019). TOP500 supercomputers - University of Cambridge, 'Cumulus' HPC ranking.
https://www.top500.org/system/179577. Accessed: 2020-07-30.

[185] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., and Tucker, P. K. (2005). Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41(1):1–28.

[186] Radaideh, M. I., Price, D., and Kozlowski, T. (2020). Modelling nuclear data uncertainties using deep neural networks. In *Proceedings of PHYSOR 2020*, Illinois. ASN.

[187] Rauand, A. and Walters, W. (2019). Applications of Serpent's fission matrix capability. http://montecarlo.vtt.fi/mtg/2019_Atlanta/. Accessed: 2020-07-30.

[188] Razavi, S., Tolson, B. A., and Burn, D. H. (2012). Review of surrogate modeling in water resources. *Water Resources Research*, 48(7).

[189] Robinson, G. M. and Keane, A. J. (2001). Concise orthogonal representation of supercritical airfoils. *Journal of Aircraft*, 38(3):580–583.

[190] Rojas Gonzalez, S., Jalali, H., and Van Nieuwenhuyse, I. (2020). A multiobjective stochastic simulation optimization algorithm. *European Journal of Operational Research*, 284(1):212 – 226.

[191] Rummelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

[192] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science*, pages 409–423.

[193] Scandpower (now Studsvik Nuclear AB) (1989). *(not located by author) User Manual CM-Presto 9. Version CM914B*. Scandpower, SE-611 82 Nyköping.

[194] Schryen, G. and Kadura, R. (2009). Open source vs. closed source software: Towards measuring security. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, SAC '09, page 2016–2023, New York. ACM.

[195] Seshadri, P. and Parks, G. T. (2017). Effective-quadratures (EQ): Polynomials for computational engineering studies. *The Journal of Open Source Software*, 2(11):166.

[196] Shafer, J. and Fate, T. (2007). Coring and core analysis: Challenges of offshore ultra deep water reservoirs. In *International Symposium of the Society of Core Analysts*, Calgary.

[197] Shin, S. W., Smith, G., Smolin, J. A., and Vazirani, U. (2014). How 'quantum' is the D-Wave machine? *arXiv preprint arXiv:1401.7087*.

[198] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[199] Singer, D. A. and Kouda, R. (1996). Application of a feedforward neural network in the search for Kuroko deposits in the Hokuroku district, Japan. *Mathematical Geology*, 28(8):1017–1023.

[200] Skinner, S. and Zare-Behtash, H. (2018). State-of-the-art in aerodynamic shape optimisation methods. *Applied Soft Computing*, 62:933 – 962.

[201] Smith, K. S. (2016). Nodal diffusion methods: Understanding numerous unpublished details. In *Proceedings of PHYSOR 2016*, PHYSOR. ANS.

[202] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

[203] Smola, A. J., Schölkopf, B., and Müller, K.-R. (1998). The connection between regularization operators and support vector kernels. *Neural Networks*, 11(4):637–649.

[204] Šmuc, T., Pevec, D., and Petrović, B. (1994). Annealing strategies for loading pattern optimization. *Annals of nuclear energy*, 21(6):325–336.

[205] Sobes, V., Hiscox, B., Popov, E., Delchini, M., Archibald, R., Hauck, C., Laiu, P., Betzler, B., and Terrani, K. (2020). Artificial intelligence design of nuclear systems empowered by advanced manufacturing. In *Proceedings of PHYSOR 2020*, Illinois. ANS.

[206] Sóbester, A., Leary, S. J., and Keane, A. J. (2005). On the design of optimization strategies based on global response surface approximation models. *Journal of Global Optimization*, 33(1):31–59.

[207] Sobol, I. M. (1979). On the systematic search in a hypercube. *SIAM Journal on Numerical Analysis*, 16(5):790–793.

[208] Stacey, W. M. (2007). *Nuclear Reactor Physics*. Wiley VCH Verlag GmbH.

[209] Stevens, J., Smith, K., Rempe, K., and Downar, T. (1995). Optimization of pressurized water reactor shuffling by simulated annealing with heuristics. *Nuclear Science and Engineering*, 121(1):67–88.

[210] Szames, E., Ammar, K., Tomatis, D., and Martinez, J. M. (2020). Few-group cross sections modeling by artificial neural networks. In *Proceedings of PHYSOR 2020*, Illinois. ANS.

[211] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, New York. IEEE.

[212] Tabachnick, B. G. and Fidell, L. S. (2000). *Using Multivariate Statistics (4th Edition)*. Allyn & Bacon, Boston.

[213] Tabatabaei, M., Hakanen, J., Hartikainen, M., Miettinen, K., and Sindhya, K. (2015). A survey on handling computationally expensive multiobjective optimization problems using surrogates: Non-nature inspired methods. *Structural and Multidisciplinary Optimization*, 52(1):1–25.

[214] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A survey on deep transfer learning. In *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 270–279, Cham. Springer International Publishing.

[215] Tang, B. (1993). Orthogonal array-based Latin hypercubes. *Journal of the American Statistical Association*, 88(424):1392–1397.

[216] Tano, M. and Ragusa, J. C. (2020). Using artificial neural networks to accelerate transport solves. In *Proceedings of PHYSOR 2020*, Illinois. ANS.

[217] Telgarsky, M. (2015). Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*.

[218] TIOBE Group (2020). TIOBE index for ranking the popularity of programming languages.
https://www.tiobe.com/tiobe-index/. Accessed: 2020-07-30.

[219] Turinsky, P. J. (2005). Nuclear fuel management optimisation: A work in progress. *Nuclear Technology (American Nuclear Society)*.

[220] Turinsky, P. J. and Parks, G. T. (1999). Advances in nuclear fuel management for light water reactors. In *Advances in Nuclear Science and Technology*, volume 26, pages 137–165. Springer, Cham.

[221] Ulam, S. M. (1961). *Monte Carlo calculations in problems of mathematical physics*, pages 261–281. McGraw-Hill, New York.

[222] U.S. Atomic Energy Commission (1955). *The Reactor Handbook, volume 1: Physics*. McGraw-Hill, New York.

[223] Van Rossum, G. (1995). Python tutorial, cs-r9526. Technical report, Centrum voor Wiskunde en Informatica (CWI), Amsterdam.

[224] Vidal, J.-M., Eschbach, R., Launay, A., and Binet, C. (2012). CESAR5. 3: an industrial tool for irradiated nuclear fuel and waste characterisation, with an associated qualification-59080. In *ICEM 2011: 14. International conference on Environmental Remediation and Radioactive Waste Management*.

[225] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272.

[226] Waldrop, M. M. (2016). The chips are down for Moore's law. *Nature News*, 530(7589):144.

[227] Wang, G. G. and Shan, S. (2007). Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380.

[228] Wang, J., Wang, Q., and Ding, M. (2020). Review on neutronic/thermal-hydraulic coupling simulation methods for nuclear reactor analysis. *Annals of Nuclear Energy*, 137:107165.

[229] Weihermiller, W. and Allison, G. (1979). LWR nuclear fuel bundle data for use in fuel bundle handling. Technical report, Battelle Pacific Northwest Labs.

[230] Werbos, P. J. (1994). *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*. John Wiley & Sons, London.

[231] Weyland, D. (2015). A critical analysis of the harmony search algorithm—how not to solve sudoku. *Operations Research Perspectives*, 2:97 – 105.

[232] Whitley, D. and Watson, J. P. (2005). *Complexity Theory and the No Free Lunch Theorem*, pages 317–339. Springer US, Boston, MA.

[233] Whyte, A. (2020). Figures for PHYSOR 2020.
Zenodo: DOI10.5281/zenodo.3446287. Accessed: 2020-07-30.

[234] Whyte, A. and Parks, G. (2020a). Quantum annealing optimization of a heuristic surrogate model for pwr fuel loading. In *Proceedings of PHYSOR 2020*, Illinois. ANS. (engrxiv.org/xzmby).

[235] Whyte, A. and Parks, G. (2020b). Surrogate model optimization of a 'micro core' pwr fuel assembly arrangement using deep learning models. In *Proceedings of PHYSOR 2020*, Illinois. ANS. (engrxiv.org/xzmby).

[236] Whyte, A., Xing, Z., Parks, G., and Shwageraus, E. (2019). Design of a deep learning surrogate model for the prediction of fhr design parameters. In *Proc. Int. Conf. Mathematics and Computational Methods Applied to Nuc. Sci. & Eng.*, Illinois. ANS. ISBN: 978-0-89448-769-9.

[237] Wilson, P. R. (2020). *Design Recipes for FPGAs*. Elsevier, Oxford.

[238] Winston, P. (2010). 6.034 Artificial Intelligence, MIT OpenCourseWare.
https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/##. Accessed: 2020-10-22.

[239] Wittek, P. (2016). *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Elsevier, San Diego.

[240] Wittek, P. (2019). Quantum computing and artificial intelligence.
https://www.edx.org/course/quantum-machine-learning. Accessed: 2020-07-30.

[241] Wolpert, D. and Macready, W. (1997). No free lunch theorem for optimization. *IEEE Transactions on Evolutionary Computation*, 1:467–482.

[242] Wu, X., Kozlowski, T., and Meidani, H. (2018). Kriging-based inverse uncertainty quantification of nuclear fuel performance code BISON fission gas release model using time series measurement data. *Reliability Engineering & System Safety*, 169:422–436.

[243] Xing, Z. (2019). *Design Space Exploration for Salt Cooled Reactor Systems*. PhD thesis, University of Cambridge, Cambridge.

[244] Xing, Z. and Shwageraus, E. (2017). Design space exploration studies of an FHR concept leveraging AGR technologies. In *ICAPP 2017*, Vienna. IAEA.

[245] Xing, Z. and Shwageraus, E. (2018a). Investigation into reactivity feedback of FHR designs with alternative coolants. In *ICAPP 2018*, Vienna. IAEA.

[246] Xing, Z. and Shwageraus, E. (2018b). Molten salt coolant reactivity feedback in alternative FHR designs. In *Proceedings of PHYSOR 2018*, Illinois. ANS.

[247] Yamamoto, A. (1997). A quantitative comparison of loading pattern optimization methods for in-core fuel management of pwr. *Journal of Nuclear Science and Technology*, 34(4):339–347.

[248] Yang, R. J., Wang, N., Tho, C. H., Bobineau, J. P., and Wang, B. P. (2005). Metamodeling development for vehicle frontal impact simulation. *Journal of Mechanical Design*, 127(5):1014–1020.

[249] Yoshiaki, O. and Kiguchi, T., editors (2014). *Nuclear Reactor Design: An Advanced Course in Nuclear Engineering*. Springer, Tokyo.

[250] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, San Diego. Neural Information Processing Systems Foundation, Inc.

[251] Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731.

[252] Zhang, Q., Zhang, J., Liang, L., Li, Z., and Zhang, T. (2020). A deep learning based surrogate model for estimating the flux and power distribution solved by diffusion equation. In *Proceedings of PHYSOR 2020*, Illinois. ANS.

[253] Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301, New York. Springer.

Appendix A

Neutron Transport Equation and Fission Matrix

A.1 Derivation of the Fission Matrix from Neutron Transport

In this section a brief discussion of the derivation of the fission matrix from the neutron transport equation is presented. This derivation is based on the explanations in [30] and [31] although the use and recognition go back further. This derivation will use the following nomenclature:

E	is the neutron Energy (MeV)
$\vec{r}$	is the cartesian position vector
$\hat{\Omega}$	is the direction vector (radians)
Σ_T	the absorption cross-section (barns)
Σ_S	The scattering cross-section
Σ_F	The fission cross-section
Ψ	is the angular neutron flux ($n/(cm^2 sradian)$)
ν	mean number of neutrons per fission event
K	the effective multiplication factor
χ	fission emission spectrum
δ	the Dirac-delta function (an infinitely short unit impulse at t_0)
$\mathbf{F}$	The fission matrix

The neutron transport equation is a balance equation describing the flux of neutrons, in terms of gains and losses. For a multiplicative medium with no external source, it will be generated

from a 'loss operator', defined as follows:

$$\mathbf{M}\cdot\Psi(\vec{r},E,\hat{\Omega}) = \underbrace{\hat{\Omega}\cdot\nabla\Psi(\vec{r},E,\hat{\Omega})}_{leakage} + \underbrace{\Sigma_T(\vec{r},E)\Psi(\vec{r},E,\hat{\Omega})}_{absorption} - \underbrace{\iint dE'd\hat{\Omega}'\Sigma_S(\vec{r},E'\to E,\hat{\Omega}'\to\hat{\Omega})\Psi(\vec{r},E',\hat{\Omega}')}_{scattering}$$

and a 'source term', which can be defined as :

$$S(\vec{r}) = \underbrace{\iint dE'd\nu\hat{\Omega}'\Sigma_F(\vec{r},E')\Psi(\vec{r},E',\hat{\Omega}')}_{fission} \tag{A.1}$$

This allows the neutron transport equation to be expressed as:

$$\mathbf{M}\cdot\Psi(\vec{r},E,\hat{\Omega}) = \frac{1}{K}\frac{\chi(E)}{4\pi}S(\vec{r}) \tag{A.2}$$

Where the source has an impulse response $H()$

$$H(\vec{r}_0\to\vec{r}) = \iint\iint dEd\hat{\Omega}dE_0d\hat{\Omega}_0\nu\sum_F(\vec{r},E)\cdot\frac{\chi(E_0)}{4\pi}G(\vec{r}_0,E_0,\hat{\Omega}_0) \tag{A.3}$$

and Ψ's impulse response, $G()$

$$\mathbf{M}\cdot G(\vec{r}_0,E_0,\hat{\Omega}_0\to\vec{r},E,\hat{\Omega}) = \delta(\vec{r}-\vec{r}_0)\delta(E-E_0)\delta(\hat{\Omega}-\hat{\Omega}_0) \tag{A.4}$$

then by superposition

$$\Psi(\vec{r},E,\Omega) = \frac{1}{K}\iiint d\vec{r}_0dE_0d\hat{\Omega}_0\frac{\chi(E_0)}{4\pi}\cdot S(\vec{r}_0)G(\vec{r}_0,E_0,\hat{\Omega}_0\to\vec{r},E,\hat{\Omega}) \tag{A.5}$$

For the fission matrix the problem is split into regions, and integrated over each region. The region of origin is J whose boundary condition is $\vec{r}_0\varepsilon V_J$ and the region of absorption is I, with limit $\vec{r}\varepsilon V_I$.

$$S_I = \frac{1}{K}\sum_{J-1}^{N}F_{I,J}S_J \tag{A.6}$$

where

$$F_{I,J} = \int_{\vec{r}\varepsilon V_I}d\vec{r}_0\frac{S(\vec{r}_0)}{S_J}H(\vec{r}_0\to\vec{r}) \tag{A.7}$$

$$S_J = \int_{\vec{r}'\varepsilon V_J}S(\vec{r}')d\vec{r}' \tag{A.8}$$

So the full matrix form of the fission matrix is:

$$\vec{S}_{t_1} = \frac{1}{K}\mathbf{F}\vec{S}_{t_0} \tag{A.9}$$

Appendix B

Supporting Technical Studies

B.1 Generating a Sampling Plan that Maximises the Variance of Means

When training surrogate models on sampled data of a multivariate system, the *central limit theorem* explains why non uniform distributions of mean arise, as mentioned in Section 2.2.1. In this subsection an attempt is made to control the input values such that the mean is as uniformly varied as possible. Note that the reason that the mean of uniform variables tends to a Gaussian distribution is that the *density* of solutions is higher in the centre of the distribution.

In this example nine sample values are selected in the range 0.8 to 5.0 in discrete steps of 0.2 to form an *experiment*. This range and discretisation of the sample set is analogous to the range of discrete enrichments used in Section 4.4, where nine values will be sampled per experiment. Mean value of the samples might affect the system response, for example mean enrichment is known to affect cycle length, the aim of the sampling plan is to create a set of random samples that maximises the variance of the mean.

Just as the Fourier transform converts an arbitrary function into a sum of weighted sine waves, the Weierstrass transform converts a function into a set of weighted Gaussians. In this section, a *Rectangular function (*$\Pi(x)$*)* is used. $\Pi(x)$ is a function similar to the uniform distribution, but is defined at all points. The rectangular function, defined as

$$\Pi(x) = \begin{cases} 0.0 & \text{if } |x| > 0.5 \\ 0.5 & \text{if } |x| = 0.5 \\ 1.0 & \text{if } |x| < 0.5 \end{cases} \tag{B.1}$$

will be converted into the expected range by translation and scale

$$f(y) = \Pi\left(\frac{y-2.9}{4.4}\right) \tag{B.2}$$

thus

$$f(y) = \begin{cases} 0 & \text{if } y < 0.7 \\ 0 & \text{if } y > 5.1 \\ 0.5 & \text{if } y = 0.7 \cup y = 5.1 \\ 1 & \text{if } y > 0.7 \cap y < 5.1 \end{cases} \tag{B.3}$$

The aim of the experiment is to to sample them to obtain a net distribution of mean values with maximum range. Equation B.3 is transformed using the Weierstrass transform into a sum of Gaussians, the basis function with the largest magnitude are then used as the scaling factors for the constituent basis functions of the distribution.

$$\begin{aligned} \mathbb{W}(x) &= \frac{1}{\sqrt{4\pi}} \int_{-\infty}^{\infty} f(y) e^{-\frac{(x-y)^2}{4}} dy \qquad \text{(B.4)} \\ \mathbb{W}(x) &= \frac{1}{\sqrt{4\pi}} \int_{-\infty}^{\infty} f(x-y) e^{-\frac{(y)^2}{4}} dy \\ \mathbb{W}(x) &= \frac{1}{\sqrt{4\pi}} \left(0 + \left[2xe^{-\frac{y^2}{4}} + 2e^{-\frac{y^2}{4}}\right]_{0.7}^{5.1} + 2 * 0.5\right) \\ \mathbb{W}(x) &= \frac{1}{\sqrt{4\pi}} \left(\left[2xe^{-6.5025} + 2e^{-6.5025}\right] - \left[2xe^{-0.1225} + 2e^{-0.1225}\right] + 1\right) \\ \mathbb{W}(x) &= 0.50170425075x - 6.26176906068 \end{aligned}$$

In Figure B.1 we can see the results of combining three random distributions, to generate a set of randomly selected experiments with a roughly flat frequency of mean values. The system uses three distributions in a ratio of $0.3 : 0.4 : 0.3$, based on the first positive values of the Gaussian domain function. Triangular random distributions to select the samples for the outer sets of experiments are used, since they are selecting 9 values. The results approximate a Gaussian with a mean at lower and upper intervals ($\mu \approx 2.1$ and $\mu \approx 4.7$ respectively). The central distribution is a Gaussian with a standard deviation of one. The result is a random set that has a similar frequency for a wide range of mean values. The problem of repeated patterns is most obvious when considering the highest and lowest mean values (in this case 9 samples at 5, or 9 samples at 0.8). Here there is only one permutation of the design; the next value $\mu = 4.9\dot{7}, or \mu = 0.83\dot{3}$, have only 9 permutations; however, this quickly explodes. As the permutations increase the chance of a repeated a experiment decreases, however the density of experiments decreases with respect to the number of permutation at that mean input vector.

Note from Figure B.2 that a very small number of experiments are repeated (0.00094%) due to the high dimensionality of the problem.

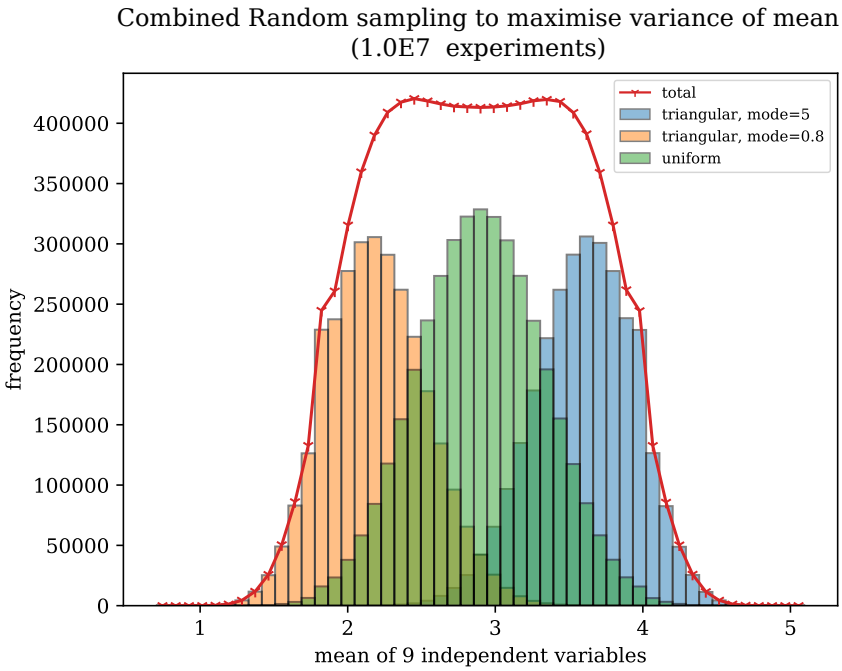


Fig. B.1 Combining Gaussian and triangular random distributions to create a set of inputs that has a high mean variance.

Consideration of the experiment and number of permutations at, and close to, the limits of the mean of the input vector should confirm to the reader that the statistics shown in Figure B.2 are representative despite the high level of stochastic noise.

Despite efforts to fight the *central limit theorem*, only a small deviation is achieved for nine independent random variables. For this reason, the approach is not used in further work. One approach is to accept the *central limit theorem* and expect that training sets will have uniform random samples for each input variable and a normally distributed sum, or to use the Sobel sequence and achieve a known space coverage.

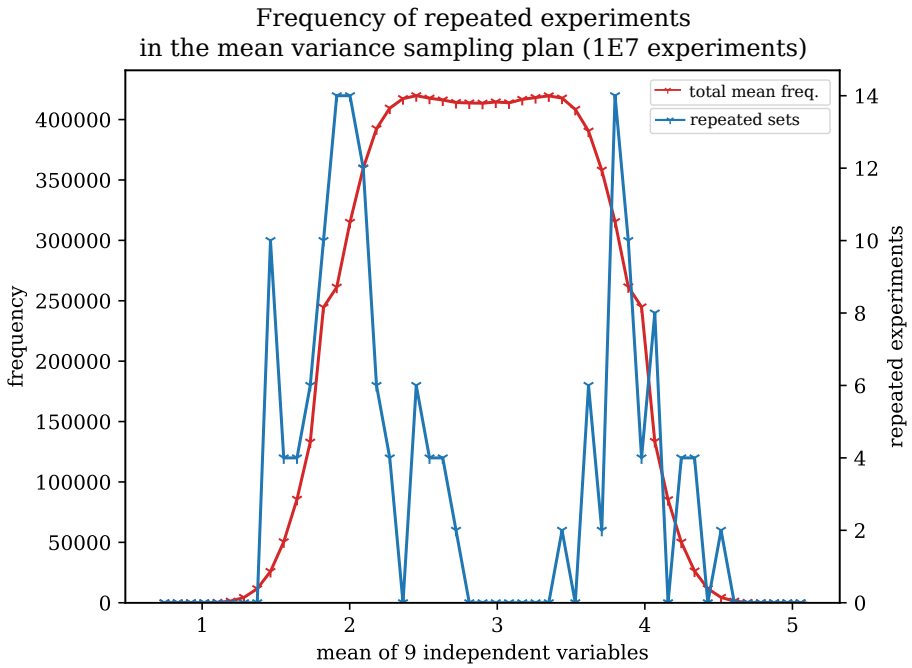


Fig. B.2 Checking the frequency of repeated sample plans in the combined Gaussian and triangular random distributions.

B.2 NSGA 2 Parametrisation Study for the 6×6 Microcore

B.2.1 NSGA2 studies in literature

The author examined work by Deb et al. [50] Li and Zhang [130] and Latterulo et al,[120].

The parameters from these studies and the Pagmo2 API'S default values are shown in in table B.1 A fully exhaustive study of the parameter variation with respect to every value was

Table B.1 NSGA2 parameters

Parameter	**Description**	**Pagmo default**	**NSGA2 references**		
			[50]	**[130]**	[120]
P	Size of the population	-	100	300 [a]	100
N	Number of generations	-	250	500	3, 12, 16, 20
cr	Crossover probability	0.95	0.9	0.1-1.0	-
m	Mutation probability	0.01	$1/n_{vars}$	-	$1/n_{vars}$
seed	Seed of internal RNG	clock	-	-	-

[a] 595 used for three objectives

not possible. However, based on an investigation of the parameter values used in the other

scientific literature. A simplified 'compass search' style method was used to maximise the mean hypervolume parameter over a number of runs with adjacent seed values. Although the origin of compass search is unclear, an early use in digital computers was by Fermi and Metropolis to establish scattering cross sections (see Anderson et al, [10] found in Kolda et al[113])

In order to compare the efficacy of a multiobjective optimisation runs the hypervolume parameter is used. Nowak et al. [166] attribute hypervolume parametrisation of multiobjective optimisation to Zitzler et al. [253].

Firstly, the population size, P and number of generations, N were identified as the most computationally expensive variables to investigate and were varied first with the other variables set to the Pagmo default values. Once suitable values of this had been established from this study the other variables were tested for the chosen population and number of generations.

Table B.2 Parameters used in this study

Parameter	**Candidate Values**
N	1 - 100
P	20, 32, 40, 60, 80, 100, 120
cr	0.7, 0.8, 0.9, 0.95, 0.9999
m	0.001, 0.01, 0.05, 0.1
seed(selection):	3453412-6
seed(population):	3453213-7

B.2.2 Number of generations

The first test ran a population of 20 individuals, for 100 generations. At such a small population size, the system is unlikely to converge to the global optimum, however the aim of the experiment is to establish how quickly the algorithm converges when acting on this problem. For this reason Figure B.3 shows the percentage increase of the hypervolume parameter as the metric for improvement while Figure B.4 shows the actual values.

The hypervolumes plotted in figure B.3, show that for this problem, the NSGA2 algorithm running for 40 generations, has an average hypervolume indicator that is 97.735% to $3d.p.$ of the maximum hypervolume achieved after a further 60 generations, furthermore figure B.4 shows that there is a full correlation between low rate of convergence runs and poor overall performance, which is significant since the end goal of an optimisation problem is simply the absolute best solution found, rather than the mean solution found. These results are used to justify the investigation of subsequent optimisation problems for at least 40 generations.

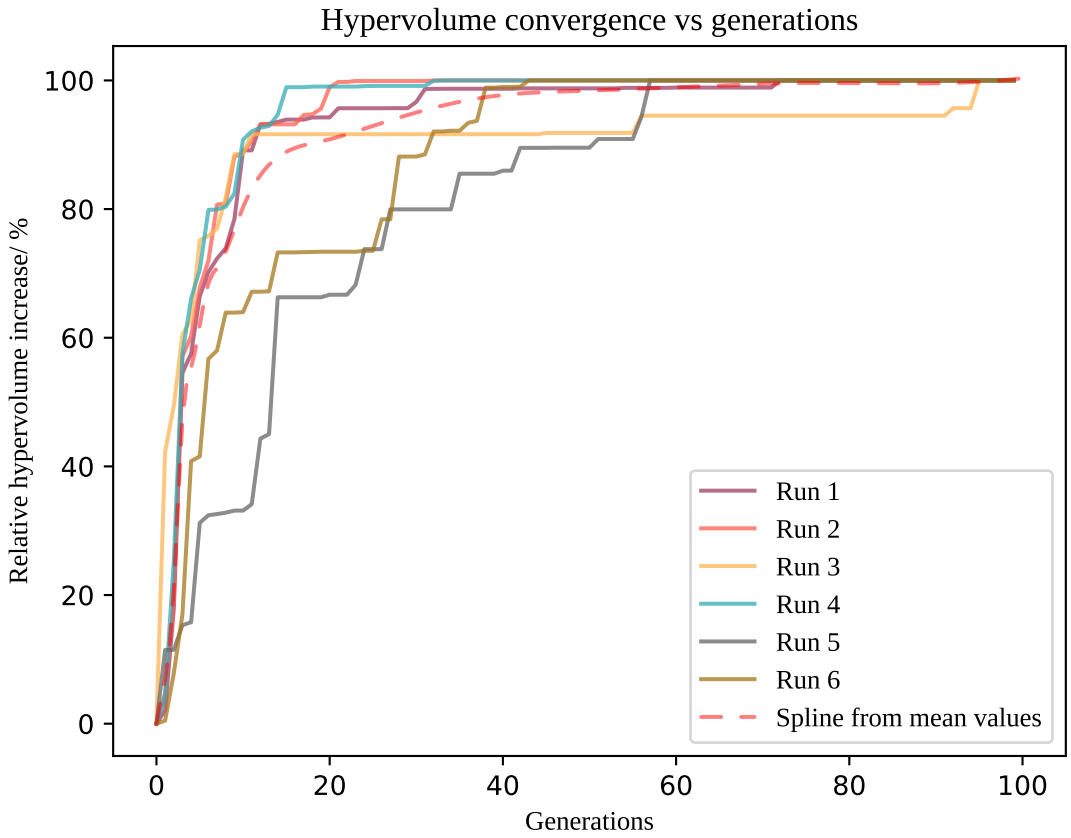


Fig. B.3 Relative hypervolume indicator plot for NSGA2 over generations.

B.2.3 Populations size

The next test investigates the size of the population, this was tested for 40 generations for a population size (*pop*) over a range of 20 to 120 individuals, due to constraints of the NSGA2 algorithm, only population numbers divisible by 4 are possible, so the values chosen were: 20,32,40,60,80,100,120. Although 120 individuals is still less than that of Li et al. [130], it is similar to other studies and represents a vast improvement in terms of number of total simulations compared to other studies.

Population size has the primary effect of decreasing the likelihood of the final solution being in a local optima. This means that while the actual mean hypervolume indicator achieved is important the variance of the results are also significant.

For this reason the results shown in Figure B.5 plot the average value of hypervolume relative to the overall average on the dashed line, then two volumes represent the mean value above the line and below the line and the max values achieved.

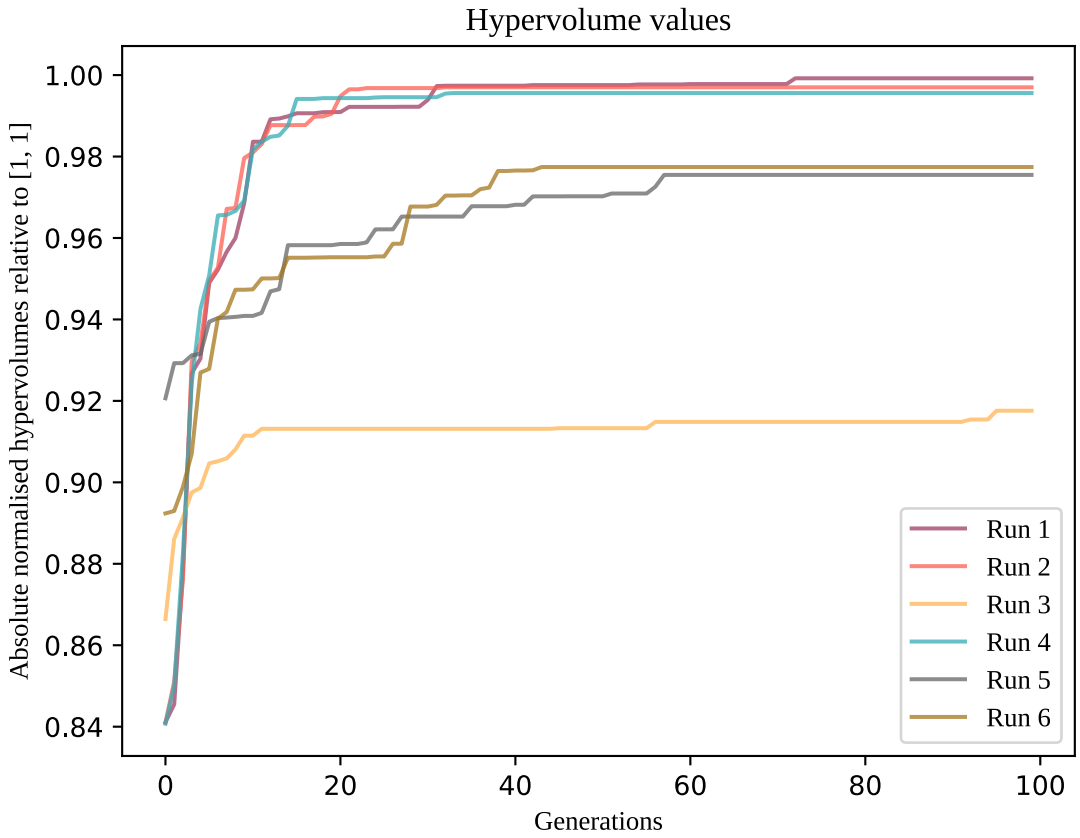


Fig. B.4 Absolute hypervolume indicator plot for NSGA2 over generations.

From this graph it can be seen that there is a positive correlation between the number of individuals in the population, P, and the value of the final hypervolume. However, the number of individuals has a large effect on the final computational budget and the gains achieved decrease as the population size goes up. The effect of population size increases is very small, of the order of 0.1% in this problem. For this reason, a population size of 50 is chosen for further experiments.

B.2.4 Population P and generations N

It should be noted that for each of the parameters so far there have been two implicit considerations. Namely that the use of more generations and larger populations cost computational time, the aim of this study is to establish the lowest cost that is effective with the NSGA2 algorithm applied to the kind of problems that are being investigated. Although the results of the previous section show that a larger population produces a larger expected hyperparameter, the increase

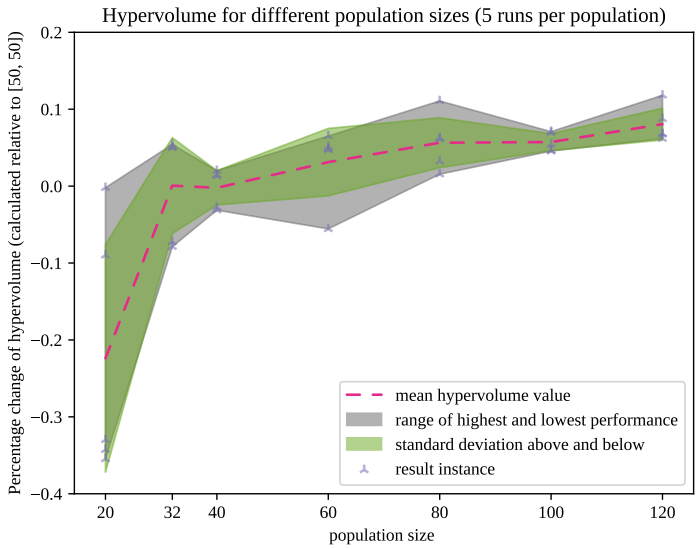


Fig. B.5 Relative final hypervolume indicator vs population size B.4.

is very small, with this in mind it becomes unfair to compare NSGA2 with a surrogate model in terms of cost, when using a very large populations size for the direct NSGA2 simulations, since we know that the expected gains will be small and the cost large.

Furthermore, due to the stability of the Monte Carlo simulations $\approx 0.1\%$ of simulations fail. This means that as N and P increase the number of simulations required increases at $N \times P$. Eventually it becomes cumbersome to run simulations that must be repeatedly restarted.

The subsequent variables do not have these implicit costs, and so can be varied without 'knock on' effects of the choices.

B.2.5 Crossover *cr* and mutation rate *m*

From the study of crossover values it can be seen that lower crossover values result in a larger range of hypervolumes, this is attributed to the low rate of crossover causing higher specialisation in the optimisation algorithm (Figure B.6)). Based on this result, the default value of 0.095 is selected for use in the rest of the study, since this gives a low range of results in the test, while still giving a large value for average evolved hypervolume.

Figure B.7 shows the effect that mutation rate has upon the average hypervolume evolved from the study. While there is a clear correlation between the mutation rate and the hypervolume evolved, and the range of the hypervolumes evolved reduces with the increase in mutation rate, however it is about ten times smaller than the total range of hypervolume evolved. Due to the

small effect, the default mutation rate of 1% is chosen for moving forwards, this is considered acceptable, since the range of hypervolume change due to mutation is less than 0.1%

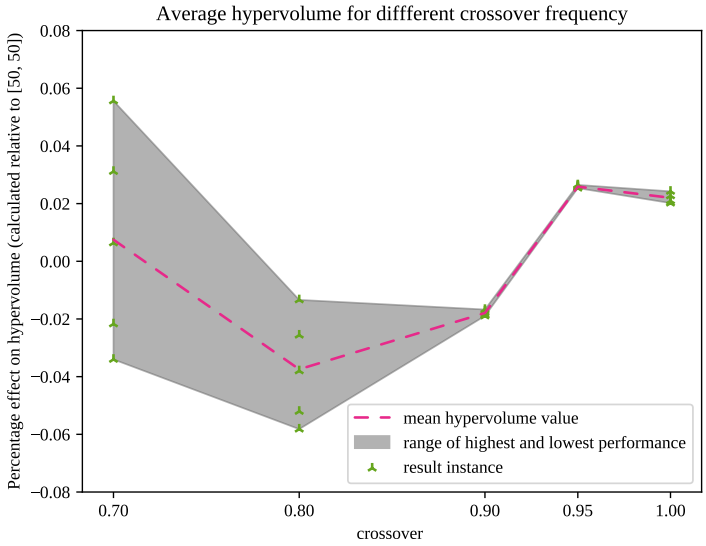


Fig. B.6 Hypervolume indicator vs Crossover rate.

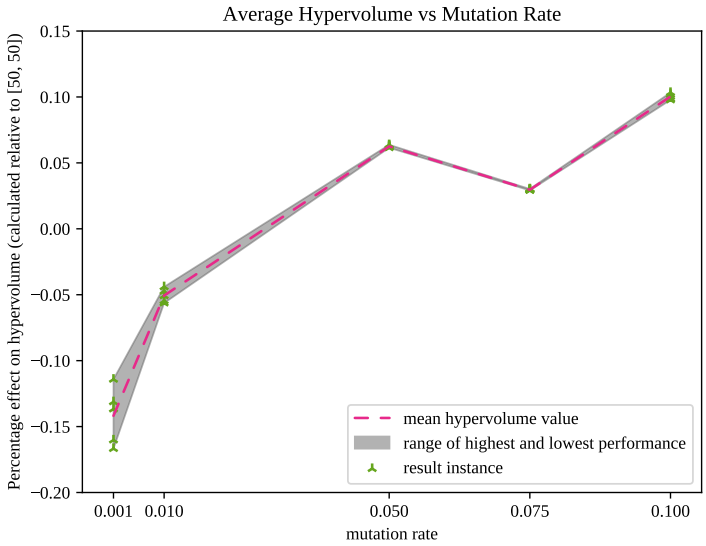


Fig. B.7 Hypervolume indicator vs mutation rate.

B.2.6 Review

The conclusion of this section is that the NSGA2 algorithm is robust to a wide variety of hyperparameter changes. That is to say that it is impossible to argue that the algorithm is particularly tuned to the *SMO* and detuned to the *DSO*. This is significant because it allows the evaluation side by side evaluation of the *SMO* and *DSO*. The NSGA2 algorithm was expressly designed to be robust to a wide variety of problems [50].

Another aim this study has been to find the parameters that represent the fairest set of parameters for use of the NSGA2 algorithm as a control optimisation method, to compare *DSO* with *SMO*. This puts the author into a state of moral hazard for parameters that have a computational cost associated with them, *e.g.* number of generations and population size, where a greater value yields better hypervolume performance, but at an increased computational cost. It would be legitimate but misleading to argue that the NSGA2 algorithm requires exceptionally large populations or extended number of generations, the argument could then easily be made that surrogate model methods represent a huge improvement on this classical algorithm.

The opposite position has been taken, to select parameters for the NSGA2 algorithm parameters that represent a realistic choice. This has been based on asking what an engineer might choose were they working with a limited budget for computational cost.

This basic 'compass search' in four dimensions, has required more than 236, 000 CPU hours (equivalent to twenty six years, and eight months of time on a single CPU time) and more than four months of continuous calender time queueing jobs on the Cambridge university HPC (ranked 75 in world HPCs at the time of this study [184]). The choice to act frugally towards the computational budget immediately makes proving the advantages of *SMO* more difficult, but hopefully lends credibility to the conclusion.

Crucially the study has shown that there is only a small increase in performance for NSGA2 on this problem between a population of forty and a population of one hundred and twenty. In this range, for a given computational budget, it is equivalent to run more simulations on a smaller population then to optimise with larger populations.

The conclusion of this study is that most of the hyperparameters make very little difference to the results in terms of average hypervolume of the optimisation for this problem.

Table B.3 is used to recapitulate the selected values of the parameters used in this study.

B.3 Kolmogorov-Smirnov Test

A Kolmogorov-Smirnov (K-S) Test has been carried out for the *DSO* and *SMO NDFs* in section 5.5. In order to use the K-S Test, modifications are made to establish a bivariate investigation.

Table B.3 Optimal parameters used in further work

Parameter	**Preferred Value(s)**
N	$\geqslant 60$
P	$\geqslant 40$
cr	0.95
m	0.01

This was carried out according Peacock [178] and investigated and shown to be applicable for practical purposes without assumptions about the distribution by Fasano and Franceschini [61].

Table B.4 K-S Test for *DSO* and *SMO*, against the rest of the runs and ensemble of runs vs categories.

K-S Test	**DSO** p-value	**SMO** p-value
run vs ensemble	0.260419	0.570637
	0.995971	0.772541
	0.004182	0.854263
	0.536309	0.737902
	0.619042	0.863347
mean	**0.4831846**	**0.759738**
mean DSO vs SMO:	**0.3170356**	

B.3.1 Review

The K-S Test is introduced in Section 1.2.4, was expected to be a useful measure of the difference of statistical results. Table B.4 shows the 'p-value' output of the K-S Test, this is the likelihood that the two sets of statistics were generated by the same distribution, *regardless of the distribution*. There is significantly more correlation between the *NDF* sets with multiple runs than between the *DSO* and *SMO*. However, in this application, the correlation is too low to be a useful test, as noted by Fasano and Francheschini [61], significant error is evolved when the correlation is low. Also, the results show a high variance.

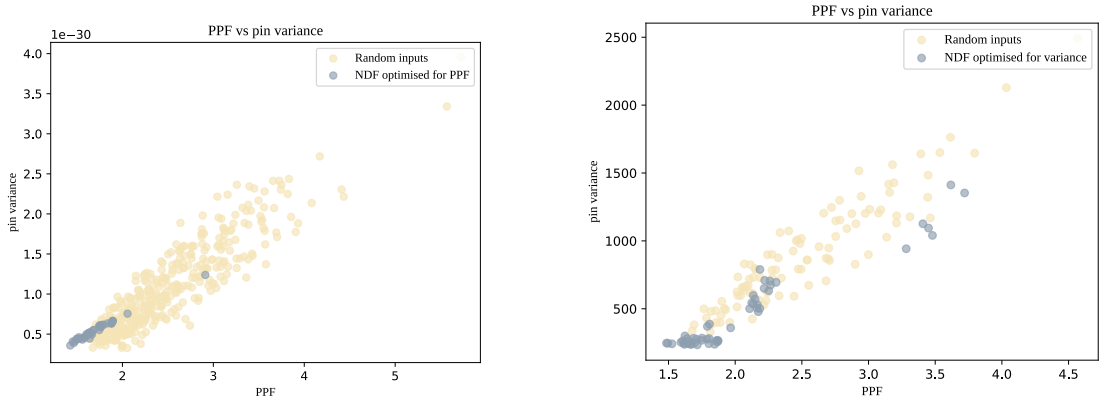
The K-S test also does not apply to categorical systems with a high likelihood of repeated solutions [4]. Although the search space for this experiment is $\approx 5.12 \times 10^{1}1$ combinations of inputs, the true *NDF* of this space is much smaller, and the likelihood of repeated values in this space is potentially high. This is noted to be a problem by [125, p. 584] and is proposed as an explanation for the low correlation of iterations of the same method, in the results.

B.4 PPF or pin power variance

In an initial version of Chapter 5, experiment 3 (p. 93), pin power variance was used as an objective variable. Variance has reduced error when calculated using a Monte Carlo method compared to *PPF* which has an error dependent on the peak power pin. *PPF* is in fact a proxy variable used in optimisation to represent *DNBR*, since the plant can be operated to a point where the peak pin departs from nucleate boiling (with a safety margin). Other reasons why a nuclear engineer might be interested in the variance of the pin powers would be if the uniformity of the coolant outlet temperature. However due to the ambiguity of the value of the pin power variance, it is presented in the appendix rather than the chapter.

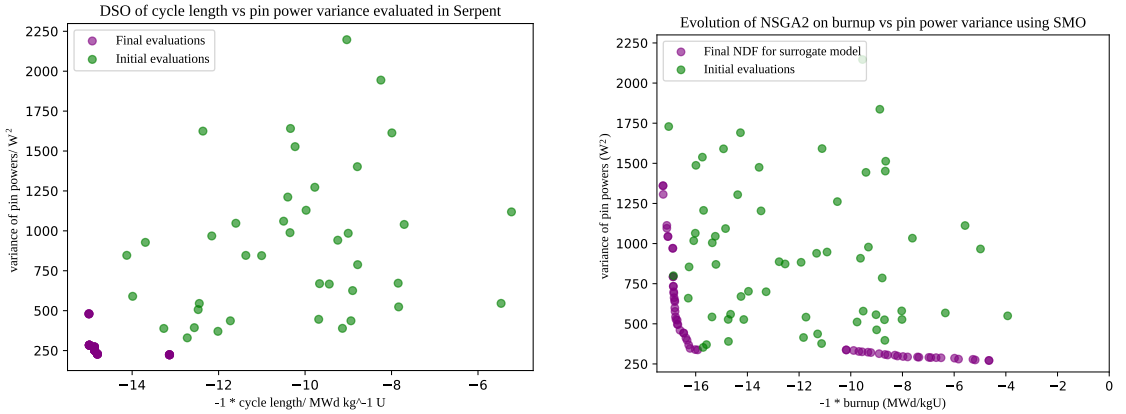
An initial investigation of pin power variance from indicates that *PPF* and variance in random *LPs* had a broad correlation, while at the *NDF* the correlation was very good, as shown in Figure B.8b. This lead to the study of pin power variance as an objective. When pin power variance is used as an objective, as shown in Figure B.8a, the resulting pin power variance is on the low side of the correlation. From this data we can see that *PPF* correlates well with variance at the *NDF*, and the *NDF* of the variance study implies relatively low *PPF*.

Following the same template as the other studies in Chapter 5, it can be seen that the *MLP* surrogate model produced creates solutions with significantly larger EOC burnup at over 17 MWd/kgU.



(a) Correlation based on data from Experiment 2 (b) Correlation based on data from Experiment 3

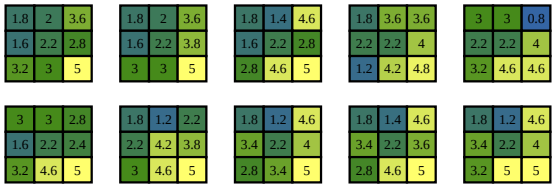
Fig. B.8 Correlation of *PPF* and pin power variance.



(a) DSO, $\sim$ 4000 cpu hrs, (hpc)

(b) MLP SMO, $<$ 0.0019 cpu hrs, (laptop)

Fig. B.9 Examples for *DSO* and *SMO* initial and final population results.



(a) Performant *LP* arrangements generated by a *CNN SMO* (seed=3454312)

(b) *DSO LP* arrangements generated for the bottom righthand quadrant of the microcore (seed=3454321)

Fig. B.10 Examples for *DSO* and *SMO LPs*, maximising burnup and minimising pin power variance

B.5 Required Neutron Population for Fission Matrix

In order to establish the correct number of neutrons to be used to generate fission matrices, the stochastic noise, ν_n of the fission matrix is estimated by using equation B.5. This assumes that the scattering is isotropic. The results for simulations carried out with a number of different population sizes are shown graphically in Figure B.12.

$$\nu_n n = |abs(F - F^T)| \tag{B.5}$$

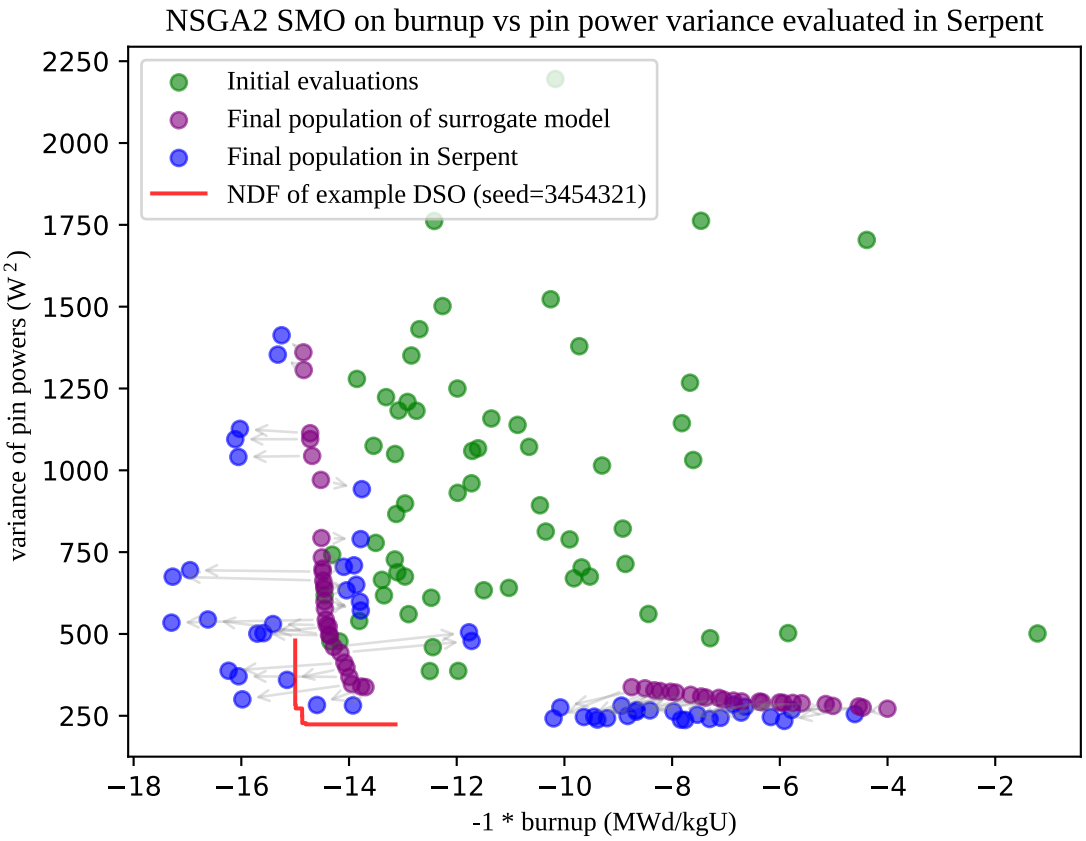


Fig. B.11 *NDF LP* for the *CNN SMO* simulated in serpent.

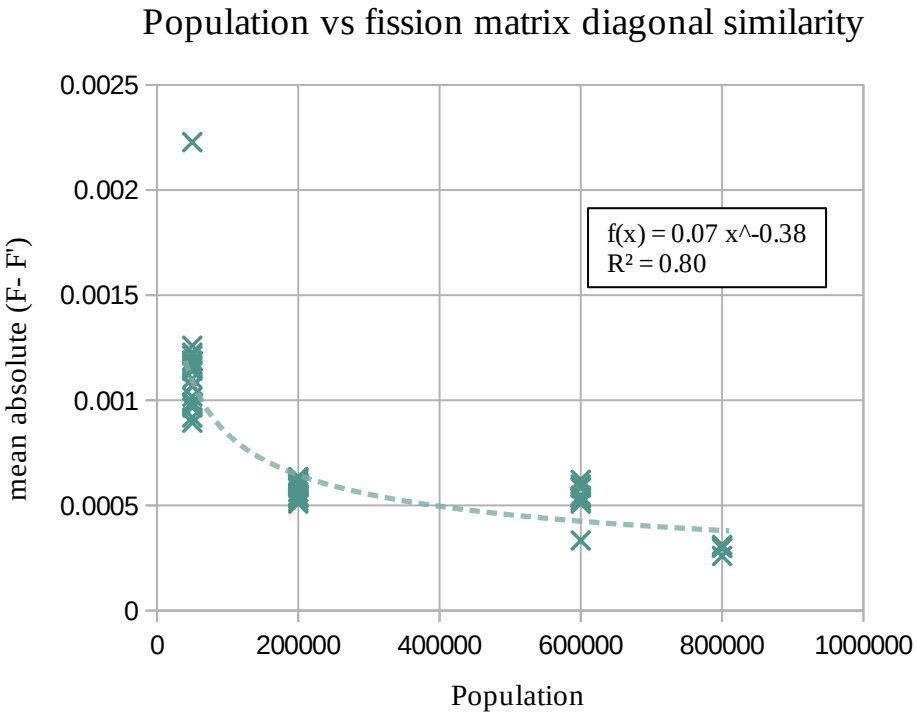


Fig. B.12 Mean difference between fission matrix and its transpose for a number of simulations

Appendix C

Application of the SMO Framework to FHR Fuel Technology

This work was produced in conjunction with Zhiyao Xing, Dr Eugene Schwageraus and Dr Geoffrey Parks and has been previously submitted to the M & C 2019 conference. [236]. It is reproduced here with some modifications to make the subject more tractable in the context of the thesis.

C.1 Introduction

Advanced Gas-cooled Reactors (AGRs) have a number of highly desirable design attributes, such as high coolant outlet temperature and online refuelling capabilities [164]. These attributes lead to high thermal efficiency and long continuous operation times. There is potential to create designs that exploit these features for Fluoride-salt-cooled High-temperature Reactors (FHRs) [69]Although FHRs differ from AGRs in that they are salt-cooled, they have attributes in common, being graphite-moderated, high-temperature reactors with a prismatic block fuel design. Figure C.1 (a) shows the essential elements of an AGR fuel element. A central tie bar is surrounded by 36 fuel pins, with a 0.4 mm steel cladding. Coolant flows around the pins and is contained within an inner graphite sleeve. It has been proposed by Forsberg *et al.* [68] that utilization of the knowledge gained from the design of AGR fuel assemblies might be advantageous in the design of FHRs. As well as being commercially proven, AGR fuel designs have the advantages of many years of operational experience, applied use of graphite at high temperature, and benefit from developed manufacture and spent fuel management techniques and infrastructure. Xing and Shwageraus [244] advanced this idea with an investigation of the design space for a number of parameters.

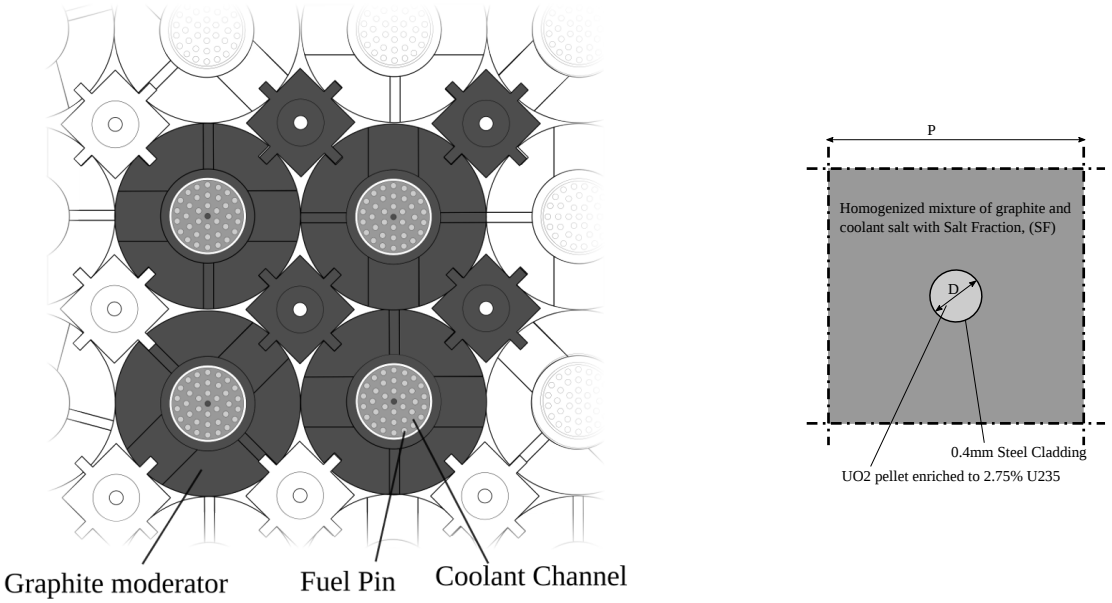


Fig. C.1 (a) AGR fuel elements are encased by graphite moderator blocks and keys (based on Nønbol [164]), and (b) An example solid pin model used in this paper.

In their investigation, as well as extended studies by Xing [243], [246], a significant amount of data was generated for fuel cell models. Single pin simulations were carried out within a homogen*i*zed moderator/coolant with reflected boundary conditions, as shown in Figure C.1 (b). Each of three types of pin (solid, annular and plates) were simulated for three different coolant salts (NaF-ZrF_4, FLiBe, FLiNaK) at two coolant temperatures (900 K, 1000 K), over a range of enrichments and salt-to-moderator volume ratios. Furthermore, for each type of pin, specific parameters were varied: in solid pins, the pitch and diameter were simulated for a range of values; in annular pins, inner and outer diameters and pitch were varied; in plate-type fuel, a variety of thicknesses and pitch sizes were explored.

The design choices in previous work [244] were made by manually selecting design samples from a uniform grid in the input design space. At each stage of fine tuning of these design parameters, more neutronic simulations were required. This gave rise to the incentive to develop a fast computational tool that can replace computationally expensive neutronic solvers and speed up the design optimization process. The FHR design variables showed combinatorial complexity, and previous simulations saw highly nonlinear neutronic performance in the output space. Simple interpolation schemes struggle to deliver accurate results across the searched space. Deep Multi Layer Perceptrons (MLPs), which have been shown to efficiently represent complex problems [217], are used in this study to develop a surrogate model of the FHR system. The model simulates neutronic parameters of interest (k_∞, Coolant Temperature Coefficient (CTC), Doppler Coefficient (DC), discharge burnup (BU)) of the unit cell FHR models from

input data such as geometric information (pin pitch, diameter), fuel type (uranium carbide (UC), Fully Ceramic Micro-encapsulated (FCM) fuel), and coolant salt. This allows the functionality of a fuel design to be evaluated in hundreds of microseconds, as opposed to minutes or hours, as expected in deterministic or Monte Carlo solutions of the neutron transport equation. The models developed in this study can be employed in future FHR preliminary design work to quickly narrow down the design spaces to optimized regions.

C.1.1 Simulation Experimental Set-up

In a previous study [243], eleven FHR families covering three fuel forms (solid, annular and plate-type fuel), two fuel materials (UC and FCM) and two salts (FLiBe and NaF-ZrF_4) were explored, out of which four were selected for further analysis. First, solid UC fuel with FLiBe coolant was identified as the best overall neutronic design. Second, the best FCM fuel model, the FLiBe-cooled solid pin was chosen, because of the robustness of the fuel form and associated safety benefits. Third, the best annular fuel design, the annular FCM FHR cooled with FLiBe was selected, because it showed promising thermal-hydraulic performance and allows higher power uprate [245]. Finally, the best performing NaF-ZrF_4 design, the solid UC pin with NaF-ZrF_4 coolant is selected as a backup option to the mainstream FLiBe-cooled FHRs, because it does not produce tritium. The neutronic data collected for these four design streams have been used in this study to train the surrogate models. The numbers of data samples corresponding to each design stream are summarised in Table C.1. A total sample population of 33,119 is used in the Beginning Of Life (BOL) experiments, and a total of 270 samples were used in the depletion experiments.

For the first part of this study, the BOL neutronic data of the eleven design families were used to train surrogate models. After proving the concept on these results, a subset of the four selected design families, which had been found to demonstrate the best neutronic performance, were brought forward for depletion analysis, the results of which were used to train new models incorporating depletion information.

In the neutronic data, three geometric inputs were varied: (1) lattice pitch to fuel rod diameter ratio (P/D), assuming an AGR fuel diameter [164]; (2) Salt mass Fraction (SF) in the salt and carbon homogeneous mixture in the simplified unit cell model (shown in Fig. C.1(b)); and (3) the enrichment of U235 in the fuel. P/D represents the geometrical envelope and heterogeneity of the model; it is varied from 1.2 to 8 in 0.4 increments to ensure both sufficient cooling of the fuel and reasonably high power densities. SF represents the measure of salt and carbon content in the core, and is varied from 20% to 100% with 20% increments. Fuel enrichment is varied from 2.5 wt% to 20 wt% with 2.5 wt% increments. Two temperature states (900 K and 1000 K) were covered for each design.

Table C.1 Simulation data summary totals.

Fuel, geometry	Coolant	Total samples	BU samples
UC, solid	$NaF\text{-}ZrF_4$	1260	45
	FLiBe	980	156
	FLiNaK	1260	
UC, annular	NaFZrF	2800	
	FLiBe	2799	
FCM, solid	$NaF\text{-}ZrF_4$	911	
	FLiBe	910	57
FCM, annular	$NaFZrF_4$	6075	
	FLiBe	6075	71
FCM, plate	$NaF\text{-}ZrF_4$	9743	
	FLiBe	306	

For the first part of this study, a new model was created and trained for each category shown in Table C.1. The data for the category was split by random selection into 80% training data and tested on the remaining 20% for its ability to accurately predict k_∞. The test set is not used in training, so it is considered an unbiased evaluation of the performance of the network. These models could be used by a system designer using the AGR framework for the design of FHR fuel elements, to guide the designer to find optimal arrangements of pins, pin sizes and to investigate design options without iteratively carrying out computationally expensive simulations.

C.2 Neuro-Surrogate Model Development

Design simplicity was favoured in order to allow the efficient development of a functional surrogate model. However, considerable optimization is possible on neural network models that is beyond the scope of this study.

C.2.1 Neural Network Architectural Investigation

Neural networks have previously been used in nuclear engineering to predict core parameters, e.g. for burnup predictions by Noda *et al.* [163] and in optimization of core loading patterns by Faria and Pereira [60] and Kim *et al.* [107]. An Artificial Neural Network, such as the one shown in Fig. C.2, is usually set up with low random initial weights between nodes so that differentials for the error values can be established. Then error is minimized by adjusting weights $(\mathbf{w}^1, \mathbf{w}^2, \mathbf{w}^3)$.

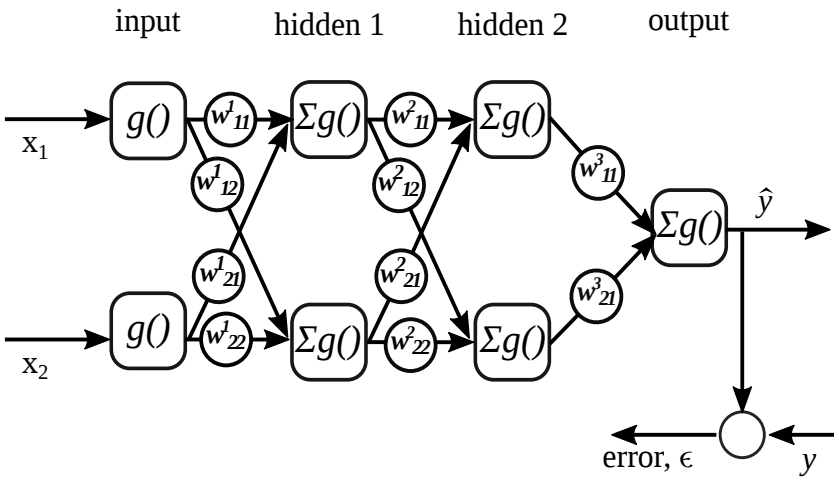


Fig. C.2 A simple feed-forward, MLP-type neural network.

Although myriad network topologies exist, relatively early work in neural networks showed that three-layer networks could act as universal function approximators [46], so work concentrated on shallow networks until the recent interest in deep learning began, summarised concisely by LeCun *et al.* [123]. Deep learning networks showed a marked improvement on shallow networks. Telgarsky [217] showed in 2015 that there are, in fact, functions that cannot be represented *efficiently* by shallow networks, and that a large set of functions are more easily represented in deep learning networks. A great deal of progress has been made in the last ten years, including novel topologies and efficient algorithm implementations [37, 6]. In 2016, Mhaskar *et al.* [151] described the kind of problems for which deep or shallow learning is advantageous. They show that, for complex problems, orders of magnitude of improvement in error performance is achieved with three or more hidden layers. Even on only moderately complex problems, performance is shown to be better with deep learning networks.

When training networks, a pseudo-random number generator, the Mersenne Twister [149], is used to generate the initial weights connecting each node (neuron) prior to training. All models are trained for 250 epochs, each with a batch of 35 data samples. The neuron *activation function* used is one that has gained popularity recently, called Rectified linear units (Relu) [79, 80], mathematically represented as:

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Relu simplifies the computational task for each neuron while still performing the function of stopping neuron weights from tending to infinity, which can happen if two weights cancel each

other out and the back-propagation algorithm modifies them in opposite directions. Compared to the traditional sigmoid function, Relu can often allow quicker learning of deep neural networks on large and complex sample spaces. The loss function uses root mean square (RMS) error to quantify the error of the network. Lastly, the 'Adam' optimizer [109] is used instead of, as is traditional, stochastic gradient descent to back-propagate the errors and update the weight of the neurons. Adam combines the advantages of two forms of gradient descent algorithm, namely the Adaptive Gradient Algorithm and Root Mean Square Propagation, to allow better performance on noisy data samples and large datasets with many input parameters [109].

Since a large corpus of data was already available for this study, it is relatively simple to use it for iteratively testing a variety of neural network topologies.

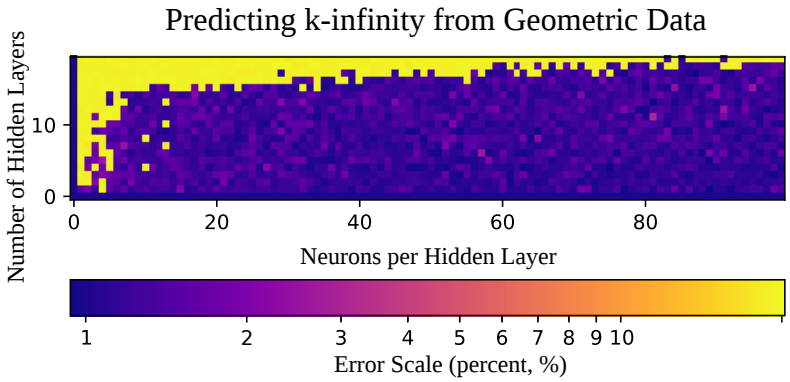


Fig. C.3 Error heatmap for different sized networks; this data guides the choice of network size used in Table C.2.

To establish a suitable network topology, tests were carried out on a range of networks. The performance of tested topologies is illustrated by the heatmap shown in Fig. C.3, with RMS error characterised by the colour scheme transitioning from yellow to dark blue, with the darker shades representing better accuracy. Each pixel represents the prediction error of a dense, rectangular, feed-forward network, with an input layer of 50 neurons and between 1 and 20 hidden layers and from 1 to 100 neurons per hidden layer in the network. The problem is 'embarrassingly parallelizable' [83, p 14] and can be run relatively quickly without requiring substantial computational resource. From the heatmap, it can be seen that with a small number of neurons per hidden layer and/or a larger number of ($\sim$20) hidden layers, the model performs poorly. With a small number of total nodes, the level of complexity of the problem cannot be effectively represented by the network. If the network has too many layers, the back-propagation of errors across many layers of weights becomes impossible.

The network is, at this point, trained using BOL data only. How the physics changes as a result of depletion has not yet been examined; without this, prediction of the reactor's economic

and safety performance would be incomplete and unconvincing. The discharge burnup of the reactor, which represents the revenue from reactor operation, has not been incorporated into the database. Also, how reactivity feedbacks change with depletion is important for the multi-batch/on-load refuelling management scheme. To incorporate burnup characteristics into the surrogate model, depletion calculations are carried out for well-performing samples from the four selected design families. Exercising the network architectural investigation using the depletion data, the heatmap for the prediction error shown in Fig. C.4 is produced. In this experiment, dense, feed-forward networks, with an input layer of 150 neurons and between 1 and 20 hidden layers and from 1 to 100 neurons per hidden layer in the network, are trained and tested. In this case, it appears that there are no longer sufficient samples to fully train the networks up to the limits of the back-propagation algorithm. Shallower networks have superior results, implying that there are not enough samples to propagate errors back through deeper networks.

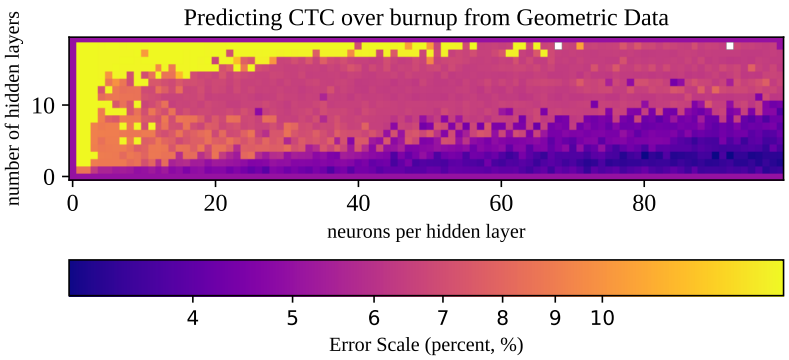


Fig. C.4 Error heatmap for depletion study of solid pin designs

C.2.2 Evaluation of Accuracy of Deep Learning Networks

In order to establish the accuracy of the deep learning networks at predicting k_∞ from the experimental input data, scripts were written that automatically tested the training of the networks over a range of random number generator seed values. Adjacent random seeds are used to study the variation of the results. The RMS error and standard deviation, σ, of the model on the test set were recorded. For stochastic algorithms, it is standard to test over 30 (or more) seed values. As before, 80% of the data was used for training and 20% for testing. This was repeated for each random seed – meaning that a new random subset of the hypercube data was used for each test set.

C.3 Results

C.3.1 Beginning of Life k_∞ Prediction

This study shows that a deep MLP will accurately predict k_∞ for a variety of proposed fuel pin designs. Table C.2 presents a summary of the results achieved using a sequential architecture deep learning network to predict k_∞ for a particular set of design parameters. Based on the results in Fig. C.3, a network size of 10 layers of 50 neurons was selected for further experiments. This can be seen to be reasonably far from the high error regions for this problem. Le Cun *et al.* [123] suggest 5 to 20 hidden layers define a deep network, but it is now common to define networks with more than one hidden layer as deep, e.g. [53].

Table C.2 Prediction accuracy (averaged over 30 runs) of sequential networks trained and tested on fuel pin design data.

Fuel, geometry	coolant	RMS k_∞ error / pcm	k_∞ σ	RMS k_∞ error / pcm	k_∞ σ
UC, solid	FLiBe	655.31	223.73	482.08	187.05
	$NaF\text{-}ZrF_4$	823.74	298.08		
	FLiNaK	749.49	249.94		
UC, annular	$NaF\text{-}ZrF_4$	701.97	262.46	559.26	214.49
	FLiBe	711.77	344.43		
FCM, solid	$NaF\text{-}ZrF_4$	552.81	181.71	534.84	180.84
	FLiBe	725.81	187.07		
FCM, annular	$NaF\text{-}ZrF_4$	617.60	221.79	319.95	125.17
	FLiBe	887.85	354.39		
FCM, plate	$NaF\text{-}ZrF_4$	371.96	135.92	381.92	155.16
	FLiBe	624.63	208.72		

Columns 3 and 4 of Table C.2 show a summary of the results achieved when a new network was instantiated for each type of fuel and salt. Columns 5 and 6 show the results for networks that used the salt as an input and were trained on a single set of data created by merging data for the same fuel type and different salts. Somewhat surprisingly these networks out-perform the more specialised networks. This is thought to be due to the extra learning opportunity gained by merging these datasets and allowing these networks to train on more samples.

Experiments show that these pre-trained feed-forward sequential networks evaluate input data to deliver an estimate of k_∞ in 320 microseconds on standard PC hardware. The speed and accuracy of these results make the approach significant for iterative optimization strategies and for initial exploration of the design space. For example, the test set of 270 simulations required $\sim$1.1 cpu hours per burnup step, totalling 2970 cpu hours, whereas the neural networks used in

this section are able to approximate the results in < 1 second on a single core of a commercial laptop PC.

C.3.2 Prediction of Discharge Burnup and Reactivity Coefficients Over Depletion

After demonstrating that the surrogate models can predict BOL neutronic performance of FHRs with reasonable accuracy, depletion analysis is performed for the selected design space up to a burnup of 80 MWd/kgHM. To incorporate fuel cycle economic considerations into the analysis, the discharge burnup to enrichment ratio (BU/e) is adopted as an output of the surrogate model to replace BOL k_∞ as an indicator. The ratio represents the amount of economic gain from the reactor's electricity production (proportional to burnup) per unit of fuel cost (approximately proportional to fuel enrichment) and should therefore be maximized. The discharge burnup of the reactor is calculated as twice that of a once-through fuel cycle, assuming the use of an on-load refuelling scheme for the FHR. The assembly (represented by unit cell models) CTC and DC are obtained at various stages during the depletion. Core-average reactivity coefficients are calculated assuming a three-batch refuelling scheme, to give an approximation of on-load refuelled core-average reactivity coefficients, in case they become less negative or even positive with burn up. Both reactivity coefficients, which are also treated as outputs of the surrogate models, should be kept below zero throughout the cycle to assure stability and favourable behaviour in transients. Using the depletion data, the surrogate models were able to generate performance levels as summarised in Table C.3.

Table C.3 Prediction percentage error (%) to 3 significant figures of sequential networks trained and tested on depletion data for fuel pins.

Fuel, geometry	coolant	BU/e	CTC	DC	BU/e	CTC	DC
UC, solid	FLiBe	7.51	7.38	8.54	5.53	5.43	5.50
	$NaF\text{-}ZrF_4$	17.8	17.5	17.6			
UC, annular	FLiBe	12.5	16.1	17.7			
FCM, solid	FLiBe	7.52	7.38	8.55			

Greater errors are evident in this exercise compared with the previous models trained on BOL data. This is partly because significantly less depletion data was available to the networks during training and also because the uncertainty on the training data is higher in reactivity coefficients. It also explains the greater errors reported in Table C.3 for the $NaF\text{-}ZrF_4$ and the annular fuel design families, which had much less data compared with the other cases, as shown

in Table C.1. Again, by merging two UC solid pin neutronic datasets, the network produced more accurate predictions. Absolute RMS error from the surrogate model is, for example, 0.131 $pcmK^{-1}$ for core average CTC in UC solid pins, whereas statistical uncertainties from the Monte Carlo Serpent calculations were found to be around 0.1 $pcmK^{-1}$ for CTC.

The surrogate models' errors range from 5.4% to 17.8% in BU/e, giving absolute errors of the order of 1 MWd/kgHM. The original For the preliminary design purpose as a fuel cycle economics indicator, this is considered reasonably acceptable. The two reactivity coefficients must be kept negative and can be used as either optimization objectives or constraints. The errors from the surrogate models require an additional margin to be put on the two parameters if used as optimization constraints.

For the entire depletion design space covered in this study, BU/e values ranging from 0.07 to 27.13 MWd/kgHM were obtained, with an average value of 8.6 MWd/kgHM. Imposing a constraint of -0.2 pcm/K on both reactivity coefficients and an upper bound of 200 MWd/kgHM on reactor discharge burnup, the current data space produced a best design that can reach a BU/e of 15.4 MWd/kgHM. Compared to in-service LWRs, which have BU/e of $\sim$10 MWd/kgHM, the AGR-like FHR shows great potential in terms of reactor economics. For future study, the surrogate models can be coupled to optimization algorithms to identify optimized design spaces. More depletion data can be accumulated to specifically target these design spaces, which can, in turn, be fed to the network to further improve its predictive power.

C.4 Conclusions

A neural network regression analysis has been applied to a collected body of FHR design data, showing that it is possible to interpolate the data and effectively predict neutronic performance parameters despite a highly nonlinear system response. Initial work shows that it has been possible to reach an expected prediction accuracy for k_∞ of $\sim$675 pcm and that a more generalised model incorporating more results within a single model performs better, achieving an expected accuracy for k_∞ of $\sim$456 pcm. The network thereafter demonstrated acceptable performance when trained on a much smaller sample space that incorporated depletion data. These results rely on the availability of a training set; therefore, this method is significant when iterative simulations are expected anyway. Although a neuro-surrogate model is unable to out-perform Monte Carlo or deterministic transport codes in terms of the accuracy, the sub-millisecond calculation speed of a pre-trained network compared to the relative expense of iteratively running neutronics simulations justifies the investment of seconds of cpu time to train an advanced regression method like a deep learning network, especially for initial studies, before a neutronics code is employed, and for design and optimization purposes, where

the fast evaluation of approximate results is desirable. Further testing will show whether meta-parameters can be used to train a network, allowing the creation of a system that could be used to predict the performance of geometries that have not been explicitly simulated.

Code for the surrogate models is available under the permissive MIT license and can be cloned from the repository: *https://ajw287@bitbucket.org/ajw287/fhr-surrogate-mandc.git*

Appendix D

Technical details

Good results for *SMO* have been achieved in this thesis on entry level hardware, but in order to verify this *DSO* simulations were required to be carried out on a number of more advanced research machines.

The results have been carefully tested to ensure that what is presented is not strongly hardware dependent. In fact, results have shown remarkable robustness across diverse systems, such as the *laptop* hardware used for *SMO*, HPCs and the 'Lise' blade server used for neutronics and the *phd* server used for scheduling and managing optimisation tasks while submitting jobs to the HPC or Lise.

This section includes details relevant but incidental to this thesis, although the hardware is expressly not the focus of the thesis, it is important that the hardware be recorded in order to ensure reproducibility. Since systems of multiple machines were applied during the research, it is advantageous to keep track of software version numbers since it was not always possible to install the same version across all of the systems being used.

Table D.1 shows abridged output of the command `lscpu` when run on each of the machines used, while Table D.2 shows the output of the software version numbers and data libraries used for simulations and optimisation.

Typically the entry level *laptop* hardware was used for surrogate model optimisation, while *DSO* optimisation and the scheduling of training set generation was run on the *phd* server, which was able to be continually connected to the internet and was therefore able to submit neutronics simulation jobs onto the HPC and Lise blade server.

D.1 Technical Information on Computer Equipment:

Table D.1 The Cambridge High Performance Computer (HPC), a departmental blade server (Lise), a laptop (laptop) and a desktop PC (phd) were used in this thesis. The `lscpu` results for each are summarised below.

	HPC (per node)	**'Lise' blade PC**	**laptop**	**'phd' server**
Architecture:	x86_64	x86_64	x86_64	x86_64
CPU op-mode(s):	32-bit, 64-bit	32-bit, 64-bit	32-bit, 64-bit	32-bit, 64-bit
Byte Order:	Little Endian	Little Endian	Little Endian	Little Endian
CPU(s):	32	32	8	8
Vendor ID:	GenuineIntel	GenuineIntel	GenuineIntel	GenuineIntel
CPU family:	4	4	6	6
Model:	85	85	142	26
Model name:	Intel® Xeon® Gold 6142 CPU @ 2.60GHz	Intel® Xeon® Gold 6130 CPU @ 2.10GHz	Intel® Core™ i5-8250U CPU @ 1.60GHz	Intel® Core™ i7 CPU 950 @ 3.07GHz
BogoMIPS	5200	4200	3600	6134

Table D.2 Software used, version numbers and compiler where possible

	HPC	**'Lise' (blade PC)**	**laptop**	**'phd' server**
WIMS Version	10.a	10.a	–	–
cross-sections	'w10j22v3.dat'	'w10j22v3.dat'		
Serpent 2	2.1.30 (February	2.1.30 (February	–	–
(sss2 -version)	14, 2018)	14, 2018)		
cross-sections	[57]	[57]		
Python Version	3.6.7 |Anaconda,	3.6.9 (default,	3.7.7 (default,	3.7.5 (default,
('sys.version')	Inc.| (default,	Nov 7 2019,	Mar 10 2020,	Nov 2 2019,
	Oct 23 2018,	10:44:02) [GCC	13:18:53) [GCC	22:54:58) [GCC
	19:16:44) [GCC	8.3.0]	9.2.1 20200306]	6.3.0 20170516]
	7.3.0]			
TensorFlow Version	–	–	1.13.1	–
(tf.__version__)				
Pygmo Version	–	–	2.11	2.11
(pg.__version__)				