12-2022

Multi-objective Optimization of the Fast Neutron Source by Machine Learning

John L. Pevey
University of Tennessee, Knoxville, jpevey@vols.utk.edu

Recommended Citation

Pevey, John L., "Multi-objective Optimization of the Fast Neutron Source by Machine Learning. " PhD diss., University of Tennessee, 2022.
https://trace.tennessee.edu/utk_graddiss/7734

To the Graduate Council:

I am submitting herewith a dissertation written by John L. Pevey entitled "Multi-objective Optimization of the Fast Neutron Source by Machine Learning." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

John W. Hines, Major Professor

We have read this dissertation and recommend its acceptance:

Vlad Sobes, Ondrej Chvala, Germina Ilas

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Multi-objective Optimization of the Fast Neutron Source by Machine Learning

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

John L. Pevey

December 2022

To my wife Abigail, she is my biggest fan and without her constant love and support I would not have been able to do this.

To my parents, Ron and Nancy and my family, who love and support me always.

To my dog Dudley, you were the best study partner I could ask for.

Acknowledgments

I would like to acknowledge my co-chairs, Dr. Hines and Dr. Sobes and my committee members, Dr. Ilas and Dr. Chvala. Without their guidance and insights this work would not have been possible.

Abstract

The design and optimization of nuclear systems can be a difficult task, often with prohibitively large design spaces, as well as both competing and complex objectives and constraints. When faced with such an optimization, the task of designing an algorithm for this optimization falls to engineers who must apply engineering knowledge and experience to reduce the scope of the optimization to a manageable size. When sufficient computational resources are available, unsupervised optimization can be used.

The optimization of the Fast Neutron Source (FNS) at the University of Tennessee is presented as an example for the methodologies developed in this work. The FNS will be a platform for subcritical nuclear experiments that will reduce specific nuclear data uncertainties of next-generation reactor designs. It features a coupled fast-thermal design with interchangeable components around an experimental volume where a neutron spectrum, derived from a next-generation reactor design, will be produced.

Two complete genetic algorithm optimizations of an FNS experiment targeting a sodium fast reactor neutron spectrum are presented. The first optimization is a standard implementation of a genetic algorithm. The second utilizes neural network based surrogate models to produce better FNS designs. In this second optimization, the surrogate models are trained during the execution of the algorithm and gradually learn to replace the expensive objective functions. The second optimization outperformed by increasing the total neutron flux 24%, increased the maximum similarity of the neutron flux spectrum, as measured by representativity, from 0.978 to 0.995 and producing configurations which were more sensitive to material insertions by +124 pcm and -217 pcm. In addition to the genetic algorithm optimizations, a second optimization methodology using directly calculated derivatives is presented.

The methods explored in this work show how complex nuclear systems can be optimized using both gradient informed and uninformed methods. These methods are augmented using both neural network surrogate models and directly calculated derivatives, which allow for better optimization outcomes. These methods are applied to the optimization of several variations of FNS experiments and are shown to produce a more robust suite of potential designs given similar computational resources.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The quantification and accurate prediction of the uncertainty in reactivity due to nuclear data uncertainty is necessary for the accurate design of next generation reactors. This uncertainty, if not well characterized, can lead to increased and potentially unnecessarily conservatism estimations of bias and uncertainty on the neutronic characteristics of a system versus what they would be with better understood nuclear data. Generally, nuclear data uncertainty can be reduced with high-quality experimental data. The uncertainty in the nuclear characteristics driven by nuclear data uncertainty can be reduced using tools such as the TSUNAMI code in SCALE [3]. the uncertainty in predicted values due to nuclear data on a given next-generation system can be quantified and compared to known critical benchmarks.

Uncertainty in a well-modeled nuclear system due to nuclear data uncertainty can be reduced by applying knowledge gained through nuclear experiments on systems which are similarly sensitive to the same nuclear data uncertainties. The Fast Neutron Source (FNS) at the University of Tennessee will be a highly re-configurable sub-critical nuclear experiment facility where these types of experiments will be preformed. Experiments will be designed to target specific nuclear data uncertainties by producing a configuration of the FNS which stresses the same nuclear data uncertainty as a target design. The system will be driven by a deuterium-based neutron generator which produces 4 x 10^9 neutrons per second at 2.45 MeV

[4]. The FNS features a coupled fast-thermal design with a relatively low k-eff fast region and a relatively high k-eff thermal region. When coupled the system has a k-eff below 0.95. The use of a coupled system more efficiently utilizes the fuel versus a purely fast system. The FNS is designed to have a neutron multiplication less than 0.95, which will ensure both sufficient margin for criticality safety.

Potential experiments in the FNS include both integral experiments based on measuring the affect on k-eff of the insertion and removal of target material into the system and activation experiments where a neutron flux spectrum is produced and a material is irradiated in it, removed, and then measured for activation. In these experiments it is important to produce the specific neutron flux spectrum of interest, maximize that flux (within safety constraints) to minimize experiment time and, in the case of the integral k-eff experiment, maximize the change in k-eff of the experiment.

The design of FNS experiments which maximize these objectives while being below the constraint is a difficult task. Due to the fast/thermal coupled nature of the FNS, targeting a fast spectrum with little to no thermal component causes the neutron spectrum target and magnitude to be in opposition. Therefore, a multi-objective optimization must be implemented which balances these objectives in order to produce potential FNS designs. The following work explores this sort of optimization using genetic and gradient descent algorithms.

1.1 Problem Statement

There are many methods to optimize multi-objective systems such as the design of FNS experiments. The most computationally expensive method would be to evaluate every possible configuration of the FNS. The design space, or the number of possible solutions to the optimization problem of the FNS, is at least 3.7 x 10^{71} possible solutions. This is with assumptions made which reduce the design space, such as limiting the number of materials in the FNS to three and enforcing symmetrical material loading. A genetic algorithm was implemented for the optimization of the FNS. Genetic algorithms are an optimization method which implements the ideas of natural selection and evolution to evolve a set of solutions with respect to the objectives. Genetic algorithms are powerful but are not guaranteed to produce the "best" solution possible. For the FNS optimization the genetic algorithm is accelerated by building a surrogate model which evaluates potential designs and increases the effectiveness of the optimization.

1.2 Original Contributions

The FNS will be a platform for producing targeted nuclear data experiments for next-generation reactor designs that may not have sufficient coverage using currently available experiments and benchmarks. Due to both the inherent complexity of the FNS and the multiple objectives necessary to produce an experiment that is worthwhile, the task of optimizing such a system is a challenging one. It is proposed that a genetic algorithm would be an effective method of optimizing such a system. Furthermore, this figure of merit of the genetic algorithm could be increased by the use of surrogate modeling in which the data produced by the genetic algorithm is used to produce predictive functions which inform and accelerate the optimization. The use of directly calculated derivatives could also be used to further accelerate the optimization.

Primary Objectives:

1. Develop a genetic algorithm for the multi-objective optimization of nuclear experiments.

2. Develop methodologies for the use of feature-extracting neural networks to be used as a surrogate models for neutronic calculations.

3. Develop methodologies for the use of gradient descent with directly calculated sensitivities (dk-eff/k-eff/dΣ/dΣ) for nuclear systems.

4. Implement these methods (#1, #2, #3) into a Fast Neutron Source optimization (#1) as a proof of concept for their applicability to nuclear design.

1.3 Organization of Document

The chapters of this document include an introduction, which includes the problem statement, the original contributions of this work and the organization of the document. Chapter two includes both background information and a literature review section where eleven examples of contemporary work are compared to the presented methods. Chapter three presents the methodology for the genetic algorithm optimization of the FNS. Chapter four presents several examples of optimizations of the FNS including the optimization of a sodium-cooled fast reactor spectra and other optimizations. In chapter five the TSUNAMI-based optimizations include the optimization of a 2-D reactor system and a comparison of TSUNAMI-based sensitivities to MCNP-calculated values. Chapter six concludes with a summary of how this work meets the proposal objectives, a general conclusion and some avenues for future work. Other sections include a bibliography, a vita, and a full MCNP and SCALE input example in the Appendix.

Chapter 2

Background and Literature Review

2.1 Background

2.1.1 The Fast Neutron Source

The Fast Neutron Source (FNS) will be a platform for performing nuclear experiments useful to the designers of next-generation reactors. It will be located in the new Nuclear Engineering building on the campus of the University of Tennessee. It will feature a coupled fast-thermal design and be driven by a DD neutron generator. Several experiments could potentially be done with the FNS including beam line measurements, insertion experiment k-eff measurements and spectra matching experiments. The FNS will also feature a highly re-configurable design which will be constructed to produce nuclear experiments that reproduce a targeted neutron spectra that will be used to reduce nuclear data uncertainties in next-generation reactors. Figure 2.1 presents a plan view of the FNS vault in the new building.

The re-configurable nature of the FNS is due to it being constructed of 6" x 6" x 10" inner dimension cassettes, with up to 20 0.5" plates of either fuel (enriched or natural uranium), polyethylene, or a target material chosen based on the requirements of the experiment. The target material is one that is selected to moderate the neutron spectra in the experimental volume to be similar to a target neutron spectra. These cassettes are then stacked in to

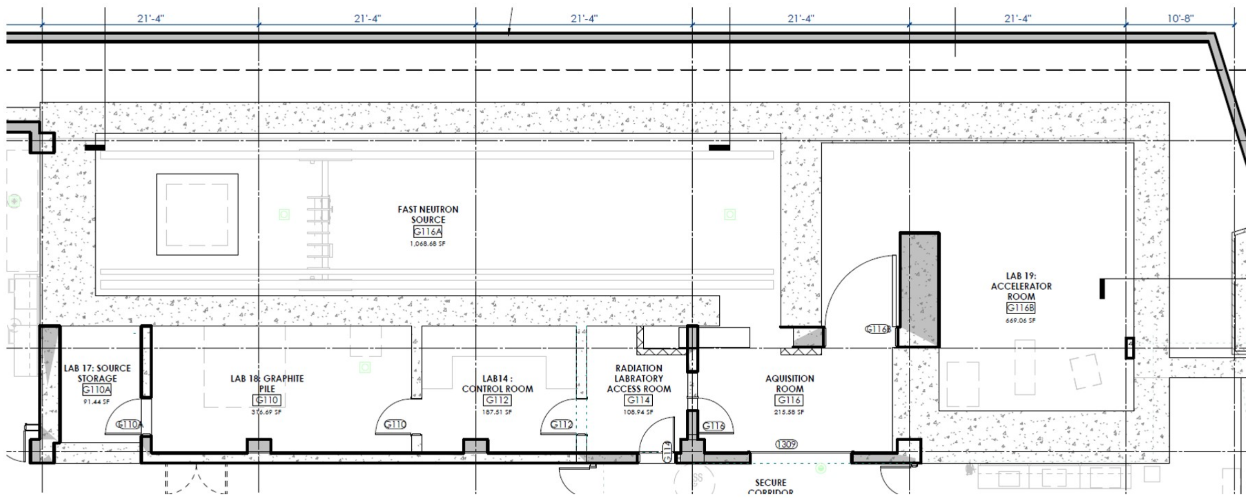


Figure 2.1: Plan View of FNS Laboratory Space

3 zones of a 5 x 5 pattern of cassettes. The neutron source is located in zone 3 and the experimental volume is located in zone 1. In some optimizations, the experimental volume is not fixed in zone 3 and is able to move into zone 2 by the adjustment of cassette pattern A in both zones 1 and 2. Figure 3.6 in Chapter 5 shows a cut through view of the MCNP model of the FNS.

The goal of the optimization of a FNS experiment is to maximize the total neutron flux in the experimental volume, maximize the similarity of that neutron flux to a target neutron flux, and, for integral experiments, to maximize the differential k-eff worth of inserting a target material into the experiment volume. These objectives directly relate to the goal of producing sub-critical experiments which exercises the same nuclear data uncertainties of a target reactor design. A constraint on k-eff of 0.95 is also enforced to ensure subcriticality during operation.

2.1.2 Uncertainty in Nuclear Systems due to Nuclear Data

The quantification of the uncertainty in cross sections is an important task in all nuclear systems. All nuclear modeling is reliant on nuclear data and therefore the uncertainty on nuclear data can be propagated to all figures of merit for nuclear systems. This includes uncertainties on k-eff in criticality safety analysis, reaction rates in shielding and detector analysis and various reactor-related values such as void coefficients, reactivity worth and other nuclear characteristics of interest to neutronics analysts.

Nuclear cross section uncertainty data is generally referred to as covariance data. The cross section uncertainty data is stored as a matrix of covariance values which is all that is needed to represent the data due to the imposition of a normal distribution on the data. This data is included in many nuclear data evaluations such as the Evaluated Nuclear Data File (ENDF) [5], the Joint Evaluated Fission and Fusion (JEFF) [6] and Japanese Evaluated Nuclear Data Library (JENDL) [7] nuclear data libraries. Figure 2.2 shows an example of a covariance matrix. On this matrix the diagonal component is the variance of the group and the off diagonal terms are the co-variance between energy groups. Uncertainty data in

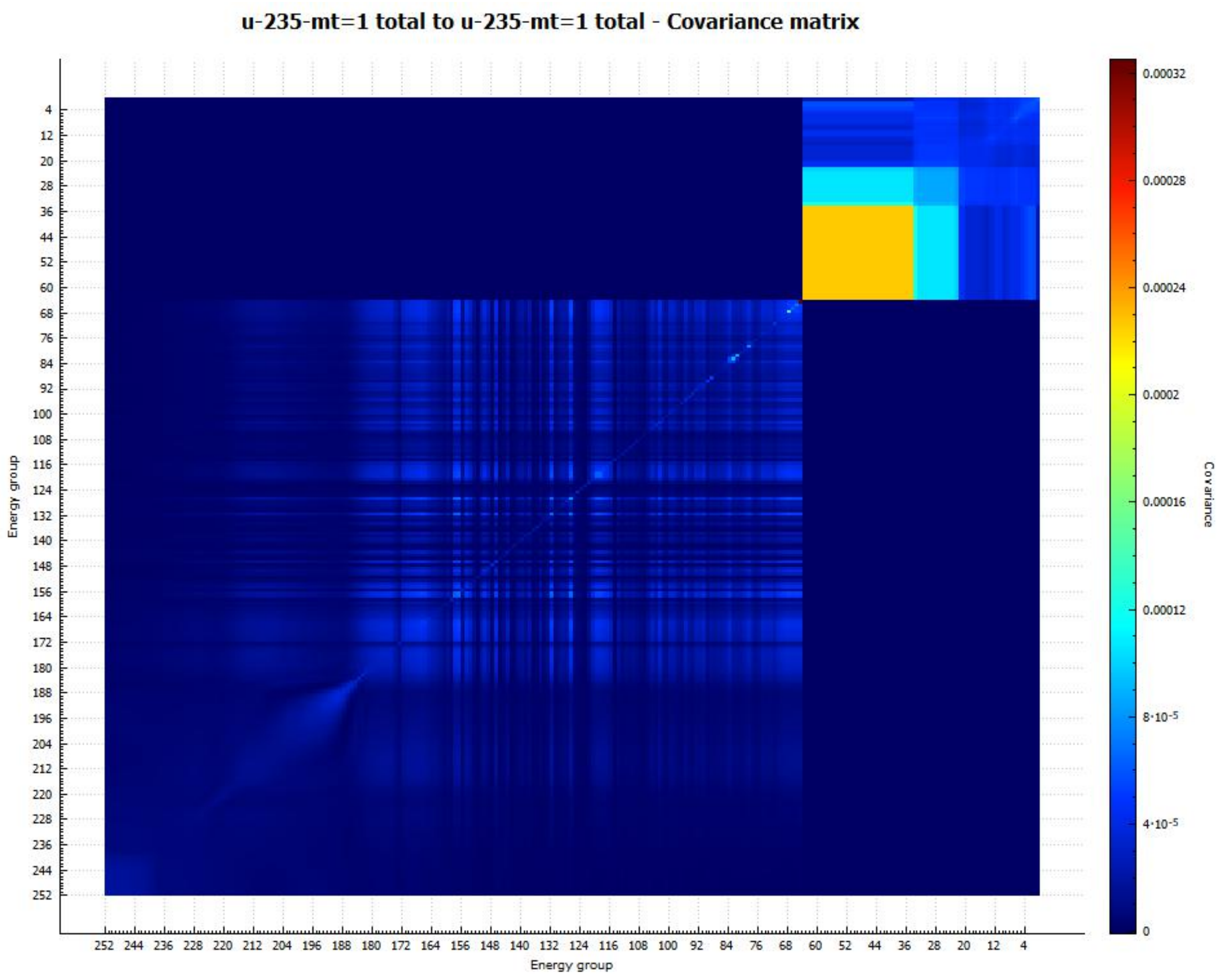


Figure 2.2: Covariance Matrix of U-235, Total. ENDF/VII.1 (SCALE)

the form of covariance matrices have also been calculated between isotope-reaction pairs. These uncertainty data show the correlated variation in nuclear data libraries that could be due to many effects such as materials, methods and tools used in the experiments which the libraries are created from. Figure 2.3 shows the group-wise uncertainty on the total cross section of U-235 in found in ENDF/VII.1 (SCALE). The maximum uncertainty on the total cross section is over 1.75% and directly corresponds to uncertainty in the predicted k-eff of any nuclear system in which neutrons at these energies interact with U-235.

Both MCNP and SCALE code packages include tools for uncertainty and sensitivity analysis. These include directly calculating the uncertainties by perturbing cross section data (SAMPLR) in both multi-group and continuous energy and indirectly calculating them by either calculating the adjoint or approximating that adjoint flux solution by either the Iterated Fission Probability method or the CLUTCH method. In this work the CLUTCH method is used. The usage of CLUTCH, as well as the background on the methodology, can be found in the Methodology section.

Uncertainties in nuclear data drive uncertainties in the figures of merit related to nuclear systems such as, most broadly, uncertainties in neutron flux and uncertainties in the neutron multiplication of the system. A large amount of nuclear data uncertainty leads to large uncertainties in the calculation of neutron flux which has a cascading effect on all nuclear figures of merit. For example, a large uncertainty in the neutron multiplication of a system directly leads to uncertainty in reactivity control which then leads to uncertainty in depletion behavior of a system. This extra uncertainty can be accounted for by increasing the reactivity margin of the system, but doing that may have detrimental effects on other characteristics of the system. Excess reactivity may need to be designed into a system to ensure that the reactor stays super critical throughout its lifetime. In commercial reactors where fuel is shuffled and reloaded at regular intervals this may not have much of an effect. However, in cases where reloading is not feasible such as in some proposed micro-reactor solutions, the effect on lifetime can be a detriment to the proposed reactor design.

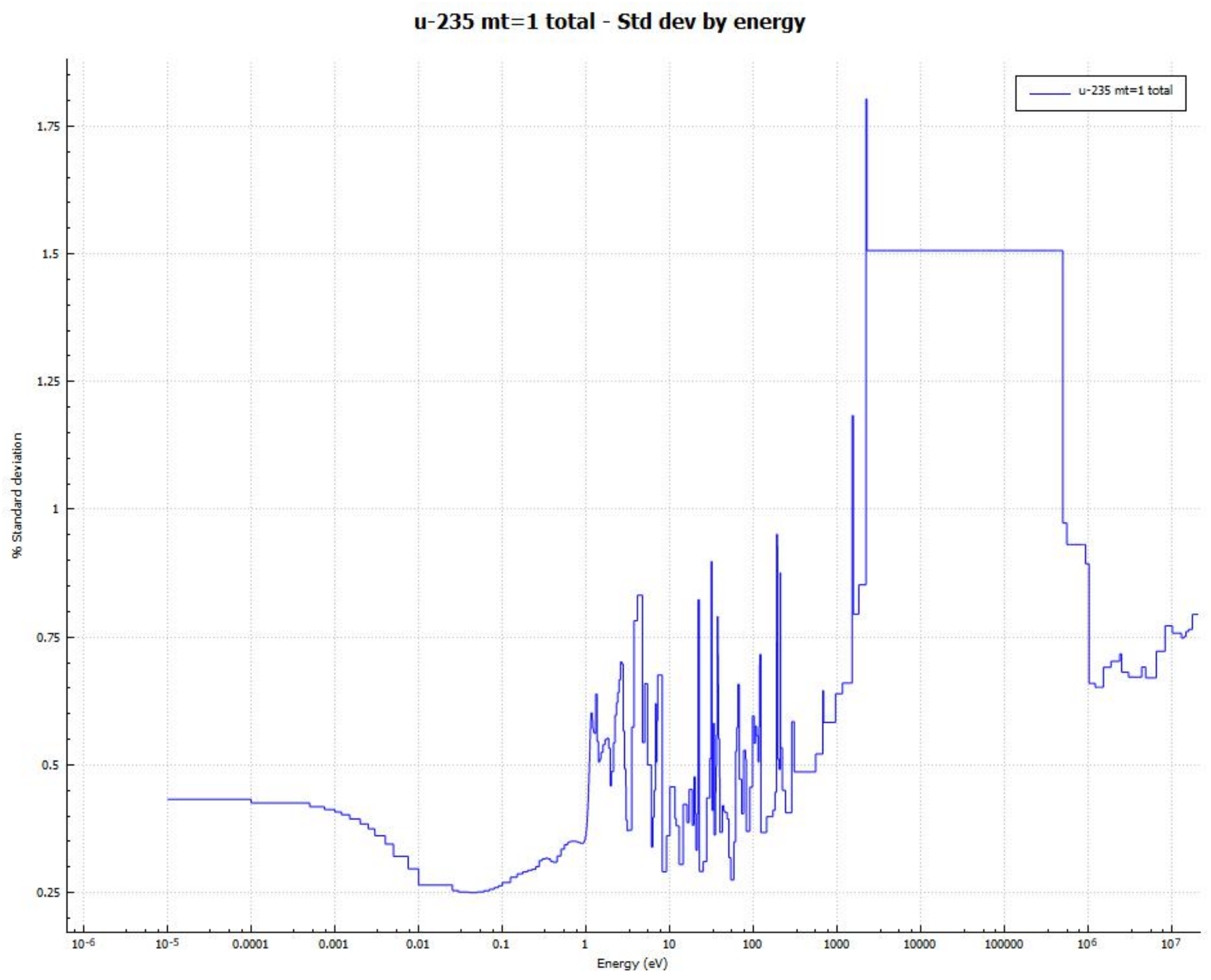


Figure 2.3: Group-wise Uncertainty in the Total Cross Section of U-235. ENDF/VII.1 (SCALE)

2.1.3 Genetic Algorithm Overview

Genetic algorithms are a framework of heuristics and methods used for the optimization of systems first proposed by John Holland [8]. Generally, they borrow ideas from the natural world and apply them to optimization problems which have an intractably large design space. Genetic algorithms can succeed where deterministic optimization methodologies such as gradient descent or Newton's Method, for example, can become stuck in local, sub-optimal solutions. Genetic algorithms avoid this trap by applying basic ideas from biology including survival of the fittest (selection and crossover) and stochastic changes (mutations). Genetic algorithms evolve a set of individuals towards one or more objectives. Genetic algorithms are not guaranteed to find the optimal solution and can fail in certain problems, such as when those without a gradient in the design space (such as seen in cryptography), but have been widely applied to many optimization problems including in the nuclear field [9].

There are many heuristics associated with genetic algorithms which govern how the algorithm balances the exploration and exploitation of the design space. The most important of these are the crossover rate and the mutation rate which control to what degree the algorithm will explore new areas of the design space (crossover) or exploit areas of the design space already explored (mutation). The crossover rate is a value between 0 and 1 which represents the probability that a newly created child will be made from the combination of two parents versus from the mutation of a single parent. Note that even when a child is created by crossover, it may still undergo a stochastic mutation. The mutation rate is the rate at which individuals that are created by crossover are mutated and to what degree they are mutated.

Another important aspect of designing a successful genetic algorithm is building an objective function that provides useful feedback to the algorithm and which results in optimizing the population towards the goal of the designer. In the simplest case of a single objective this is straightforward, where individuals can be strictly ranked by their objective function evaluation. If the design space is similarly simple, such as if there is only a single input variable, the algorithm is essentially a stochastic gradient descent with the ability to break out of a local minima due to the crossover.

With multiple objectives the selection and evaluation of the objective function can become more important. The balancing of one objective versus another can be crucial to the successful application of a genetic algorithm for optimization. The simplest way to account for multiple objectives is to combine the objectives into a single value by a user-defined function. This can be simple addition, multiplication or other function. Then, this single value is then optimized. A user-defined weighting function can also be applied to the objectives in order to balance objectives that may be orders of magnitude different or to push the algorithm to prefer one over another. These methods can work in cases where the trade-off between the multiple objectives are well understood by the designer, but may result in an optimization that favors one objective over another. The choice objective functions can greatly influence how effective a genetic algorithm is for the purpose of the optimization. One method of dealing with multiple objectives is to run a genetic algorithm optimization multiple times, each with a different set of objective function weightings. In this way, the designer produces not just a single ”best” individual but instead a suite of individuals that balance the various objectives called the Pareto front. Other optimization algorithms do not implement weighted objective functions but instead implement functions and heuristics in the algorithm with the goal of producing and optimizing the Pareto front. The Non-Dominated Sorting Algorithm-II [1] used in this work is one of these types. This algorithm is discussed in more detail in the following section.

2.1.4 Artificial Neural Networks

Artificial neural neural networks are a type of machine learning which use simplified notions of biological neurons in the brain to build a general function which can be ”trained” to approximate any function of interest. The first artificial neural networks, called the perceptron was invented in 1958 by Frank Rosenblatt [10]. The perceptron comprises functions which take as an input a vector of values which are combined with a vector of weights and a singular bias value. These values are then summed and used as input into a non-linear function. Initially the heavy-side function was used. In this function the output would either be a 1 or a 0 depending on the summed value. This type of function in similar to what was observed in nature, but it was quickly observed that other functions which are easily differentiable are more effective at building trainable perceptrons. Examples of these include the hyperbolic tangent function, the S-shaped sigmoid function:

$$f(x) = \frac{1}{1 - e^{-x}}$$

and the rectified linear unit (ReLU) [11] function. The ReLU is a function that returns a 0 if the input is below a threshold and returns a linearly increasing value if the input is above that threshold. The slope above the threshold is set by the designer. Some times using this function can result in neurons ”vanishing”, which is when the neuron becomes stuck returning 0 during training. This is because below that threshold there is no gradient to learn from. To avoid this potential problem, a modified ReLU called the Leaky ReLU [12] is used in this work. In the Leaky ReLU, below the threshold is a linear function with a relatively small slope.

Initially, perceptrons were a single layer deep, with the input vector being fed into multiple, parallel perceptrons. The output of these functions was then combined to produce a prediction. It was quickly realized that such a network was, at best, a linear classifier [13] that could not integrade complex logic such as exclusive-or logic. This logic-type is a crucial operation for complex, non-linear, classification. It was shown that combining multiple layers of perceptrons, i.e. perceptrons that fed their outputs into other perceptrons, did allow for non-linear classification. These are called multi-layer perceptrons (MLP) as opposed to single-layer perceptrons. See Figure 2.4 for a visual of how these networks are constructed. The increasing complexity presented new challenges, the most difficult being how to effectively adjust the weights and biases of the network, known as training, to produce the desired output.

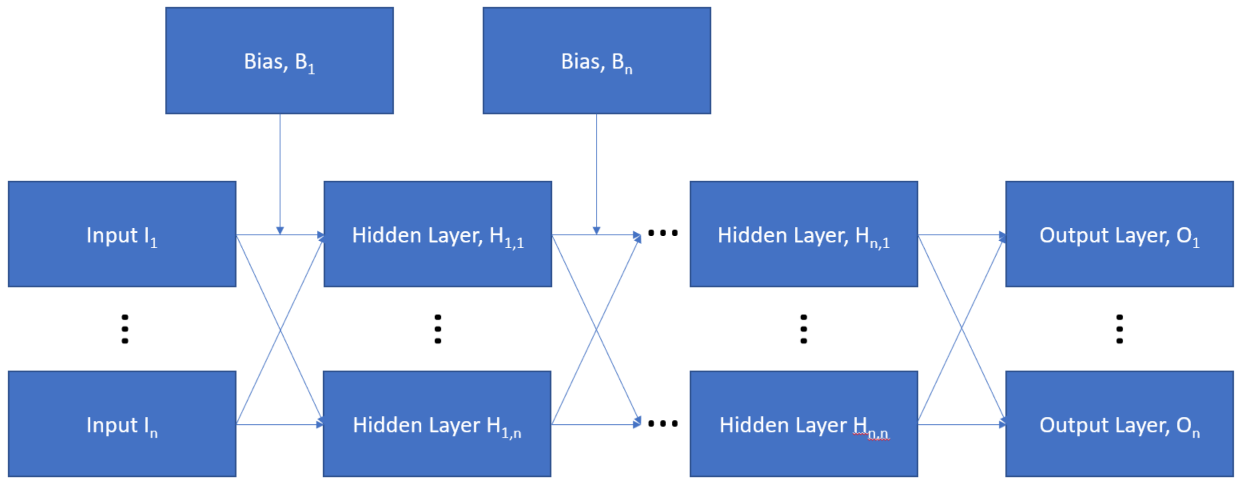


Figure 2.4: Multi-Layer Perceptron Overview

The weights and biases of perceptrons were initially set by hand using the Hebbian method, and then through various methods [14] until 1982 when a method call backpropagation was first applied to the training of neural networks [15]. With the backpropogation algorithm [16], the gradient of the weights and biases in the neural network are computed with respect to a loss function. These gradients are used to update the weights and biases of the networks starting from the output backwards towards the input. In this work, backpropagation by the ADAM method [17], a stochastic gradient descent algorithm, is used.

The training of MLPs is separated into the forward and the backwards pass. In the forward pass, a set of training data is inputted into the network to produce and initial output or prediction from the network. Training data are sets of known inputs and their corresponding outputs. After the forward pass, in the backward pass, the weights and biases of the network are updated by backpropagation to minimize a loss function. Common loss functions are mean average error and mean squared error (MSE). The training of MLPs is done using subsets of the training data, known as a batch, at a single time. A batch of inputs are fed through the network (forward pass) and then the weights and biases are updated by back propagation. Training a neural network with all of the training data is called an epoch. Often, complex networks are trained for thousands or more epochs.

When training a neural network like this it is important to withhold some of the training data into a test data set. The test data set is used to evaluate the network on data that it has not been trained on. Large networks can easily ”memorize” the training data. This is called over-fitting. Often when over-fitting happens, the network has not actually learned useful features necessary for the prediction and it will be unable to accurately predict non-trained-on data examples. Withholding some of the training data and preferring a network which more accurately predict the outputs of that data produce a more robust network. This training loss can also be used to stop training early if the network is no longer reducing the loss on the testing data.

2.1.5 Convolutional Neural Networks

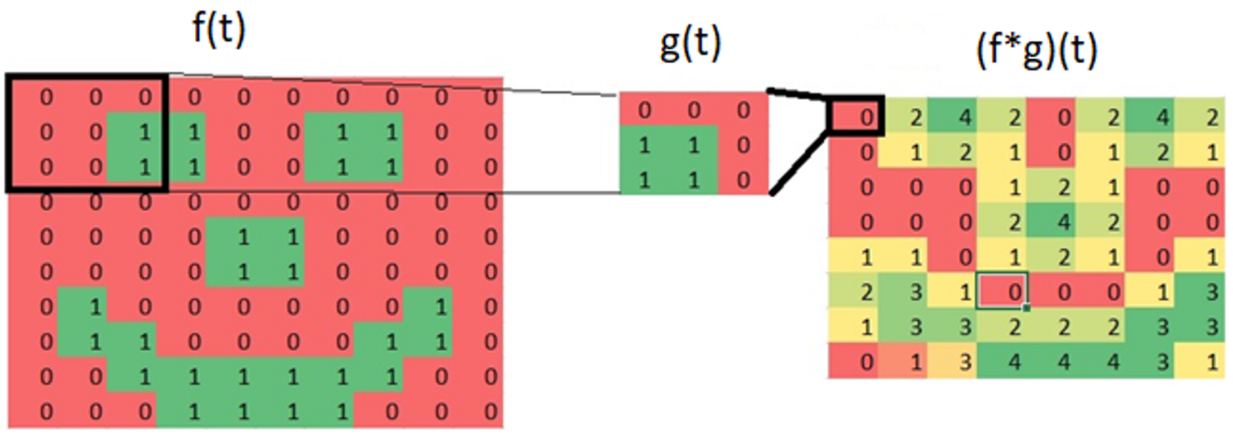
A convolutional neural network (CNN) [18] is a popular machine learning architecture that builds upon the MLP. It is used to learn spatial features within data in order to make prediction. Examples of data that CNNs have been trained on are image classification [19], natural language processing [20], and even complex games such as Go [21]. In all of these tasks, the relative position of the pixels, words, or game pieces are integral to the network producing the desired result.

CNN's are similar to MLPs but differ in a few important ways. The architecture of a CNN is comprised of several layers including convolutional layers, pooling layers and MLP layers. The convolutional and pooling layers are where the spatial features are extracted and the fully-connected layers (similar to the feed-forward network architecture described above) are where the learned features are combined to make a prediction.

Like other neural network architectures, CNN's are built using input, hidden and output layers. But unlike other neural network architectures, there are two stages of hidden layers. The first stage is the feature extracting stage where convolutional, non-linearity and pooling layers are combined to produce a feature map of the data. The input to the convolutional layers are 1D, 2D or 3D matrices which conserve the spatial information of the data. Convolutional layers are sets of learned filters which are convolved across the input to produce an activation map. These filters are learned/updated through backpropogation. The convolution operation is simply taking the dot product between a subset of the input and a filter. This process is repeated by shifting the filter across the input data. An example of this operation is given in Figure 2.5.

Next, in the non-linearity layer, a non-linear function is applied to each value in the activation map. There are several types of functions that can be applied in this layer, but for this work the ReLU function is used. The final layer in this stage is the feature pooling layer. The goal of this layer is to down-sample the activation maps while preserving useful information. This is done by applying a function to, usually, non-overlapping sets of features. Figure 2.6 shows an example of a max pooling layer where the maximum value within each

H



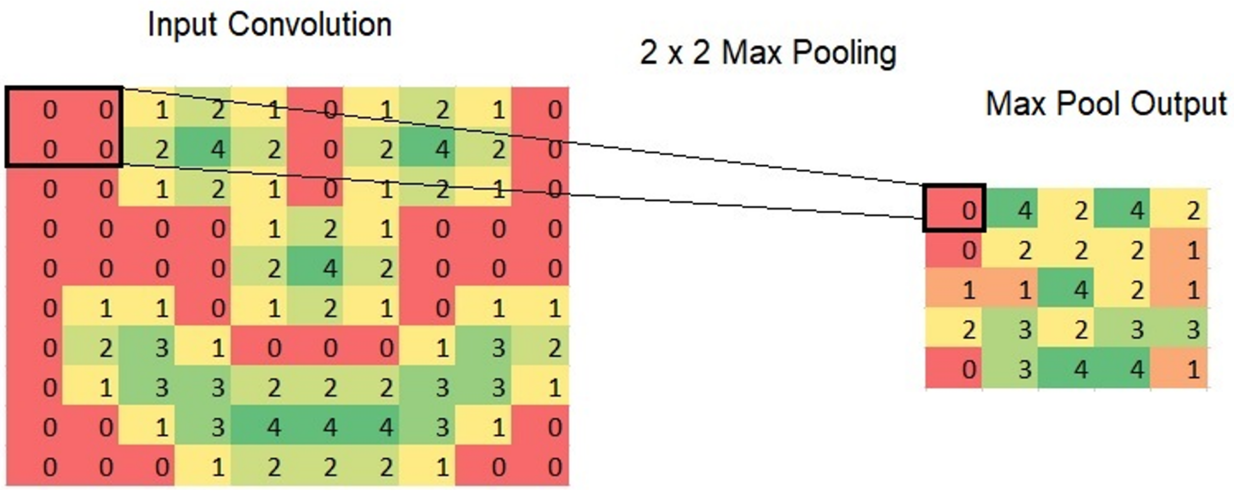
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0
0	0	1	1	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0	1	0
0	1	1	0	0	0	0	1	1	0
0	0	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0	0

0	0	0
1	1	0
1	1	0

0	2	4	2	0	2	4	2
0	1	2	1	0	1	2	1
0	0	0	1	2	1	0	0
0	0	0	2	4	2	0	0
1	1	0	1	2	1	0	1
2	3	1	0	0	0	1	3
1	3	3	2	2	2	3	3
0	1	3	4	4	4	3	1

Figure 2.5: Convolution Operation Example

H



0	0	1	2	1	0	1	2	1	0
0	0	2	4	2	0	2	4	2	0
0	0	1	2	1	0	1	2	1	0
0	0	0	0	1	2	1	0	0	0
0	0	0	0	2	4	2	0	0	0
0	1	1	0	1	2	1	0	1	1
0	2	3	1	0	0	0	1	3	2
0	1	3	3	2	2	2	3	3	1
0	0	1	3	4	4	4	3	1	0
0	0	0	1	2	2	2	1	0	0

0	4	2	4	2
0	2	2	2	1
1	1	4	2	1
2	3	2	3	3
0	3	4	4	1

Figure 2.6: Max Pool Operation Example

2 x 2 set of data is preserved. Like in a MLP, the stacking of these layers gives the network the ability to take into account more complex features in the data. Lastly, the output from the convolutional layers is vectorized and fed into an MLP, which combines these features in order to produce an output.

2.2 Literature Review

In this section, a survey of relevant applications of genetic algorithms to nuclear engineering systems will be evaluated and compared to the method proposed in this proposal. In addition, examples of using convolutional (and other types) of neural networks to predict nuclear characteristics will also be discussed and compared to the proposed method. The breadth of study in the optimization of nuclear systems is broad and so the following examples are just a few of the many works in this field.

2.2.1 A New Approach to Nuclear Reactor Design Optimization Using Genetic Algorithms and Regression Analysis

In this paper [22] a multi-objective genetic algorithm is applied to a multi-physics-based design of a gas cooled fast breeder reactor that indirectly couples a thermal-hydraulic solver for heat transfer and hot channel analysis (JAVA based) to a single fuel pin model and a whole core MCNP6 calculations. The design space of the reactor design problem is defined by the radius of a fuel pin, the mass flow rate of coolant, the isotopic enrichment of the fissile material in the fuel and the temperature of the coolant at the inlet of the reactor. A spline based multivariate regression is built from a set of trial data. These spline functions are then used as a surrogate model for both the Monte Carlo and thermal hydraulic solvers. A set of ten objective functions with equal weight are summed to produce the total fitness score. Some interesting conclusions are drawn from the data including stating that the optimal solution had been found. This is a statement made as a fact that is generally not verifiable with 100 percent certainty given the large design space. Their argument is that because the final individual with the highest objective function had an input variable (pin radius, enrichment, etc.) within a single standard deviation of the average of all individuals evaluated and so therefore is near the optimal individual.

The work varies from the work proposed here in that the surrogate models that they are building are of polynomial functions built by varying single variables within the design space. This type of method is valid and has shown to be effective in other applications but imposes the assumption that the system is easily defined by polynomials and that there are no second order effects of these changes on the other functions (i.e., that varying the pin radius does not change the channel heating function). The final individual that maximizes the summed objective function is not reevaluated with either the thermal hydraulic or neutronic solvers to verify the individual's objective function.

2.2.2 A New Approach to the Use of Genetic Algorithms to Solve the Pressurized Water Reactor's Fuel Management Optimization Problem

This paper [23] implements a GA for a nuclear fuel management optimization. It is a relatively early paper published in 1999 that presents both an example implementation of the genetic algorithm for the reactor management optimization problem and applies their algorithm to the Travelling Salesman Problem (TSP) as a benchmark to validate their new heuristic, the List Model, against previous implementations of genetic algorithms. They compare their fuel management optimization problem to the TSP problem in that due to varying depletion effects, each depleted fuel assembly is unique and cannot be placed repeatedly, like how the travelling salesman must visit all cities and not visit the same city twice. Previous implementations have used various heuristics to ensure each child made from crossover is a "valid tour", and in this paper they introduce what they call a List Model (LM) to accomplish this. The LM is a method of coding and decoding the genes and genotypes of individuals that ensures that each is a valid solution to the fuel management problem. This type of coding/decoding is not necessary in the work described in this proposal due to it being a beginning-of-life optimization with only four possible material types. Optimizing a system where each fuel assembly is unique would require a similar heuristic to ensure that input variables are not used (visited) multiple times.

In the TSP the task is to minimize the travel distance that a salesman must take to visit all cities while not visiting the same city twice. A 30-city benchmark with a known minimum distance was chosen. In this study, their implementation of the LM heuristic compares favorably with other, benchmark, implementations of Random Keys and Order Crossover. The fuel shuffling problem is not truly exactly analogous to the TSP because in the TSP the distance between all cities is fixed, but in the fuel shuffling problem the objective function evaluation is dependent on which both fresh and depleted fuel assemblies are placed where in the reactor. This is like the FNS problem in that the worth of any given material in any given location is dependent on the other materials in the system, so having a fuel versus a moderator plate in a location is dependent on what flux is going to be entering the plate, which is dependent on what plate materials that flux passed through near the plate. This type of relationship between the input variables of the system makes this problem a difficult one to solve.

2.2.3 Improvements, Validation, and Applications of a Metaheuristic Optimization Method for Neutron Spectra Tailoring at the National Ignition Facility

Dr. Bogetic's PhD Thesis [24] focuses on the optimization of neutron flux matching by experiments at the National Ignition Facility in Livermore, California. This facility produces a burst of mostly mono-energetic 14.1 MeV neutron source with a fluence of 10^{16} n/cm2 over 100 picoseconds. The optimization problem is to modify that mono-energetic neutron source into an exit spectrum of interest using a set of attenuating materials. The experiment being optimized also shares the volume with other experiments that may impact the neutron spectra and therefore must be modelled as well. A genetic algorithm was used to explore the design space of this problem, which was limited due to volumetric and mass constraints, among others. This optimization was limited to choosing up to ten materials from a large potential set of materials. One other notable thing implemented in this code is the use of a coupled deterministic and Monte Carlo solver ADVANTG and MCNP. ADVANTG is a deterministic code for generating both energy and space dependent mesh-based weight windows for use in MCNP. This type of variance reduction reduces the uncertainty on the neutron tally in MCNP and effectively accelerates the calculation. Variance reduction like this would help accelerate the MCNP FNS calculations. This implementation of a genetic algorithm for neutron spectra shaping is similar in many ways to the FNS optimization and goes beyond it in some respects as far as actually implementing the designs in actual experiments and validating the models used. No attempt to accelerate the optimization is made, though the possibility of acceleration with neural networks is mentioned as an avenue for future study.

Another notable implementation in the code is the use of operators to help diversify the population before the objective function evaluation. Instead of using a random walk to decide the specifications of a child made by mutating a single parent, (i.e. when the chance for crossover is less than 1 and there is a chance for a child to be created exclusively from a single parent) the mutations are chosen not by a uniform distribution but by sampling a Levy Flight distribution. This distribution is used in many various approximations of the physical and natural world. For example, foraging patterns of sharks in nature can be described by a combination of both Brownian motion and the levy flight. In essence, it gives a larger chance for a larger step to occur than would with either a true random-walk or a Brownian distribution. This type of mutation may lead to more effective exploitation of the design space.

2.2.4 Multi-objective Optimization Strategies for Radiation Shielding Design with Genetic Algorithm

In this paper[25] a multi-objective genetic algorithm is implemented to optimize a nuclear shield. The objectives the optimization is to optimize a design for the Savannah reactor which would simultaneously minimize mass, volume and both the gamma and neutron dose through the shield. This is also a multi-variable optimization that combines up to 15 different materials in order to meet the objectives.

In this study, two genetic algorithm methods are analyzed. The first, what they call a "classic multi-objective genetic algorithm", takes the three objective functions multiplied by weighting factors and combines them into a single value which is then minimized. The pros and cons of such a method, such as the inherent knowledge of the optimization domain required to select these weighting factors is discussed. A suite of combinations of weighting factors are used and compared. The second genetic algorithm used is what they call the "evolutionary multi-objective optimization strategy". This method implements the NSGA-II algorithm, including both the non-dominated sorting and crowding distance calculation to produce the Pareto front of potential individuals. These methods are compared to each other. Other settings of interest are the use of chromosome encoding/decoding of the shielding materials' thickness and material type, a single-point crossover, and a mutation chance for each bit in the chromosome to swap from 0 to 1 or 1 to 0. MCNP is used as the neutronics solver. In conclusion they state that both multi-objective optimization algorithms they examined were able to produce satisfactory designs and that the use of optimizations like this (the use of multiple weighting factors and, separately, the use of NSGA-II) would be especially useful for optimizing systems where the designers have little knowledge of the design space.

In general, this study does not show much that is new to the nuclear field. The use of weighted objective functions is a common method for these types of optimization and NSGA-II, while not common in nuclear optimizations, is a benchmark genetic algorithm. The main takeaway from this paper is the large number of calculations needed using the "classic" method compared to the single run "evolutionary" application. A total of 15 iterations of the "classic" genetic algorithms were run using different combinations of weight factors to collapse the objective into a single value. Each calculation was of 200 generations and 100 individuals evaluated per generation, meaning a total of 300,000 function evaluations with MCNP were needed. The "evolutionary" calculation was of a single run of 200 generations and 100 individuals, meaning 20,000 objective calls. In the end, the NSGA-II algorithm produced more individuals that dominated the original Savannah nuclear reactor shield design by having both less volume and less dose outside of the shield.

2.2.5 Coarse-grained parallel genetic algorithm applied to a nuclear reactor core design optimization problem

In this paper[26] a parallel genetic algorithm optimization is shown which optimized a simplified pressurized water reactor. The main idea presented in this paper is a comparison of a previous genetic algorithm optimization of the same pressurized water reactor and one in which the objective function is parallelized, but all other functions (crossover, mutation, etc.) are on a single processor. In this study the genetic algorithm is run on a set of parallel processors where each individual processor had its own unique population, mutation and crossover operations. This type of genetic algorithm is called an island model genetic algorithm, where each computer, or "island", has a unique population which is evolved by the normal genetic algorithm methods. Individuals on each island are given a chance to migrate from island to island which increases the diversity of the populations of each island and expands the amount of design space covered by each calculation. In this study the best individual was transferred to the adjacent island every ten generations. This way not only are the objective functions parallel but so are the populations themselves with some amount of connection between islands, in this case over a local area network. The islands may also have a topographical or geographical component to them, in that those individuals from island one may only migrate to island two, individuals from island two may migrate to island one or three, etc.

For this optimization, a model with eight input variables including fuel radius, cladding thickness, moderator thickness, enrichment of zones one, two and three (inner to outer cylindrical sections of the reactor), fuel material (two choices) and cladding material (three choices) as defined by a binary chromosome encoding. The objective is to minimize the average peak-factor of the reactor with the constraints that the reactor must be critical (1.0 +/- 1%) and sub-moderated, providing a target average flux. A diffusion theory-based solver is used to calculate the objective function. The fitness function evaluation is designed such that if all constraints are met, then the value is simply the average peak factor. However, if any of the constraints are not met then a penalty is applied which reduces the peak factor. In this way a design with a favorable peak factor that meets the constraints is preferred by the algorithm.

Ten experiments were run for between 1, 2, 4 and 8 islands. These results were then compared to ten previous optimizations using a Master-Slave genetic algorithm run on a single computer. The results show that the parallelization of the problem results in significant speed increase that linearly increased as more computer "islands" were added. Even when computer time was held constant, the island genetic algorithm outperformed the standard genetic algorithm. This is attributed to the island feature of the genetic algorithm being better able to explore the design space with multiple parallel populations.

The parallelization proposed in this paper is meant to help address the limitations of a single objective function optimization by spreading out the calculation into many calculations that each, presumably, could be going about optimizing the system in a different way. The migration function helps normalize the calculation by sharing the best individuals between calculations, which increases the amount of genetic diversity and therefore the amount of design space evaluated during the calculation.

The amount that the FNS optimization may benefit from this sort of genetic algorithm is hard to judge but dividing the calculation into separate islands would likely increase the computer utilization and therefore increase the amount of the design space that is explored. The longest and most computationally intensive step of the current genetic algorithm for the FNS is the objective function evaluation where an MCNP calculation is run for every individual in the population that meets the k requirement. Each calculation is not overly converged, but the entire optimization must wait for all the individuals to be evaluated before continuing. By dividing the population across several islands, the total utilization of a high performance computing cluster, like the nuclear engineering cluster available at the University of Tennessee, would likely be increased. But it is not clear that the benefits seen by this algorithm in optimizing a single objective design space would translate to a multi-objective algorithm like the one proposed for the FNS.

2.2.6 Optimization of Transcurium Isotope Production in the High Flux Isotope Reactor

This PhD Thesis and accompanying paper authored by Dr. Hogle [27] outlines an optimization method for several potential objective functions related to the creation of Californium-252 in the High Flux Isotope Reactor at the Oak Ridge National Laboratory. A genetic algorithm coupled with a neural network surrogate model. The neural network used is a three-layer feed forward network trained by the Widrow-Hoff learning algorithm, a least-mean-square algorithm. This algorithm calculates the change in each neuron's weight and bias by multiplying the neuron's input, the error, and the learning rate together. This training method is one of the original proposed training methods for training neural networks (perceptron networks). A neural network was trained on 19 inputs including weights of various isotopes present in a target, the weight of the potential filter materials and the number of irradiation cycles. The network then predicted the mass of ten isotopes of interest. This network was trained on 2,500 examples of an irradiation cycle using CENTRM/ORIGEN codes in the SCALE package. The network was trained for 79 epochs on the 2,500 example vectors and was stopped when the error on the validation error no longer decreased. The network compared favorably to testing against both a KENO-VI and a CENTRM calculation with much faster evaluation time. This surrogate neural network model was then used as the objective function for a genetic algorithm with multiple objectives multiplied by a weighting factor and summed. This summed objective function was then minimized throughout the optimization. Care was taken to ensure that the neural network was not used to extrapolate and only used to interpolate based on the training data.

The optimization algorithm used in this PhD dissertation is similar to what is proposed in that a neural network will be trained to approximate an expensive calculation. The differences being how the data is made for the training of the surrogate neural network, what type of network is used, and the use of a multi-objective genetic algorithm versus using a single objective function that approximates a multi-objective objective function. In this work the data for the neural network was produced before the optimization began. This is logical since the neural network was to be a replacement of the objective function, not, as in the work proposed, a heuristic that accelerates the optimization. Doing something similar may be possible for the FNS but would likely require a large amount of training data and compute time. The feed-forward neural network architecture used in Dr. Hogle's work does not have an ability to extrapolate to unseen data. The network architecture in this proposal does have some limited ability to extrapolate to unseen data by learning "features" from the training data that are useful for making predictions of unseen configurations of the FNS. The use of a single objective function approximating a multi-objective function is a valid was of dealing with multiple objectives but requires either expert knowledge of the design space and an idea about where the optimal solution will lay in the design space or multiple calculations with multiple sets of weight vectors so that the design space is adequately sampled. The use of a multi-objective optimization such as the NSGA-II does not require this, since the goal is not to optimize to a single "best" individual but instead to optimize to a suit of potential individuals.

2.2.7 Automated design and optimization of pebble-bed reactor cores

In this paper[28] a genetic algorithm is used to explore the design space of a high-temperature gas-cooled nuclear recirculating pebble bed nuclear reactor. This optimization differs from the others in this section and that of the FNS in that it is not just steady-state or beginning-of-life behavior that is being optimized but instead the lifetime of the reactor. Another unique challenge of pebble-bed reactors is that the fuel is mobile throughout the lifetime of the reactor. These challenges are tackled in this paper by coupling a diffusion code, PEBBED, to a mathematical representation of pebble flow within the reactor both axially and radially. The optimizations presented were made on a single computer core with each evaluation of the fitness function taking approximately 100 minutes. A total of six generations were evaluated with 30 individuals in each generation taking a total of two weeks. The authors state that parallelization is a priority for future optimizations. Two examples of genetic algorithms for the optimization of pebble bed reactors are presented in this paper, but both examples are optimized twice with different fixed variables. The two examples vary in both reactor type and in fitness functions evaluated. The genetic algorithm combines multiple objectives into a single objective value by user provided weights.

The first design presented is of a low-power high temperature pebble bed reactor with a transportable vessel. In this optimization four objective functions were selected, pebble bed radius (110-160 cm), outer reflector width (20-60 cm), height of active core (1000-1200 cm) and uranium enrichment (12-16%). The goal of this optimization is to produce the highest thermal power pebble bed core which would fit inside of the largest train car able to be used anywhere in the United States. This initial optimization was judged to be lacking due to the "best" individual having a maximum fuel temperature during a depressurized loss-of-coolant accident than was unacceptable and the total width of the core was 15 cm too long for the train car it was being optimized for. They reduced the core power from 200 MW(thermal) to 175 MW(thermal) and reran the optimization. The resulting design from this case produced a narrower core with higher enriched fuel that met the constraints on the optimization.

The next design optimized by genetic algorithm is a 600 MW(thermal) pebble bed high-temperature gas-cooled reactor. The design features high outlet temperatures of 1000 degrees Celsius used for process heat and, unlike the previous optimization, the size of the pressure vessel was not constrained. Instead, the genetic algorithm was programmed to give marginally better rewards for lower radius designs. Like the previous optimization, this initial calculation resulted in a “best” individual that did not meet the requirements of the optimization. In this case, the best individual had a k-eff of 0.9694, far below the target k-eff value of 1.03. Adjusting the fuel enrichment to 10.4% from the initial 8% increased the k-eff of the best individual and, separately, decreasing the burnup to 43.6 MWd/kg HM from the original 80 MWd/kg HM. This increased the fuel in the reactor by enriching the feed stock and removing spent fuel more quickly.

This optimization is unique among the others evaluated in this proposal and different from the FNS optimization in many ways. The first and most obvious is that these studies evaluated not just the steady-state geometry of the core but the behavior of the core through its lifetime and evaluated the robustness of the core design to a depressurized loss-of-flow reactor accident. This expanded evaluation of the potential designs is done at considerable computational expense and required both relatively few individuals (30) and few total generations (6) while still requiring 2 weeks of compute time.

2.2.8 Application of a Genetic Algorithm to the Fuel Reload Optimization for a Research Reactor

In this paper[29] a multi-objective genetic algorithm using non-dominated sorting is applied to a fuel shuffling problem for a TRIGA MARK II reactor. The goal of this optimization is to produce a fuel shuffling scheme which maximizes k-eff, minimizes power peaking, minimizes number of fuel element swaps and minimizes amount of burn-up of any one fuel element. The k-eff maximization and reducing power peaking are opposing objectives. The requirement to minimize the number of fuel swaps is necessary because the shuffling of the reactor will be done by hand and so reducing the complexity of the shuffling operation is preferred to reduce the risk of a miss-loading of the core. The maximum amount that any fuel element can be burned up is because depleted fuel elements must be replaced with fresh fuel, so minimizing the total number of fuel elements needing to be replaced after any one cycle is advantageous.

The genetic algorithm employed in this model encoded the loading pattern into a chromosome. Mutation and crossover operations are done on the chromosome representation. Mutation is conducted by a binary swap of two genes in the chromosome. Single point crossover is used to create children. Elitism is also used where all non-dominated solutions are kept in an archive and are transferred into the parent population in each step. The authors ran multiple calculations with various crossover and mutation rates and reported that the best crossover rate being 0.5 and the best mutation rate of 0.01.

The methodology of this paper seems sound and the optimization behavior through the generations does improve for both the k-eff and the power peaking objective. One takeaway from this paper is that the study of various mutation and crossover rates provides useful information for the setting of those parameters for nuclear reactor optimization. In past optimizations of the FNS (See Appendices C and D) the crossover rate was fixed a 1.0, meaning all children were created by combining two parents. A crossover rate like this is useful when a large design space needs to be explored, but it can cause optimizations to stall because children with better fitness are not being created and so the computational time would be better spent exploring the design space around the parents (by random mutation alone).

2.2.9 Feasibility Study on Application of an Artificial Neural Network for Automatic Design of a Reactor Core at the Kyoto University Critical Assembly

This paper[30] presents a feasibility study on the design of a simplified version of a critical assembly located at Kyoto University. This optimization is similar in many ways to the one proposed for the FNS with some notable exceptions. The geometry is simplified to be an 11 x 11 grid where the center is a void and the remaining 120 locations could be either fuel, graphite, beryllium, polyethylene, or lead. This fueled section is approximately 50 cm tall and is surrounded on the X and Y sides by an aluminum reflector. In each of these 120 locations a total of 157 plates of 5.08 cm x 5.08 cm x 0.3175 cm dimension can be loaded. On the top and bottom of the core in the (Z direction) can be any of the five previously mentioned materials except fuel. The goal of the optimization is creating a pattern with a k-eff of 1.015 +/- 0.005 and to maximize the ratio of fast flux (defined as ¿1 MeV) to total flux. Because this is a critical assembly with reactivity controlled by control rods, the power of the assembly is assumed to be constant and so, unlike in the FNS calculation, the ratio of the flux in the eigenvalue calculation can be used and the expensive source calculation is not necessary.

The design space for this assembly is much larger than the FNS with a total of 120 locations radially (11 x 11 grid with the center being void) and up to 157 plates stacked axially. Along with the 42 possible materials for the upper and lower reflector the total number of possible combinations are $(5{,}157 * 42 + 4)^{120}$ which is approximately 1.2 x 10^{13313}. This makes the FNS look simple by comparison. This complexity is reduced by both separating the optimization into two steps, the first which optimizes between four and seven plate sections and the second which takes those sections and combines them into the full core. When combined into the full core a 1/8th core symmetry with 9 radial locations is enforced to further reduce the design space. The optimization algorithm used in this paper is broken into two steps, the first for the optimization of the unit cells and the second for the optimization of the entire core:

1.1. 50 Random patterns for each of the 4, 5, 6, and 7 plate unit cells are created for a total of 200 patterns. An optimization flag is set to “0”.

1.2. Each of these patterns is used to build a model of a single fuel assembly.

1.3. The k-eff and fast flux/total flux values for each of the patterns is extracted from the output. The fast flux/total flux value is used as-is and the k-eff value is divided by 2.0. This adjustment to k-eff is done to bring the k-eff values between 0 and 1, which is the range of the sigmoid activation function used in the neural network.

1.4. Two feed forward neural networks are built with sigmoid activation functions throughout and widths of 400, 80 and 1 neurons. This is a bit unclear, but the input to the network seems to be 80 one-hot-encoded values for a total of 400 0's and 1's corresponding to 80 regions in the core and five total material types. The authors state that these 80 regions are half of the core. These networks are trained on the random patterns made in steps 1-3.

1.5. A gradient descent is then done using the k-eff and flux neural networks as surrogate models. An initial core configuration which is entirely fuel is modified by a single plate change until all plates in the core have been evaluated. This is done for each of the objectives.

1.6. At this point the algorithm has produced two unique plate patterns for each of the four unit-cell types: one which maximizes the k-eff and one which maximizes the flux ratio while maintaining a k-eff above 1.20. These designs are compared to the previously evaluated patterns. If they are better than the previous generation (as evaluated by the networks) then proceed to step 7. If they are equal (or worse) proceed to step 8.

1.7. Five cases are created from the four unit-cell types and for each objective function. This is a total of 40 new patterns. Four of these cases are created by genetic algorithm mutation and crossover routines. If the optimization flag has been set to "1" (by previously proceeding to step 8) then nine cases are made by genetic algorithm for a total of ten. The algorithm then returns to step 2 where the newly created patterns are evaluated by MCNP and the networks are retrained.

1.8. If the optimization flag is "0" (i.e. that this step has not previously been reached in the algorithm) then it is set to "1" and go to step 7. If the optimization flag is "1" at the beginning of this step, then this section of the optimization is completed.

The fuel assemblies evaluated in the previous steps 1-8 are used then to optimize a full core as follows:

2.9. From the fuel assemblies produced previously the three highest k-eff assemblies and the three highest flux ratios (while meeting the k-eff constraint) are chosen.

2.10 Using these six best assemblies and fully loaded fuel and reflector assemblies a total of 50 patterns for the full core are randomly created.

2.11. MCNP is used to evaluate these 50 patterns for k-eff and flux in a single eigenvalue calculation.

2.12. Three neural networks are then created. These networks are for predicting k-eff, fast flux (¿1 MeV) and thermal flux (¡1 eV). These three networks have the same architecture as each other but are different than the previous step's networks. This three-layer (single hidden layer) network uses sigmoid activation functions and layer widths of 504, 24 and 1. The 504 width of the input is due to the core having 21 regions and 24 material locations (after one hot encoding).

2.13 A gradient method is then implemented using the trained neural networks as the surrogate model. Beginning with the fuel regions adjacent to the void boundary fuel assemblies are chosen which maximize k-eff. When the system has a k-eff above 1.02 the remaining reactor assemblies are chosen which maximize the flux ratio. Then, all fuel assembly locations (9 total) are re-chosen to maximize the flux ratio while maintaining the k-eff above 1.02.

2.14 This new core design is compared to previously created core designs and is kept if it is better as evaluated with the neural networks.

2.15 Twenty new cases are created by taking the core design and applying a 30% mutation chance to each of the fuel assemblies. Fuel assemblies are separated divided into either "Fuel" or "Reflector" groups depending on if the optimization that created them was maximizing k-eff or maximizing flux. Any assembly cannot mutate to a different assembly type but can mutate to a different pattern within their original type. Return to step 2.11 to evaluate the current 20 designs with MCNP.

The optimization steps implemented in this paper was reproduced here because it is most like what is being proposed for the FNS. The major differences are the use of feature extracting neural networks instead of a fully connected neural network (See section 5.1.1 for a study on those) and because we are using those networks, we are not subdividing the problem into first an optimization of the cassettes and then the entire assembly. This study does not mention any checks made on how the network is doing in predicting their objective functions either, if it is getting better as the amount of training data increases or not, and the amount of training data in this case is relatively small compared to what is proposed in this work. On the other hand, it is nice to see that this type of methodology is being explored by our peers, if in a slightly different way.

2.2.10 Convolutional Neural Network for Prediction of Two-Dimensional Core Power Distributions in PWRs

This study[31] examines the use of residual neural networks for the prediction of 2D assembly-wise power distributions and assembly-wise pin power peaking factors given an initial macroscopic cross section information, with the goal being to accelerate the optimization of loading patterns for the Korean Standard Nuclear Power Plant (OPR1000). This paper presents a modified convolutional neural network called a residual neural network (ResNet, [32]). The residual neural network is like the convolutional neural network but adds skip connections which, with a learned probability, allow the outputs of a given layer n-1 of the network to "skip" over the layer n in the network and be fed in as input (along with the output of the skipped layer) into the n+1 layer alongside the output of layer n. This behavior attempts to avoid the problem of vanishing gradients where the output of a given layer (usually in deep networks) may go to zero during training. By having the network be able to not only use the result of the previous layer as input but also the result of the layer before that, this problem may be avoided. Using skipped generations also initially makes a larger network smaller and easier to train. During training the network effectively grows in size as the skipping mechanism is reduced.

This ResNet is like those used in the 1D and 2D studies in this proposal, except for the use of the skip connections. The network architecture used in this paper has six total convolutional layers, skip connections and batch normalization. They compare this network architecture to a fully connected network. The input to these networks is five cross section values used in the diffusion code for a total of 17x17x5 input values. They utilize rotational symmetry to reduce the input size from 34x34x5. The five cross section values are fast/thermal nu-fission cross section, fast/thermal absorption cross section, and the fast to thermal scattering cross section. They also use rotationally symmetric padding which expands the input to be not just the quarter core but includes what fuel assemblies would be present in the next layer of assemblies as well.

The ResNet compares favorably to a feed forward fully connected network (MLP). It shows the ability to learn to predict the outputs faster and with better accuracy, shows a small ability to extrapolate (i.e., use learned features in a useful way) and does this in roughly the same amount of computer time. They trained these networks on three sets of data, at beginning of life equilibrium case, a depletion case and a depletion case with void boundary conditions on the outside of the geometry. On all three tasks, the ResNet outperforms the MLP with the ResNet having average absolute errors of 0.48%, 0.62%, and 3.27% while the MLP had 1.00%, 0.83% and 24.35%. These results are not surprising except that the third case with void boundary conditions is a test of how well the network is able to extrapolate to other geometries. As expected by the architecture, the MLP is less able to extrapolate to this new geometry and the ResNet is more able due to it applying learned features from the previous training set more effectively than the MLP.

The original residual neural network (aka the ResNet) compared very deep convolutional neural networks (34+ convolutional layers) with and without these skip generations and they showed that large networks are easier to train using this skip mechanism. The utility of this type of network vs. a standard CNN is not explored in this paper as they just implement the ResNet alone.

2.2.11 A Deep Learning Based Surrogate Model for Estimating the Flux and Power Distribution Solved by Diffusion Equation

In this paper[33] a convolutional neural network is used to predict both flux and power peaking values in a simplified 2D representation of a core divided into 9 x 9, 10 cm nodes. These nodes are fixed as either reflector or fuel material. An in-house diffusion solver is used to produce 2 group fluxes throughout the core and corresponding power peaking values. The neural network they use is a fully connected convolutional network that uses convolutional layers to down sample (feature extraction) and de-convolutional layers (the opposite of a convolution operation) to up-sample back to the 9 x 9 geometry. This way they produce not just a single value. They produced a set of 600,000 samples of the 9 x 9 geometry with varying cross section values in the fuel nodes and fixed cross section values in the reflector nodes. The geometry is reflected on the interior sides with fuel and void boundary conditions on the outside of the geometry with reflector material. This limitation reduces the complexity of the problem and enforces a "fuel on the inside, reflector on the outside" paradigm on the problem. They then split that set into equally sized training and validation sets which were used to train the neural network. They show similar results to what is presented in this proposal in that the CNN based network can be trained to predict the values of interest and an MLP network used for comparison overfits the data.

In summary, this paper shows a similar methodology to others in this field by using CNN's to predict distributions of values, such as Convolutional Neural Network for Prediction of Two-Dimensional Core Power Distributions in PWRs, discussed above. The difference in this case compared to the FNS application is the use of a diffusion solver which does not take explicit geometry or material values but instead takes as input collapsed cross sections which represent homogenized volumes in the core. The diffusion solver has the benefit of being much faster computationally than Monte Carlo codes and therefore the benefit, or increase in computational speed, is less. Also, no discussion of the true benefit of CNN's as feature extractors is presented and the neural networks are just shown to be able to be trained to predict a training set that is relatively limited by fixing the fuel (with variable cross sections) and reflectors (with fixed cross sections). In conclusion this paper agrees with the results shown in this proposal and is another interesting application of CNN's to nuclear problems.

Chapter 3

Methodology

3.1 Solving the Neutron Transport Equation

Any discussion of nuclear computer methods begins with a clear definition of the equation being solved for, the neutron transport equation.

$$\begin{aligned}\frac{1}{\nu(E)}\frac{d\Phi(r,E,\Omega,t)}{dt} &= -\Omega\cdot\nabla\Phi(r,E,\Omega,t) - \Sigma_t(r,E,\Omega)\Phi(r,E,\Omega,t) + S(r,E,\Omega,t)\\ &+ \int_{E'} dE' \int_{\Omega'} d\Omega'[\Sigma_s(r;\Omega'\to\Omega;E'\to E)]\times\Phi(r,E',\Omega',t)\end{aligned}$$

This equation describes fully each neutron in a given system by describing its position in space, r, its direction, Ω, the kinetic energy, E, and the time t which the neutron is at position r with energy E moving in direction Ω. The neutron flux, $d\Phi(r,E,\Omega,t)$, is the summation of both negative and positive terms which describe the possible addition and removal of neutrons from a system. Note that in this version of the neutron transport equation fission is treated as a special The neutron loss terms are negative and consist of the of leakage and neutron interaction terms. The positive terms are the source term which is a direct production of neutrons at r, E, Ω and how neutrons could be scattered from Ω' and E' to E and Ω.

The solving of the neutron transport equation can be done in several ways including by diffusion, discrete ordinance and Monte Carlo methods. This work relies on solving the neutron transport equation by the Monte Carlo method, which will be discussed in more detail. Monte Carlo methods are a class of algorithms which use random numbers to repeatedly evaluate what could be a complex function which will, given sufficient samples, approximate the function.

In this case, the complex seven parameter neutron transport equation is approximated by tracking the history of events in the life of a sufficiently large number of neutrons. The initial characteristics of a given neutron are given by probability distributions for location, energy and direction. Then, depending on the material cross sections that the neutron is traversing, the neutron travels a distance and either escapes the geometry (leakage) or interacts with the geometry material. The interactions are governed by energy dependant cross sections. During the lifetime of each neutron, the paths and interactions are collected to provide more information about the neutronic characteristics of the system. Monte Carlo methods are generally more expensive than other methods and can require large amounts of computer resources to produce accurate data. For this analysis the Monte Carlo codes MCNP and KENO-V (SCALE) codes were used and are discussed in more detail in a following section.

3.2 Genetic Algorithms

3.2.1 FNS Genetic Algorithm Objective Functions

The objective functions used in this analysis were the total flux in the experimental volume, the representativity of the neutron flux in the experiment volume compared to a target spectra and the delta in k-eff from replacing the material in the experimental volume with a target material. These objectives are directly related to the overall objective of maximizing the worth of the FNS experiments while minimizing the experiment time. Potential experiments that the FNS may be used for are time-of-flight experiments in a beam line driven by the FNS neutron flux, material irradiation and measurement and direct k-eff measurements by neutron measurement in detectors around the FNS.

The total flux in the experimental volume directly relates to amount of experiment time required for a given FNS experiment. A large neutron flux in the experiment volume would mean more irradiation of materials in that volume, higher readings in neutron detectors placed around the volume and higher flux in a beam line which could start in the volume. Shorter experiments also means more economical use of the neutron generator, which has a fixed lifetime. Total flux can be calculated directly using MCNP and SCALE (MAVRIC) using the source-driven capabilities in these codes.

While maximizing the flux it is also important that the uncertainty in k-eff (or other neutronic figure of merit of interest) due to nuclear data uncertainty is similar in some respect to that of the target reactor design. For this analysis, this is done by maximizing the representativity of the neutron flux in the experiment volume. Representativity, also known as the integral index E in the Scale 6.2.4 manual, is simply the cosine of the angle between two flux vectors. Representativity varies between -1 and 1 where a value of 0 means that the vectors are perpendicular and are therefore not similar. A value of 1 means the vectors form lines that point exactly in the same direction and are therefore similar to each other (i.e. the flux in each energy bins are similarly proportional to each other in each spectra). A negative value would mean that the vectors are anti-correlated. Practically, representativity values of neutron flux spectra vary from 0-1. This metric does not include the magnitude of the vectors.

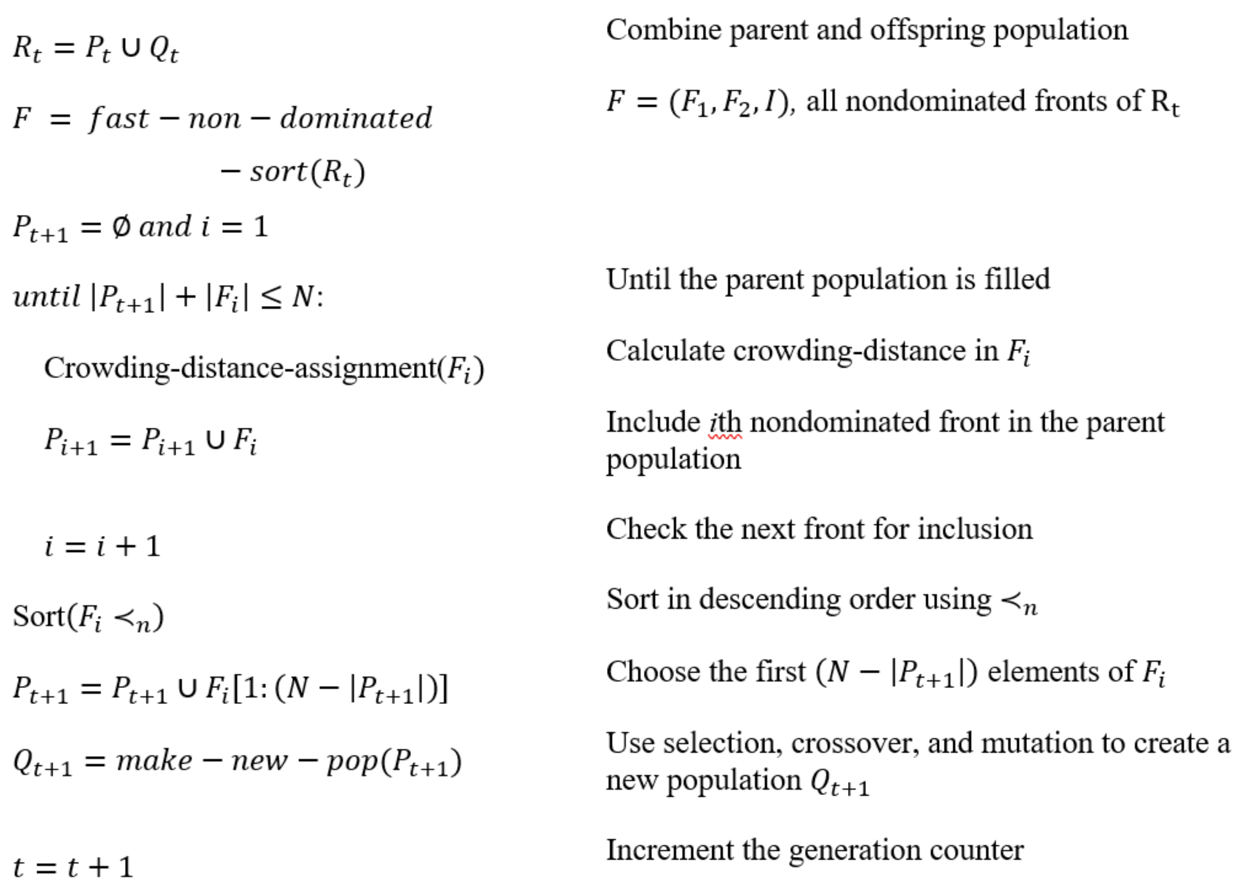
The last objective is referred to as the insertion experiment k-eff value. It is the difference in the calculated eigenvalue between the FNS with void in the experimental volume versus it completely filled with a target material. The goal of this objective is to push the population of FNS configurations towards those which are more sensitive to the change of material in the experimental volume. Having an FNS configuration that is more sensitive to the insertion of a target material increases the usefulness of the experiment towards the goal of reducing nuclear data uncertainty in that target material. Future studies that compare the nuclear sensitivities directly would further refine this objective. In the current optimization, this is an absolute value where both positive and negative k-eff values are considered equal but in the future it may be advantageous to prefer negative insertion experiment k-eff values.

3.2.2 The Non-Dominated Sorting Genetic Algorithm-II

The specific genetic algorithm used in this work is the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [1]. This algorithm, like other similar genetic algorithms [34], utilizes specific features which make it able to optimize multi-objective problems. These are the use of an elitist principle, utilizing a mechanism for preserving the diversity in the populations during optimization, and an emphasis on preserving individuals which are non-dominated by others in the population. The NSGA-II algorithm is presented in Figure 3.1.

Elitism is the inclusion of the previous generations' parents in the current selection of the current generation's parents. The use of elitism ensures that subsequent generations of individuals do not represent a worse Pareto front than previous generations. In this way, parents which may have the best objective evaluations are not lost from one generation to the next. A variation of this is to include all evaluated individuals in the selection of the current generations' parents. The drawback to this variation is that the list of individuals which are being sorted continues to grow each in each subsequent generation. Even with efficient sorting algorithms (such as the one discussed below) this can become a burden on the computational resources allocated to the optimization.

The next heuristic implemented in the NSGA-II algorithm is the non-dominated sorting algorithm. This algorithm produces a set of lists of individuals which are increasingly dominated by others in the group. Individual X dominates individual Y if individual X has one or more better objective evaluated compared to individual Y. The non-dominated sorting algorithm is presented in Figure 3.2. The algorithm is as follows, for each individual calculate the domination count, $\mathrm{n_p}$ which is the number of other individuals in the set which dominate solution p. Also calculate $\mathrm{S_P}$, which is a set of solutions that the solution p dominates. All individuals in which $\mathrm{S_P}$ equals zero represent the first non-dominated front



```
$R_t = P_t \cup Q_t$                                        Combine parent and offspring population
$F = fast - non - dominated - sort(R_t)$                    $F = (F_1, F_2, I)$, all nondominated fronts of $R_t$
$P_{t+1} = \emptyset$ and $i = 1$
until $|P_{t+1}| + |F_i| \leq N$:                           Until the parent population is filled
    Crowding-distance-assignment($F_i$)                     Calculate crowding-distance in $F_i$
    $P_{i+1} = P_{i+1} \cup F_i$                            Include ith nondominated front in the parent population
    $i = i + 1$                                             Check the next front for inclusion
Sort($F_i \prec_n$)                                         Sort in descending order using $\prec_n$
$P_{t+1} = P_{t+1} \cup F_i[1:(N - |P_{t+1}|)]$             Choose the first $(N - |P_{t+1}|)$ elements of $F_i$
$Q_{t+1} = make - new - pop(P_{t+1})$                       Use selection, crossover, and mutation to create a new population $Q_{t+1}$
$t = t + 1$                                                 Increment the generation counter
```

Figure 3.1: NSGA-II Algorithm [1]

```
$for\ each\ p\ \in P$
$S_p = \emptyset$
$n_p = 0$
 $for\ each\ q \in P$
   $if\ (p \prec q)\ then$                      If $p$ dominates $q$
       $S_p = S_p \cup \{q\}$                   Add $q$ to the set of solutions that are dominated by $p$
   $else\ if\ (q \prec p)\ then$
       $n_p = n_p + 1$                          Increment the domination counters of $p$
    $if\ n_p = 0\ then$                         $p$ belongs to the first front
       $p_{rank} = 1$
      $\mathcal{F}_i = \mathcal{F}_i \cup \{p\}$
$i = 1$                                         Initialize the front counter
$while\ \mathcal{F}_i \neq \emptyset$
    $Q \neq \emptyset$                          Used to store the members of the next front
    $for\ each\ p \in \mathcal{F}_i$
        $for\ each\ q \in S_p$
           $n_q = n_q - 1$
           $if\ n_q = 0\ then$                  $q$ belongs to the next front
              $q_{rank} = i + 1$
              $Q = Q \cup \{q\}$
      $i = i + 1$
      $\mathcal{F}_i = Q$
```

Figure 3.2: Non-Dominated Sorting Algorithm [1]

(aka Pareto front). Next, for each individual in the first front, the S_P set is cycled through and the n_p of that individual is reduced by one. The individuals which now have a domination count of 0 represent the next non-dominated front. These individuals are only dominated by the individuals in the initial non-dominated set. This is repeated until all individuals in the set are sorted into non-dominated ranks. Using the standard "big O" notation, this sorting requires $O(MN^2)$ evaluations, where M is the number of objectives and N is the number of individuals being sorted.

The set of parents of the next generation is created by adding individuals based on their non-dominated rank. If there are fewer places available for individuals than there are individuals in the current non-dominated rank then a decision of which ones to include must be made. This is done by initially taking all individuals in the current non-dominated rank. Then, if there is still room for more individuals, the crowding distance for each individual to each other individual is calculated. Individuals with the highest crowding distance are selected first. Crowding distance is an estimate of the density of solutions around each solution and is calculated by computing the average distance of two points on either side of a given solution with respect to each objective function. The total crowding distance is the sum of the crowding distances for all objectives. Note that in the calculation of the crowding distance that the values are normalized to avoid problems of objective functions on varying scales. The crowding distance algorithm is presented in Figure 3.3.

The NSGA-II is well-suited for optimizations of three or less objectives functions. With an increased number of objective functions other genetic algorithms may be necessary. Other genetic algorithms such as the more contemporary NSGA-III [35], which handles the increased complexity of having more than three objective functions by, in comparison to the NSGA-II, modifying the selection operator to which includes a metric for evaluating individuals based on their distance from user-defined reference points (or hyperplane) in the objective space. In this way, the algorithm will keep ensure the diversity of the population by preferentially keeping individuals which are close to these predefined points. Many multi-objective genetic algorithms have similar heuristics.

```
$l = |\mathrm{I}|$                                                  Combine parent and offspring population
$for\ each\ i, set\ \mathrm{I}[i]_{distance} = 0$                    Initialize distance
for each objective $m$
    $\mathrm{I} = \mathrm{sort}(\mathrm{I}, m)$                        Sort using objective values
    $\mathrm{I}[1]_{distance} = \mathrm{I}[l]_{distance} = \infty$    So that boundary points are always selected
    for $i = 2\ to\ (l - 1)$                                         For all other points
    $\mathrm{I}[i]_{distance} = \mathrm{I}[i]_{distance} + (\mathrm{I}[i].m - \mathrm{I}[i].m/(f_m^{max} - f_m^{min})$
```

Figure 3.3: Crowding Distance Assignment Algorithm [1]

3.3 Computer Codes and Modules

3.3.1 MCNP

Monte Carlo N-Particle (MCNP) [36] code is a Monte Carlo based radiation transport code produced and maintained by the Los Alamos National Laboratory. It has the capability to track many types of particles including, but not limited to, neutrons, gammas, and electrons. MCNP can be used for calculating source-driven calculations such as neutron dose, detector response and shielding analysis. It can also be used for criticality-related calculations such as criticality safety analysis. With only minor adjustments to a given MCNP input either a source driven calculation or an eigenvalue calculation can be performed. It also features a wide array of tallies and variance reduction techniques which make it a very powerful tool for neutronics analysis. The Evaluated Nuclear Data File version 7.1 cross section library (ENDF/B-VII.1) [37] was used for this work.

3.3.2 SCALE Package

SCALE is a modeling and simulation suite of codes used for neutronics analysis including reactor systems, critically safety evaluations and shielding, among many others [38, 39]. It comprises codes for tackling these tasks. The two modules most pertinent to this analysis will be explored in more detail here. Like with MCNP code, the ENDF/B-VII.1 library of cross sections was used for this analysis.

KENO

KENO-V is a Monte Carlo neutron transport solver included in the SCALE code package. KENO-V is commonly used for criticality safety, reactor analysis and reactor design. Both continuous and multi-group neutron cross section libraries are included in the installation of SCALE. It features full three dimensional modeling capability with many features built in for outputting relevant tallies and nuclear characteristics.

TSUNAMI

The Tools for Sensitivity and Uncertainty Analysis Methodology Implementation (TSUNAMI) [3] is the suite of modules within SCALE used for both sensitivity and uncertainty analysis. Sensitivities are calculated based on both eigenvalues and reaction rates. These can then be used for both experimental applicability and bias estimation. Within TSUNAMI, there are methods for producing sensitivities for 1D, 2D and 3D geometries with either multi-group or continuous energy data libraries. In this dissertation, the sensitivities of interest are the sensitivities of k-eff with respect to changes in macroscopic cross section. Calculating sensitivities requires two pieces of information, the forward and the adjoint flux. The forward flux describes where in a geometry neutron interactions occur while the adjoint flux describes how the figure of merit of interest is sensitive to neutron interactions throughout the geometry. With multi-group nuclear data, the adjoint flux can be calculated directly in KENO V, but with continuous energy nuclear data the adjoint flux must be approximated. Within TSUNAMI there are two methods for approximating the adjoint flux. They are the Iterated Fission Probability and the contributon-linked eigenvalue sensitivity/uncertainty estimation via track-length importance characterization (CLUTCH) methods [40]. These two methods produce the same sensitivities, but in general, the CLUTCH method is more efficient from a computational standpoint due to it being easily multi-threaded and because it requires less memory during the calculation.

With the CLUTCH method the sensitivities of k-eff to changes in macroscopic cross sections for all materials in the geometry are calculated in a single forward calculation. Sensitivities require an adjoint flux to be calculated, but this is not currently possible when using continuous energy nuclear data. The adjoint flux is approximated during the CLUTCH calculation by building a function which relates how many fission neutrons are produced after any given neutron reaction over a user-defined cuboid mesh. This function, called the F*, is computed after the initial skip generations and before the active generations of the calculation by tracking neutron interactions and how many fission neutrons are produced afterwards. In the TSUNAMI documentation the F* mesh size is generally 1-2 cm voxels which cover all fueled geometry and between 10-100 neutron histories are recommended for converging the function. After the F* mesh is converged, it is then used to estimate the contribution of every neutron interaction. Once sensitivities are calculated it is then required to verify that the sensitivities are correct by calculating the integrated sensitivities for the isotopes with the largest sensitivity by direct perturbation.

3.3.3 Python

Python [41, 42] is a popular computer programming language which supports both procedural, object-oriented and functional programming. The initial version of Python was released in 1991 and has been continuously updated since. It is designed to be a ”readable” language driven by language constructs such as required indentation.

Python is designed to be highly extensible by having the ability to easily incorporate code written by others in the form of importable classes and objects. It is regarded as a rather ”slow” programming language compared to others such as FORTRAN or C++. Python was chosen for this work due to its readability, the wide range of supporting documentation available for it and the author’s familiarity with it. Python version 3.7.4 was used for this work.

The CNN surrogate models used in this work are created using Tensorflow [43] library for Python. This library was produced by The Google Brain Team initially for internal use, but, since 2015, has been released for academic use. It provides a workflow and tools for training and the inference of deep neural networks. In practical terms, using Tensorflow accelerates all aspects of building and debugging neural networks, from data pre-processing to evaluating network architectures to post-processing of results of the networks. Training of the neural networks was accomplished both locally on the author’s personal NVIDIA GTX 1080 graphics card and/or on the NVIDIA Quadro P5000 located on the University of Tennessee Nuclear Engineering Cluster.

As described fully in Chapter 4, the optimization presented in this work relies on both text files with Python code run from the command line and Jupyter Notebooks [44]. Jupyter

Notebooks are a web-based document format for both executing and publishing many different code languages. In general, the bulk of the genetic algorithm optimization is run using Python code in text files while post- and pre-processing (including building of the surrogate models) is done using Jupyter Notebooks.

3.3.4 Matlab

The Interior Point Method [45] optimization presented in this work is implemented using a coupled Python and Matlab (version 2019a) code [46]. The Interior Point Method is implemented using the fmincon function within the Matlab Optimization Toolbox. The fmincon function is coupled to TSUNAMI-calculated sensitivities that are combined with a novel penalty function discussed in more depth in Section 5.1.

3.4 MCNP and SCALE Modeling

The FNS input is produced for both source driven calculations and eigenvalue calculations. The source-driven calculation is used for producing spectra-related data including representativity and flux values. The eigenvalue input is used to evaluated the FNS patterns value for both k-eff constraint and is for the insertion experiment k-eff objective function. The major contributors to k-eff and nuclear characteristics of the FNS system are included in the model including the reflector, the fuel/moderator plates and the aluminum cassette material. Shielding outside of the reflector, the concrete pedestal and the geometry of the neutron generator are not included.

3.4.1 Building the FNS in MCNP

The modelling of the FNS begins with the smallest component of the FNS, the plates. These plates are modelled as 0.5" x 6" x 6" cuboids which are defined in separate "universes" from each other in the MCNP model. These plates fill the entire volume of the cassette, leaving no gap between either the plates themselves or the plates and the cassette shell, as seen in

Figure 3.4. These plates are then placed into the cassette geometry using a 1D lattice to form a pattern for the cassette. Each cassette is created by defining the aluminum shell of the cassette and filling the inside volume with the appropriate cassette pattern using the 'fill' command. In the latest iteration of the FNS not all cassettes are exactly the same dimensions. The interior pattern 'A' cassette in zone 2 is able to be stretched or compressed during the genetic algorithm optimization. The size of cassette pattern 2A is a variable during the optimization. Because of this, the cassette pattern 1A in the Zone 1 is removed and the cassette pattern 2A in zone 2 is able to stretch as needed from Zone 2 to Zone 1. This stretching may not reflect the true geometry in all cases use to the stretching of the cassette potentially interfering with the shutdown plate between zones 1 and 2 if there are more than 20 plates in the pattern. See Figure 3.5 for a cut-through visualization of the FNS MCNP model with labels of the various parts of the geometry.

The remaining volume of the FNS inside the reflector is either void between the cassettes or a void shape for the shutdown plates. The carbon steel reflector of the FNS is also modelled. The purpose of this material is to reflect neutrons back into the FNS with minimal activation and neutron absorption. Figure 3.5 shows the reflector. Earlier iterations of the FNS MCNP model included the concrete pedestal, but during optimization this is voided out to decrease calculation time by not tracking neutrons which enter the concrete pedestal. In the work presented where only the interior 3 x 3 cassette patterns in each zone can have the full range of plate materials, this is appropriate. If the full 5 x 5 cassette pattern is used then the concrete pedestal should be added back into the model. During the optimization the shutdown plates are not modeled. The instrumentation of the FNS which may require access through the reflector are not modeled.

In the source-driven variation of the FNS MCNP input a point source is placed within what would be cassette pattern A of Zone 3 at about 2/3 of a zone width into the FNS. This placement is rather arbitrary as the actual location of the source in the geometry is not known at this point. The source is a DD neutron generator produces a 2.45 MeV neutron at 4e9 per second. The neutron generator used in these calculations is the DD109X, produced by

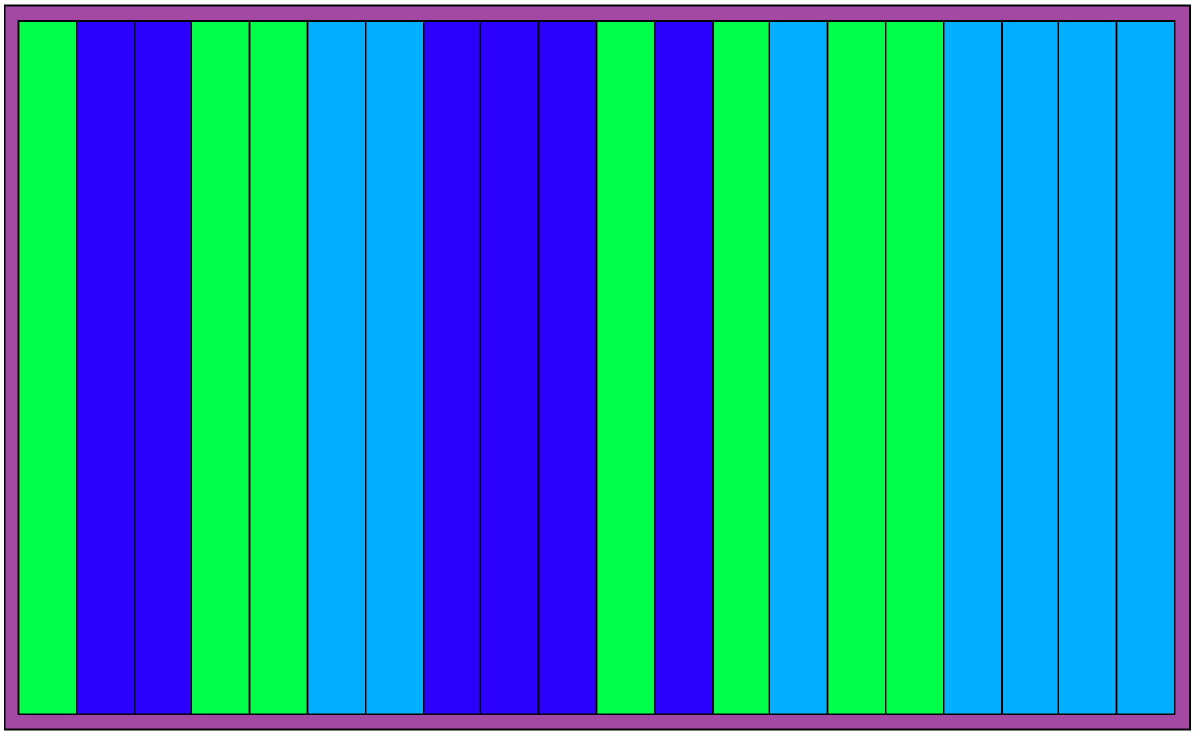


Figure 3.4: View of MCNP Cassette

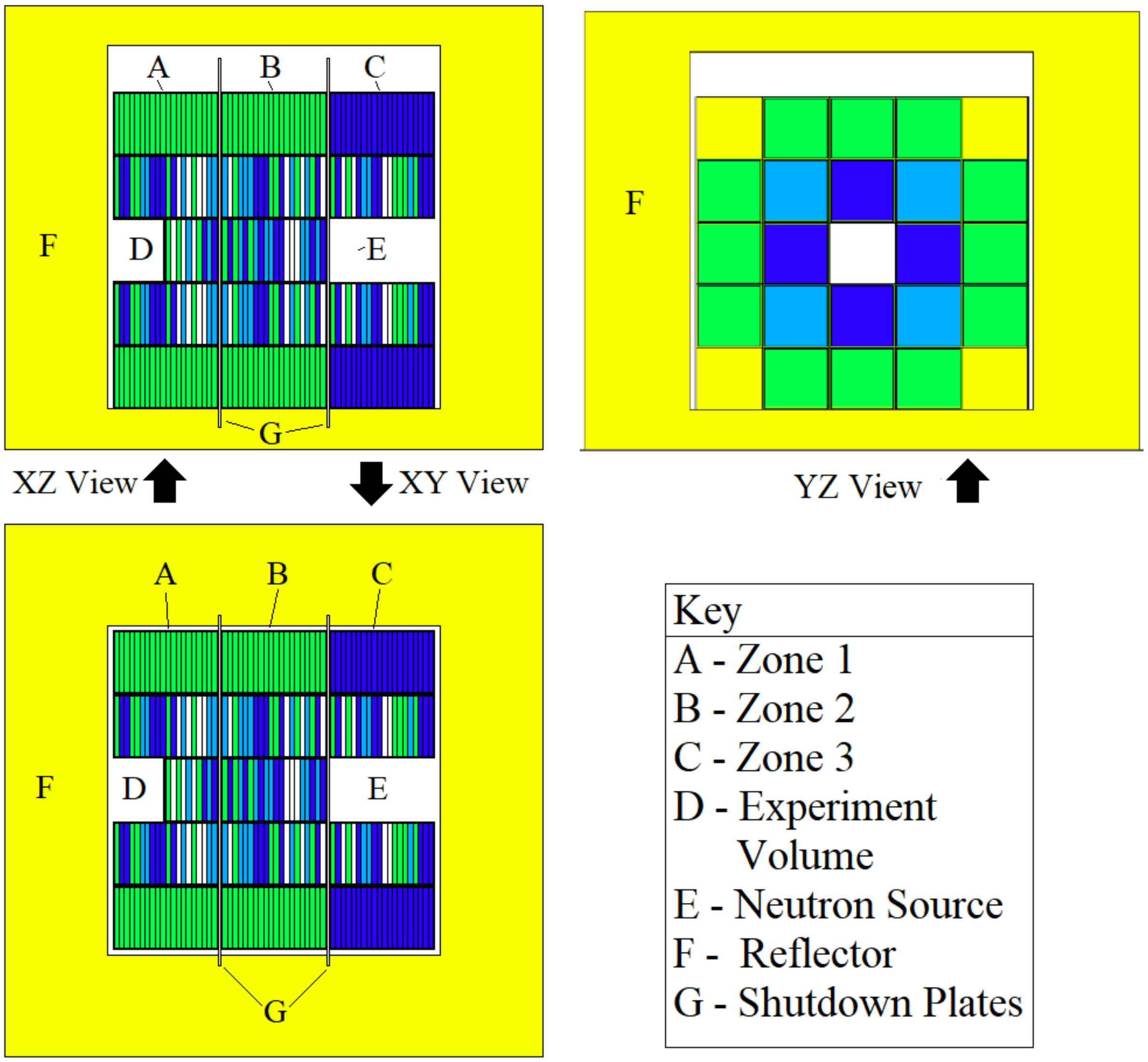


Figure 3.5: View of MCNP Model

Adelphi [4]. The geometry of the DD neutron source is not modelled. These simplifications are judged to be acceptable at this stage of the design of the FNS and future, more exact, modeling will be required for final design and construction.

An example of a commented FNS MCNP input is included in this dissertation as Appendix A.1. Note that during the genetic algorithm optimization a template based on this input was used that varies from the one presented, which has been modified with additional comments for clarity. In the model used during the generation several keywords are added to make inserting unique FNS patterns a simple task.

The construction of this model makes k-eff and source-driven calculations in MCNP relatively quick with k-eff calculations requiring less than 5 minutes to get to 75-100 pcm uncertainty and source-driven calculations requiring ¡20 minutes for a sufficiently converged neutron flux tally in the experimental zone of less than 1%. This model does not feature any variance reduction which would further increase the efficiency of running these models.

3.4.2 SCALE Model

KENO-V is a Monte Carlo solver in the SCALE code package. It is used primarily in criticality safety and reactor calculations. A model of the FNS was built using the same physical and material dimensions and definitions as the MCNP model. The single exception being that because KENO-V is an eigenvalue solver, it was not necessary to variably change cassette pattern 2A in size. Instead, when the pattern for cassette 2A is less than 20 plates, void plates are added to the pattern. The physical dimensions of the cassette do not change. An example of an FNS SCALE input can be found in A.2.

The KENO-V model of the FNS was built with a similar process as the MCNP model and with the same geometry as the MCNP model including the cassettes and the carbon steel reflector. The cassettes are modelled in the same 5x5 pattern and in three zones. The concrete pedestal, shutdown plates and experiment volume are not included in this model. The cassettes are modelled using the 0.5” x 6” x 6” dimension and are placed in the model using the ARRAY card. These patterns are then placed in their own two dimensional arrays which represent the zones themselves. The zones are then placed into the FNS. This model can be seen in Figure 3.6. An example of a SCALE KENO-V model can be found in the Appendix.

The final 80 parents of the accelerated FNS optimization were modeled with this KENO-V input. The KENO-V solver consistently calculated k-eff values lower than the MCNP calculation, with 78 of the 80 parents having a difference in the k-eff of less than 0.01 delta k-eff, with 26 of them having a delta k-eff of less than 0.005. This bias may be due to slight difference is likely due to the use of ENDF/B-VII.1 cross sections in the KENO-V model, while the MCNP model uses ENDF/B-VII.0.

A second SCALE/KENO-V model of the FNS was developed as well in which all plates within a single cassette type are homogenized into a single volume. This model is used for both KENO-V and for the TSUNAMI calculations. A KENO-V model can be fairly easily modified into a TSUNAMI-3D model by the addition of specific cards such as updating the parameter card to include the hyper parameters for the version of TSUNAMI being used. The geometry and material definitions are exactly the same between the two codes. The parameters and changes needed to modify a KENO-V input into a TSUNAMI input are detailed in Section 5.2.

3.5 Overview of Tools for Optimization

In this chapter, the tools and methods for running building the surrogate models and running the FNS genetic algorithm are presented.

3.5.1 Surrogate Modeling and CNN Tools

Building the CNN surrogate model requires the following steps:

1. Create training and testing data set either by collecting all data produced by a genetic algorithm.

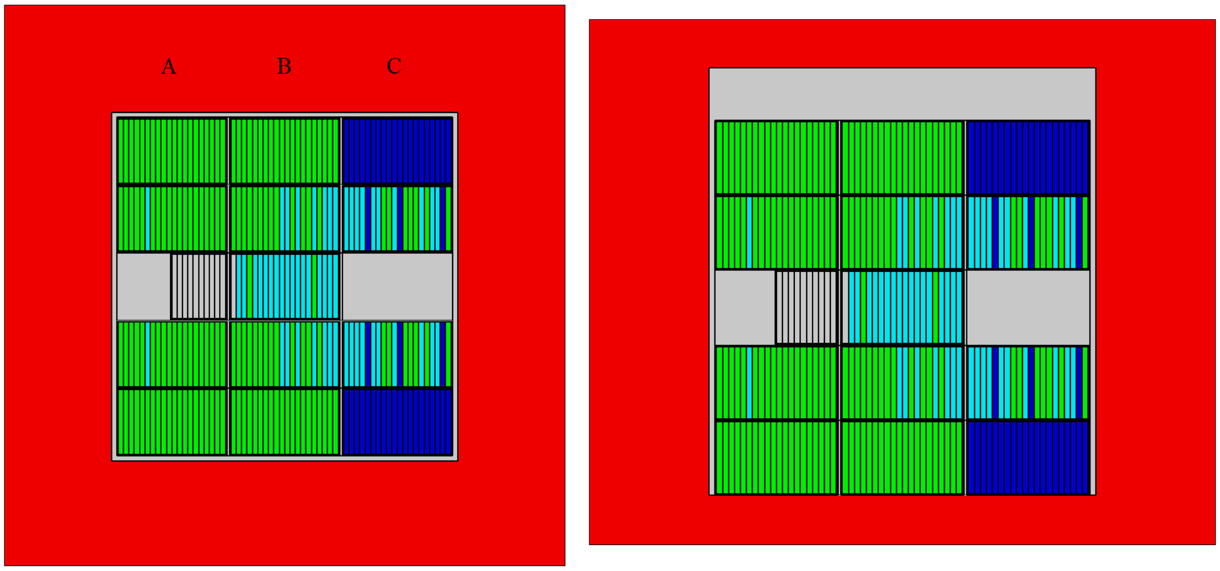


Figure 3.6: XY (Left) and YZ (Right) View of SCALE Model

2. Process training and testing data into a matrix of the form (A x A x B x C), where A is the width of the FNS model, B is the number of plates across the FNS model, and C is the number of material types.

3. Hyper-parameter optimization of surrogate models using training and testing data from step 1.

These three steps and the scripts used to accomplish them will be discussed in the follow sections. The surrogate models built for Chapter 6 are used as examples throughout this section.

Creating Training and Testing Data

The data used to create the surrogate models are created by optimizing the FNS using a standard NSGA-II algorithm. This algorithm produces a suite of Pareto front individuals which represent the trade-off in the objectives of the optimization. In the FNS case the objectives are the total flux in the experiment volume, the representativity of the neutron spectra in the experiment volume versus a target spectra and the change in k-eff due to filling the experiment volume with a target material. In addition to the objectives, a k-eff of 0.95 constraint is also enforced and a surrogate model to evaluate this constraint is created. These objective and constraint functions are explored further in Section 2.1.1.

A total of 8,100 individual patterns were evaluated in the NSGA-II optimization of the FNS used to produce the training, validation and testing data sets for the FNS surrogate models. The data is compiled into two sets, one which is of the entire 8,100 individuals evaluated for k-eff and the other which is the subset of those individuals which meet the k-eff constraint of 0.95. These two data sets are then subdivided by random selection into training and testing data sets. All of these data are stored in comma separated (csv) files with the pattern in the first 150 columns and the values in the next 1-3 columns.

Pre-processing of Training and Testing Data

The separated training and testing data "csv" files are then processed into a form that can be used as an input into a neural network. The first step is to perform a one-hot encoding of the vector of material numbers describing each pattern and then to transform the now 2D vector into a 4D vector describing the location of the plates in 3D space relative to each other.

The first step in pre-processing the training and testing data is to perform the one-hot encoding operation on the material matrix for each pattern. These patterns are then one hot encoded in order to be read in and effectively used by the CNN models. One hot encoding removes any perceived relative information between categorical data. For example, in the FNS design a void plate is designated as material 1, polyethylene as material 2, enriched fuel as material 3 and the target moderator as material 4. These numbers are how the materials are described in the MCNP/SCALE models but material 1, void is not more similar to material 2, polyethylene and less similar to material 4, the target moderator. But, a neural network that is just looking at the numbers may try to use those values as a source of information, when none actually exists in those numbers. One hot encoding removes the relative information by transforming the N sized vector of numbers describing the materials into a N x M vector which M is the total number of categories. For example, given four possible material types, a material that is described by a 2, would become the vector [0, 1, 0, 0]. Doing this increases the size of the input from a vector of size 150 to a 2 dimensional matrix of size 150 x 4.

These 2 dimensional matrices are then converted into matrices of size 60 x 2 x 2 x 4. These numbers represent the total number of plates cross the X dimension of the FNS (3 cassettes of 20 plates each, 60), the Y and Z dimension of a corner of the FNS and the 4 possible material types (void, polyethylene, enriched uranium fuel and sodium). The orange box in Figure 3.7 shows the 2 x 2 symmetry of the FNS which makes this possible while the purple box shows what would be required for modelling the entire FNS. The training and testing data sets are then stored as "hdf5" files for later use.

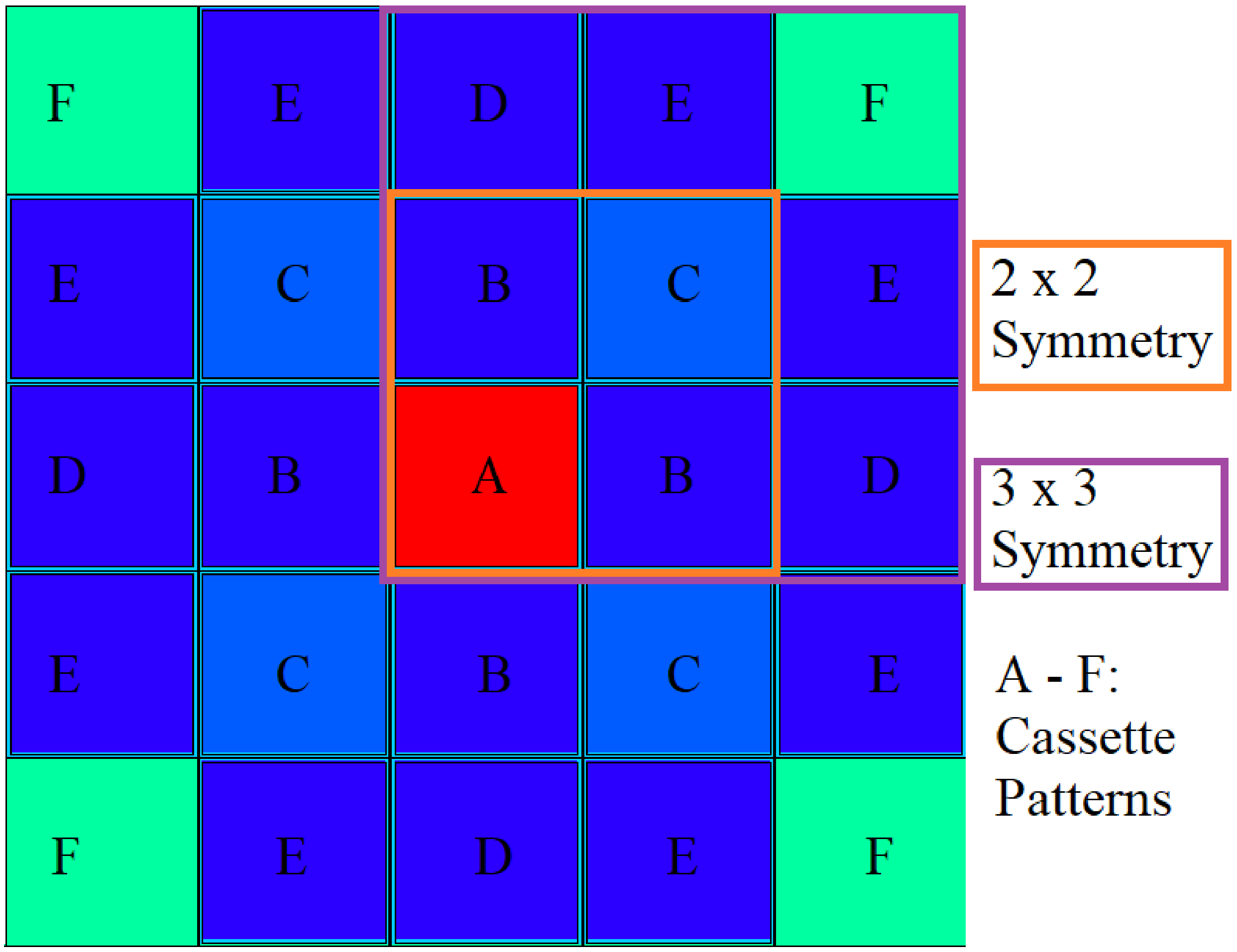


Figure 3.7: FNS 2 x 2 and 3 x 3 Symmetry

Training of the Surrogate Models

The training of the surrogate models is completed in two stages. The first is a hyper-parameter search using the training and testing data and the second step takes the top ten architectures from the hyper-parameter search and trains the networks for an extended number of batches. The networks which have the lowest loss on the training data set are then selected as the surrogate model architectures.

The KerasTuner python library is used to perform the hyper-parameter search of the CNN architectures. This library provides a wrapper for defining minimum, maximum and minimum change size for various hyper-parameters associated with the CNN surrogate models. Figure 3.8 shows the general algorithm for the hyper-parameter search. Hyper-parameters are chosen, the architecture is trained and then evaluated. The choice of the hyper-parameters is done by the 'tuner' object within KerasTuner library. Three tuners are available in KerasTuner; the random search, Bayesian, and hyper band algorithms. The Bayesian optimization algorithm with the default options is used for finding the hyper-parameters in this work, as discussed more thoroughly in Section .

3.6 Using the FNS GA Code

The FNS genetic algorithm code is a series of Python classes which interact to perform all facets of the optimization. Table 3.1 summarizes all of the Python input and other miscellaneous files used.

The main Python files used is Run_GA.py. This file describes the options of the genetic algorithm including number of generations, the various genetic algorithm options and debugging options. This file is run through the command line and it then calls on Genetic_Algorithm.py to run a genetic algorithm of the FNS. The other Python files are called as necessary to accomplish various tasks such as building/running/evaluating MCNP files, using the CNN surrogate files, and the crossover/mutation routines within the genetic algorithm. The files ending in "hdf5" are the first generation CNN surrogate models.

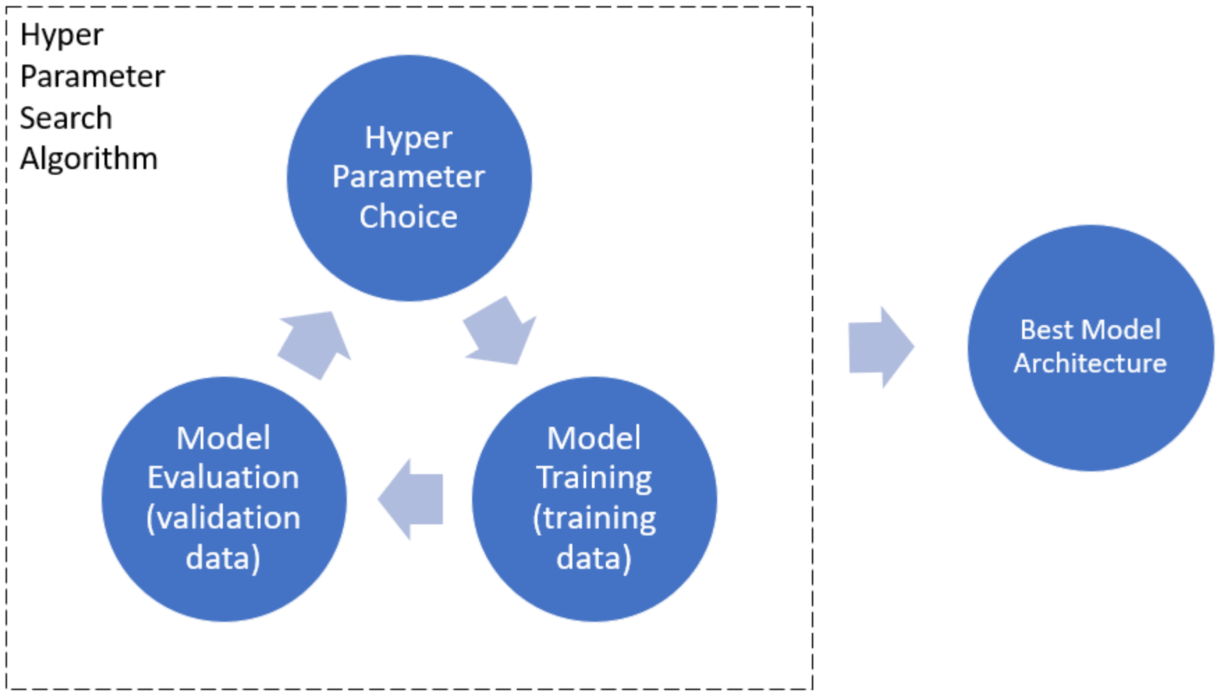

Hyper Parameter Search Algorithm
Hyper Parameter Choice
Model Training (training data)
Model Evaluation (validation data)
Best Model Architecture

Figure 3.8: KerasTuner optimization algorithm

Table 3.1: FNS Genetic Algorithm File Summary

File Name	Description
CNN_Handler.py	Python class for building CNN surrogate models
Genetic_Algorithm.py	Python class for running genetic algorithm
Individual_v1.py	Python class for individual
MCNP_File_Handler.py	Python class for MCNP
Run_CNN_GA.py	CNN surrogate genetic algorithm options file
Run_GA.py	Genetic algorithm options file
int_exp_model_reinit_0th.hdf5	Insertion experiment k-eff surrogate
keff_model_reinit_0th.hdf5	k-eff surrogate
representativity_model_reinit_0th.hdf5	Representativity surrogate
tf_model_reinit_0th.hdf5	Total flux surrogate
3d_FNS_integral_exp_template_adj_cas_2A_keff.txt	MCNP Insertion experiment k-eff template for 1-30 plates in cassette A
3d_FNS_integral_exp_template adj cas 2A keff 0 cassette A.txt	MCNP Insertion Experiment k-eff template for 0 plates in cassette A
3d_FNS_template_adj_cas_2A_keff.txt	MCNP k-eff template for 1-30 plates in cassette A
3d_FNS_template_adj_cas_2A_keff_0_cassette_A.txt	MCNP k-eff template for 0 plates in cassette A
3d_FNS_template_adj_cas_2A_source.txt	MCNP source calculation for 1-30 plates in cassette A
3d_FNS_template_adj_cas_2A_source_0_cassette_A.txt	MCNP source calculation for 0 plates in cassette A

The text files are templates for the various types of MCNP calculations used during the optimization. One set of template files cover FNS geometries with between 1-35 plates in cassette pattern 2A. In the other set, with "0_cassette" in the file titles, are used when there are no plates in cassette 2A. This is done because several changes are needed to accommodate having 0 plates in cassette A, including the removing of the aluminum cassette itself.

Chapter 4

Genetic Algorithm Optimization Results

This chapter discusses the building of the CNN-based surrogate models and includes a comparison of a genetic algorithm with and without surrogate models. In addition, a comparison of TSUNAMI-calculated sensitivities to MCNP and surrogate-model calculated delta k-eff values are presented. All Python codes used in this analysis can be found here: https://github.com/jpevey/Dissertation-Work.

4.1 Geometry

All work in this chapter is done using the MCNP 3x3 model described in the Section 3.4. In summary, the model used for the MCNP optimization includes three zones of 25 cassettes separated into up to six unique patterns of 20 plates. During the optimization the center cassette (pattern A) of zones 1 and 2 is combined into a single cassette that has a variable number of plates. The length and composition of the center cassette is a free parameter that can be changed during optimization. The location of the experimental zone moves along with the length of cassette 2A. The total number of cassette patterns is 7 with a maximum of 150 plate locations and a minimum of 120. In all cases, three plate materials are available, sodium metal, 9.75% enriched uranium and polyethylene plates. A fourth plate type, void, is also used to account for the voids in the FNS in both the experimental volume and the neutron generator.

4.2 Initial NSGA-II Optimization of FNS

In this section an NSGA-II genetic algorithm is run to optimize the FNS. This calculation is used to compare to the CNN-surrogate model based optimization discussed in the next section.

4.2.1 Optimization Options

Initially an NSGA-II genetic algorithm is run using MCNP to solve for the three objective functions. This genetic algorithm is implemented with the parameters found in Table 4.1. This algorithm used standard crossover and mutation routines which combined parents on a plate-by-plate basis where a unique child would be produced by cycling through all possible plate locations (up to 150 in this configuration) and randomly selecting either the plate from parent 1 or parent 2 with equal probability. Mutation is done by a fixed probability that any children would be subject to a 10% chance of a plate to change from the current plate material to another.

In addition, a linearly increasing constraint on representativity is enforced on the parent population. This constraint has no effect at generation 0, but increased linearly until at generation 50, individuals with a representativity above 0.95 were preferred. This constraint is enforced such that if in any generation there were not 20 parents which meet the representativity constraint, that the constraint is reduced until there are at least 20 eligible parents. In addition, all previously evaluated individuals are included in the Pareto front sorting calculation. This is done so that individuals evaluated while the representativity constraint is less restrictive and may have been dropped from the Pareto front due to sub-optimal evaluations of the other objective functions would be included in later generations.

Table 4.1: Initial FNS Genetic Algorithm Hyper-parameters, Sodium Metal Optimization

Description	Hyper-parameter
Stopping Criteria	Generation Count (100)
Parent Population	20
Child Population	80
Crossover Rate	50%
Mutation Rate	10% per plate
Mutation Type	Single plate material change
Initial Population	100 (randomly created)
Objective/Constraint Solver	MCNP
Representativity Constraint	0.95 at Generation 50

This type of constraint allows the algorithm to potentially explore more of the design space before the design space is reduced by the constraint and to leverage already explored areas of the design space. The additional computational expense of sorting more individuals is less than the potential loss due to omitting previously evaluated patterns.

4.2.2 Optimization Results

A genetic algorithm using the options described in Section 4.2.1 was run on the University of Tennessee Nuclear Engineering cluster. This cluster is a Beowolf [47] cluster which is comprised of hundreds of CPUs. These CPUs vary in speed and is widely used for research by others, but, in general, a single generation requires approximately 10 minutes for all k-eff calculations and 30-35 minutes for all of the source-driven calculations needed for both the objective and constraint evaluations. In total this calculation required approximately 3 days of wall time to complete 100 generations.

Figure 4.1 shows the average objective function evaluation and k-eff for all individuals (parents and children) in each generation. As can be seen in this figure, the optimization objectives initially plateau before the representativity constraint is fully enforced at generation 50. After generation 50, the objectives plateau at new levels. The insertion experiment k-eff, which before generation 50 results in a negative reactivity insertion has a positive effect after generation 50. This is due to changes of the spectra caused by the representativity constraint. Sodium metal has a non-trivial neutron absorption cross section at lower neutron energies, seen in Figure 4.2. When representativity is low and the spectrum is more thermal, sodium metal can act as a weak neutron poison rather than as a reflector when inserted into the experimental volume. As representativity rises and the spectra shifts to be more fast and away from the relatively large absorption cross section at lower neutron energy levels, the sodium acts as more of a reflector than as a poison.

The change in the total flux and representativity of the parent population during the optimization can be seen in Figure 4.3. Between generation 3, the first generation where there are 20 individuals in the parent population which meet the k-eff constraint, and generation

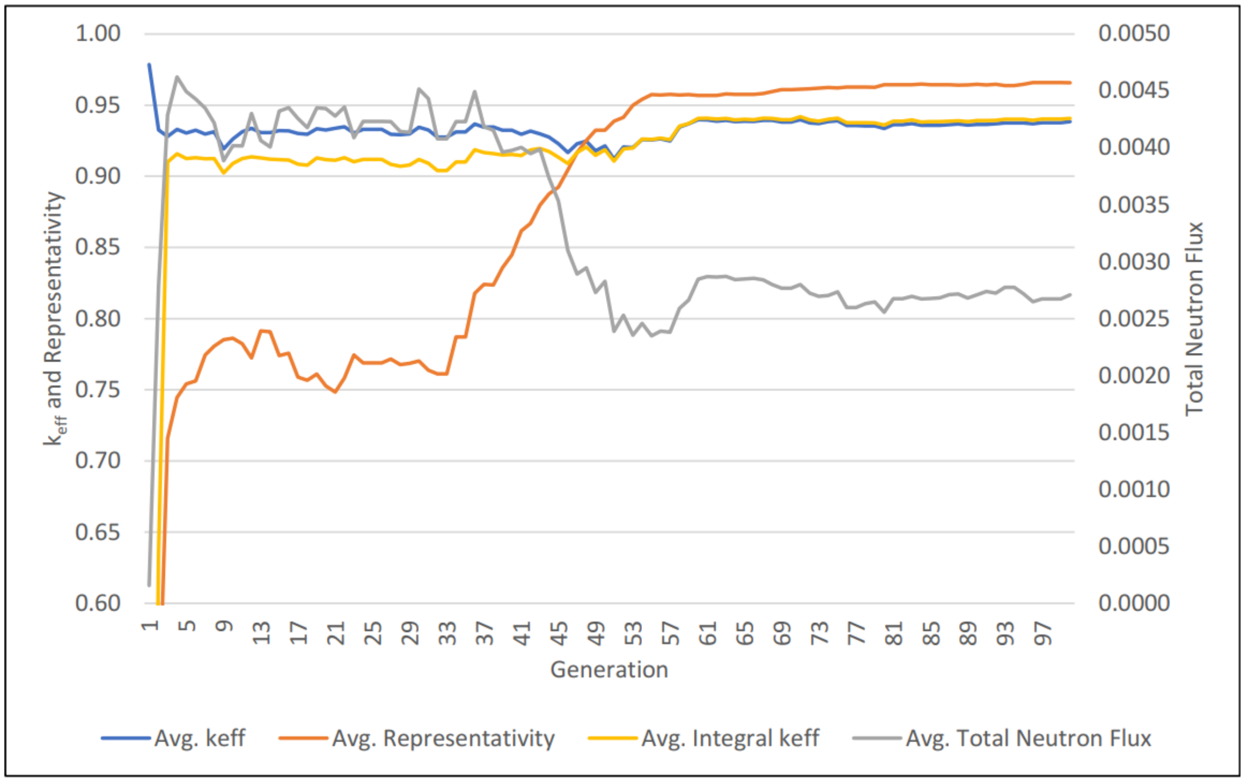


Figure 4.1: Initial Genetic Algorithm, Average Parent Objective Function vs. Generation

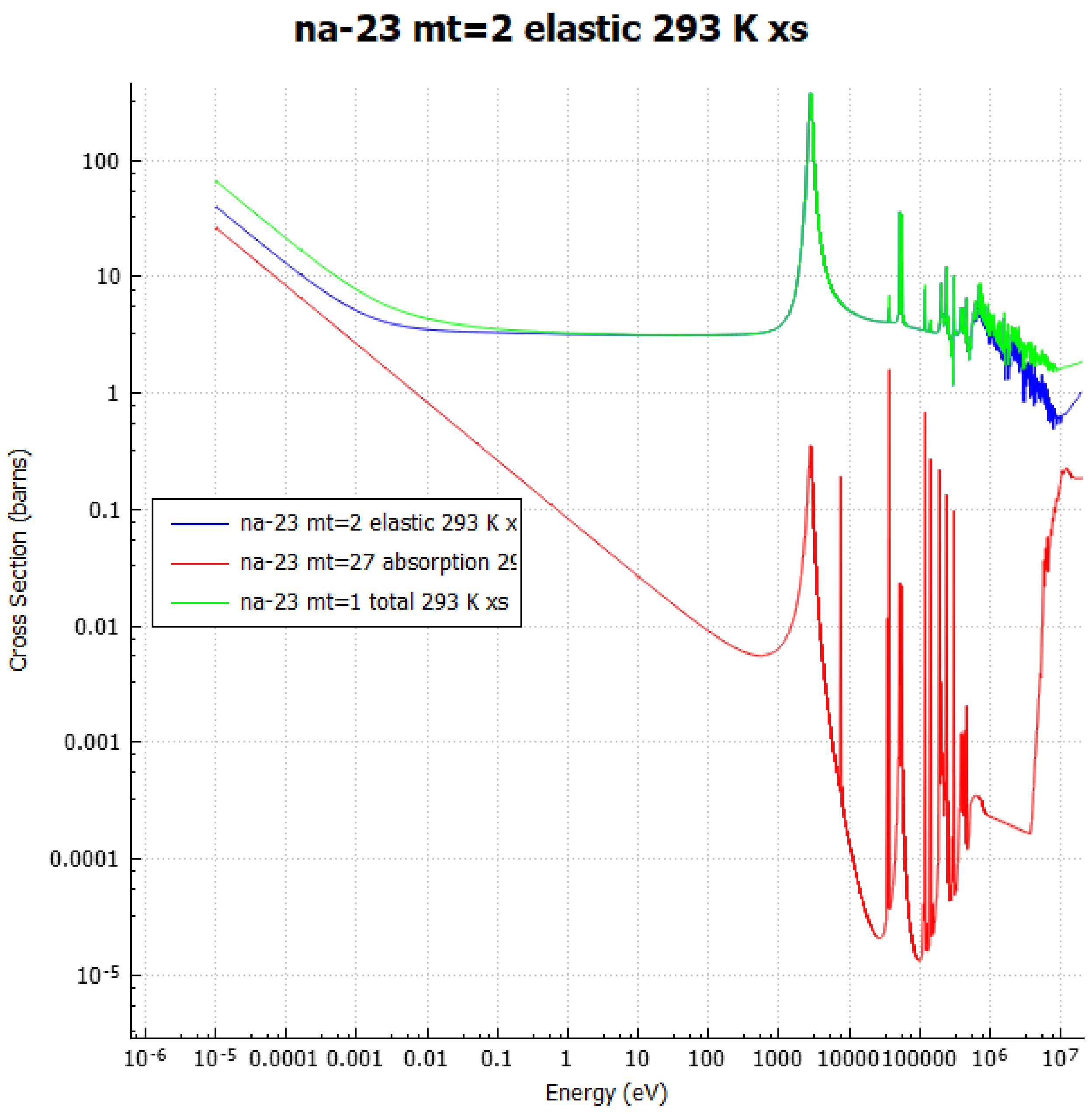


Figure 4.2: Selected Na-23 Cross Sections, ENDF 7.1

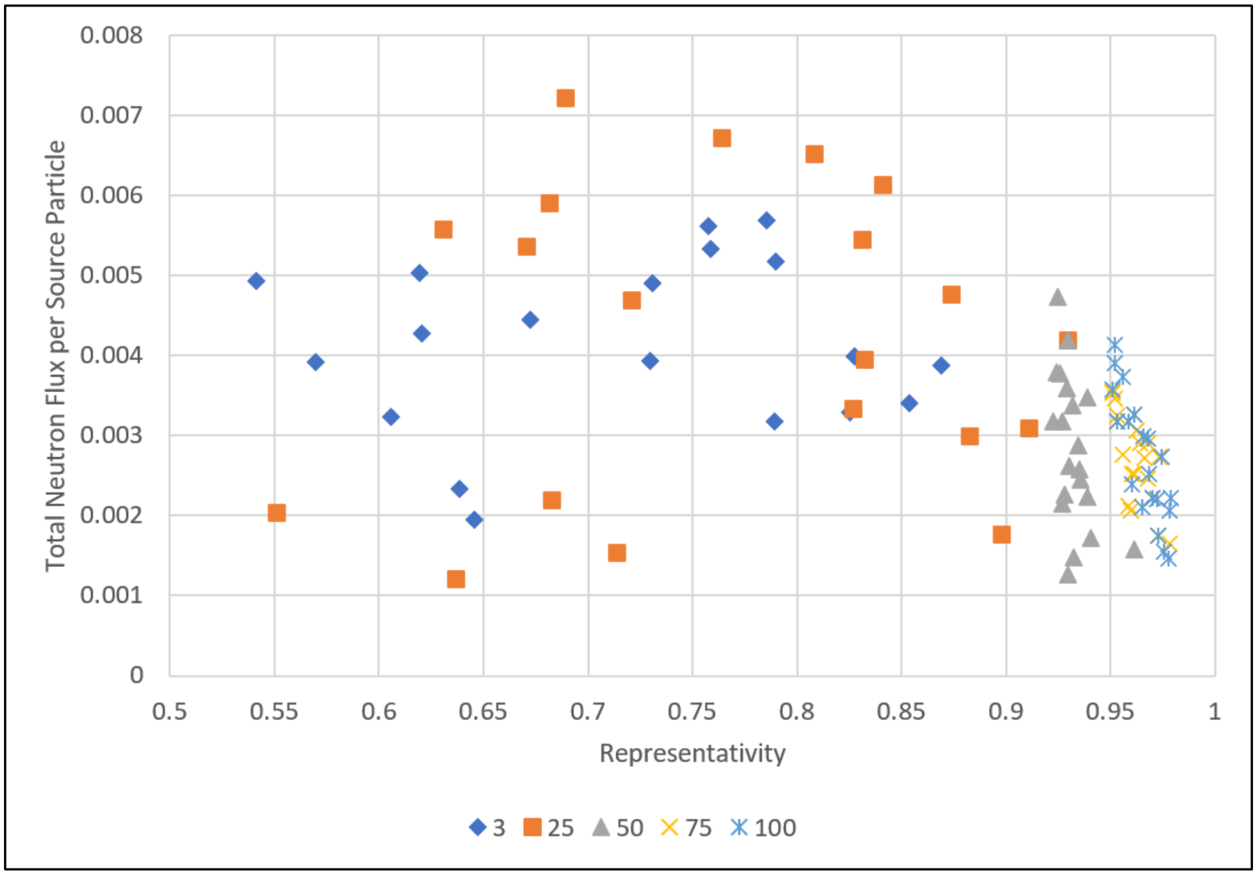


Figure 4.3: Initial Genetic Algorithm, Total Neutron Flux

25, the maximum representativity and total flux increase. At generation 50 the maximum total neutron flux decreases and the minimum representativity increases. This is due to the representativity constraint taking hold.

After generation 50, when the representativity constraint is fully enforced, if there are not 20 individuals which have a representativity above 0.95 then the individuals with the top largest representativity values are chosen for crossover. This puts pressure on the population to meet the representativity constraint and by generation 54 it is met. After generation 54 some progress is made in increasing total neutron flux and representativity. By generation 100 the progress has plateaued.

The average FNS pattern composition in the Pareto front also changes through the optimization, as seen in Figure 4.4. As the representativity constraint is enforced near generation 37, fuel plates begin to replace polyethylene. This is to be expected as polyethylene is a strong neutron moderator which will generally slow neutrons. Because the target neutron spectrum is of a fast reactor, having too much polyethylene near to the experimental volume reduces the representativity of the neutron spectrum. At approximately generation 55, fuel plates begin replacing sodium metal. This change corresponds to an increase in the average k-eff in these generations. Like the objective and constraint functions, after generation 85 the average plate counts remain relatively consistent until the end of the optimization.

In summary, this genetic algorithm optimization of the FNS produced a viable set of FNS patterns according to the constraints and objectives of the optimization. Table 4.2 shows the minimum, maximum and average objective and constraint evaluations for the final 20 parents in this optimization. Within this Pareto front of patterns there are examples with high and low representativity, high and low total neutron flux and both positive and negative examples of reactivity insertion due to the insertion of sodium into the experimental volume.

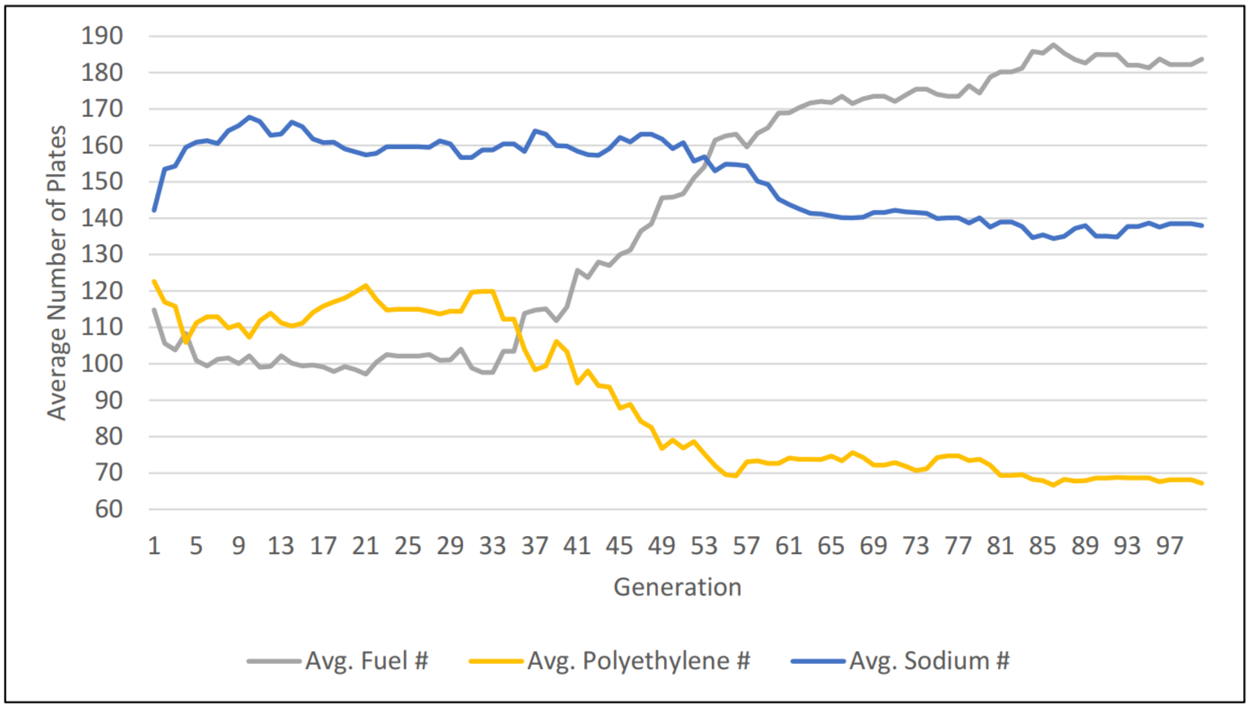


Figure 4.4: Initial Genetic Algorithm, Plate Count vs. Generation

Table 4.2: Summary of Sodium Metal Standard Genetic Algorithm Optimization Objectives, Final Generation

	k-eff	Insertion Experiment delta k-eff	Representativity	Total Flux
Max	0.94987	0.00592	0.97893	0.00413
Min	0.91207	-0.00195	0.95095	0.00146
Average	0.93829	0.00224	0.96557	0.00271

4.3 CNN-Surrogate Accelerated NSGA-II

In this section the surrogate models for the total neutron flux, representativity, insertion experiment k-eff, and k-eff are discussed.

4.3.1 Building a CNN-Based Surrogate Model of the FNS

The optimal selection of hyper-parameters of a neural network can be a difficult task. This process can be accomplished in several ways including using expert judgment, a random search or applying one of many search algorithms [48]. The optimization of hyper-parameters for the CNN-based surrogate models in this analysis is done by a Bayesian search option using the KerasTuner Python library. This type of search can be a powerful method of global optimization and is often used in machine learning applications [49].

Practically, using the KerasTuner Python library involves describing a given network as a HyperModel object which takes as input the various hyper-parameters associated with a network and returns the network. The network is then trained for a relatively small number of epochs. Reducing the number of epochs increases the number of architecture configurations that can be tested, but below a threshold (that is found by trial and error) produces networks which cannot be effectively trained in the next stages.

A Bayesian optimization [50] of CNN neural network architectures was performed for the FNS objectives and constraint model architectures. The range of potential values for each parameter that describes the architecture can be found in Table 4.3 and Figure 4.5 is a graphical representation of how these parameters fit into the network architecture. A total of 300 network architectures were evaluated with 30 initial randomly created architectures and 270 subsequent architectures which are selected by the Bayesian update rules.

An NVIDIA GTX 1080 graphical processing unit (GPU) is used for this training. When compared to a computational processing unit (CPU), a GPU can greatly increase training times of neural networks. This is due to the forward and backwards pass, or the training, of the network being a matrix operation that is able to be executed in parallel. The number of

Table 4.3: CNN Bayesian Optimization Parameters

Hyper-parameter	Value Range	Step Size
First Convolutional Layer Width	16-64	1
Kernel Size	3-4	1
Dense Layer Width	32-256	32
Max Pool Size	1-10	1
Number of Hidden Convolutional Layers	3-7	1
Dropout Percentage	0-50%	10%
Learning Rate	1E-7 - 5E-2	1E-5

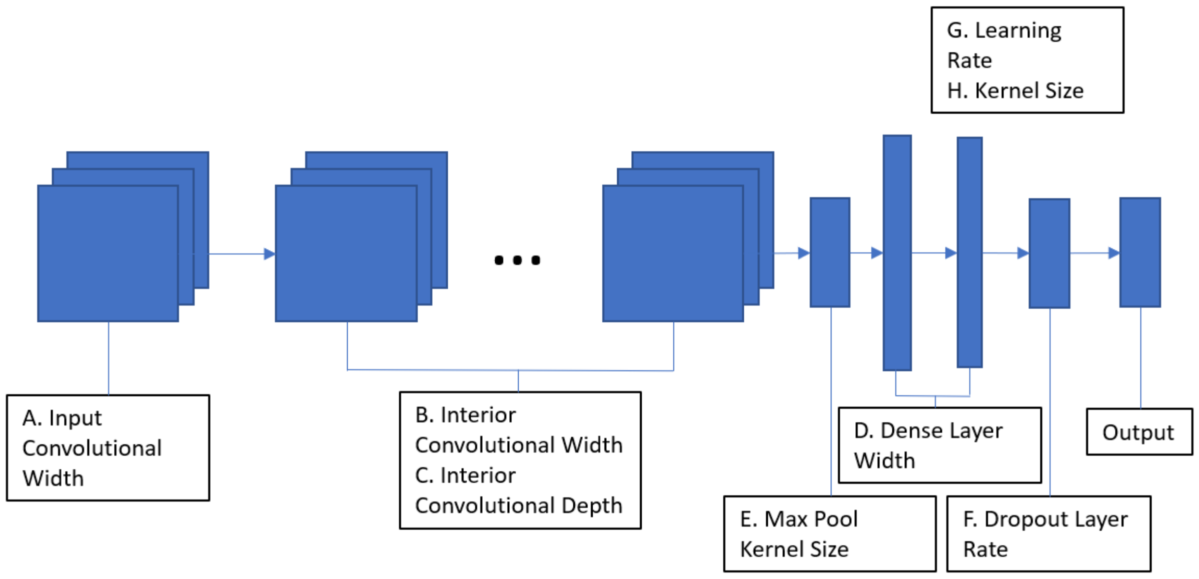


Figure 4.5: CNN Hyper-parameter Tuning Parameters

training examples trained in parallel is called the batch number. In this case a batch number of half of the available training data was used. This means that half of the training data was passed through the network and then the trainable parameters in the network were updated with respect to those training examples together. The amount that the network trainable parameters are updated is called the learning rate. For the training of these networks, the complete set of training data was fed into the network 100 times. A single cycle of training a network on all available training data is called a epoch. This training was done two times with different random number seeds. The training and testing data sets were produced by randomly assigning all individuals into a training data set comprising approximately 90% and with approximately 10% of individuals placed into the testing training set.

The Bayesian search of the network architectures using KerasTuner library required approximately one day of computational time per objective/constraint function. The top ten networks which had the lowest error on the testing data were selected for further training. Figure 4.6 shows the training and testing loss for both mean squared-error and mean average loss for these top architectures. The total flux, representativity and k-eff architectures show classical over-fitting of the data in that after a certain amount of training that the error of the testing data no longer decreases while the error on the training data does. At this inflection point the networks are no longer learning features which are any more useful for predicting the testing data but are learning features useful for predicting the training data. They are, in essence, memorizing the training data. Because the final architecture for each surrogate model is selected by which minimizes the testing data loss and not the training data loss, the training of the networks beyond the point where the testing loss is decreasing represents wasted computational resources. Future hyper-parameter searches could include a stopping criteria to reduce this waste.

The architecture which was best able to minimize the loss on the testing data were then selected. These final network architectures can be found in Table 4.4. The TensorFlow model objects are saved locally and used in the subsequent genetic algorithm optimizations.

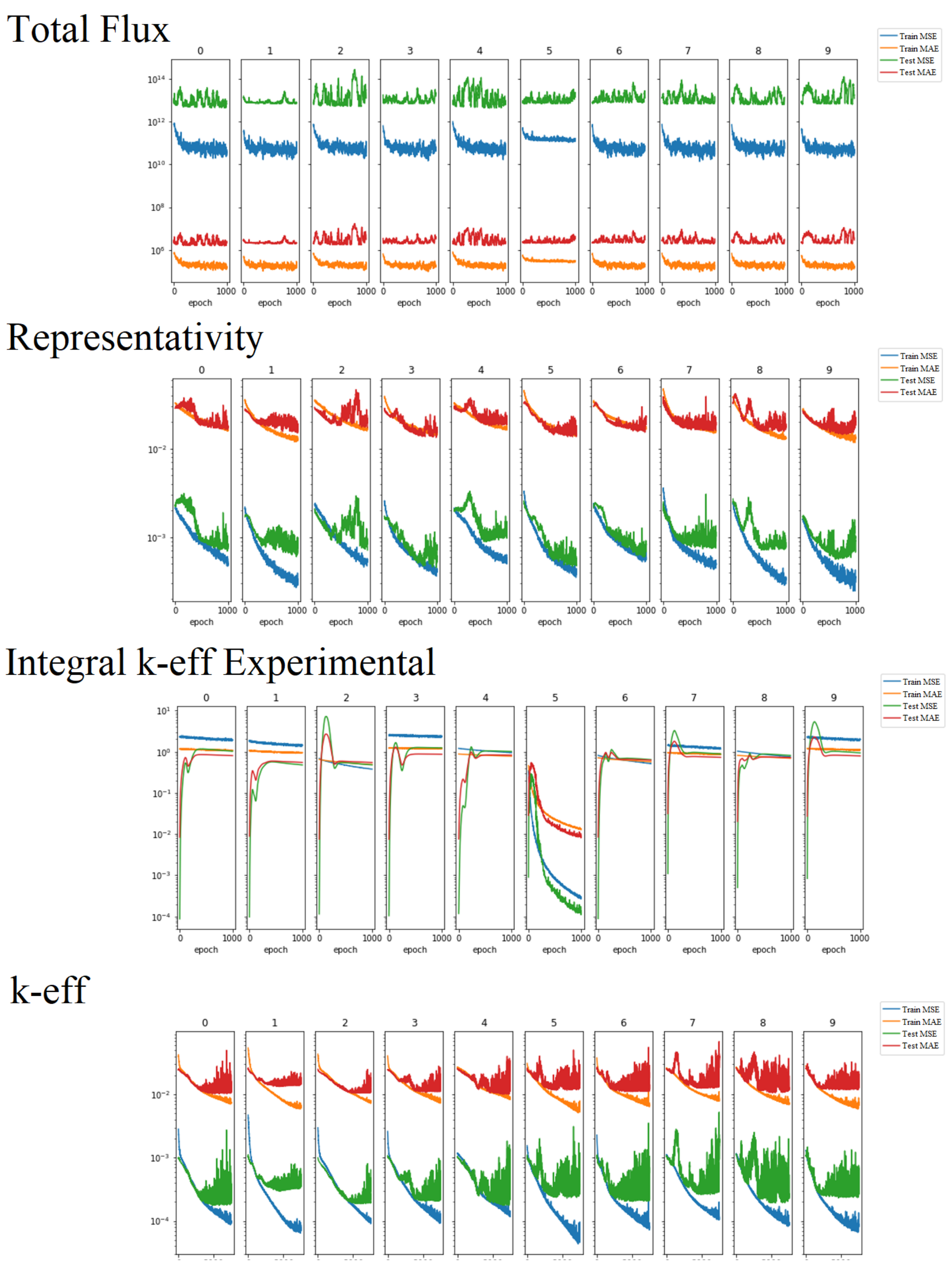


Figure 4.6: Training Loss of Top 10 Network Architectures for Each Surrogate Model

Table 4.4: Final CNN Hyper-parameters

Hyper-parameter	k-eff	Total Flux	Represen-tativity	Insertion Experiment k-eff
1st Conv. Width	32	40	56	16
2nd Conv. Width	64	64	16	32
3rd Conv. Width	64	40	48	40
4th Conv. Width	40	24	16	24
5th Conv. Width	N/A	N/A	N/A	40
6th Conv. Width	N/A	N/A	N/A	16
Kernel Size	4	4	3	3
Dense Layer Width	32	256	32	96
Max Pool Size	9	4	10	7
# of Hidden Conv. Layers	3	3	3	5
Dropout Percentage	0.5	0.2	0.5	0.4
Learning Rate	0.0208839	0.0160466	0.0317394	0.0018139

4.3.2 FNS Genetic Algorithm with Surrogate Model Acceleration

A NSGA-II optimization of the FNS was done using the CNN-based surrogate models built as described in Section 4.3.1. In this genetic algorithm optimization, the CNN network architectures which showed the best ability to learn to predict the fast flux, representativity, insertion experiment k-eff and k-eff are trained on data produced during the optimization. The purpose of incorporating the surrogate models into the optimization to both produce a better suite of individuals that push out the Pareto front and more effectively use computational resources by not evaluating FNS patterns which will not meet the constraint criteria.

In this calculation, the CNN-based surrogate models were trained on data produced by the genetic algorithm and used in place of MCNP to predict the objective functions and k-eff constraint. These are referred to as the interior and exterior optimizations. The interior optimization uses CNN-based surrogate models to evaluate the objective functions and k-eff constraint. This interior genetic algorithm uses the same genetic algorithm options as the non-surrogate model genetic algorithm, with the exception of increasing the number of children to 1000, the number of parents to 80, and the total number of generations evaluated is reduced to 10. The final 80 parents after these 10 generations, representing the final Pareto front, are passed to the outer optimization. Here the objective functions and constraint are evaluated with MCNP.

These 80 new individuals are added to both the training data sets and the master list of all individuals evaluated. A new Pareto front is produced from the master list and passed to the interior optimization to be used as the initial population of individuals. This sorting can be computationally expensive but is less so than the potential sub-optimal selection of parents for the next generation.

The interior genetic algorithm uses a generation count of 10. In each generation 1000 unique children are produced and 80 parents are selected from them for the next generation. After 10 generations, the top 80 individuals representing the Pareto front are evaluated with MCNP for both k-eff and the objective functions. These 80 individuals are then added to a master list of all individuals evaluated and then split into training and testing data sets using an 80/20 split. These data sets were then used to train the surrogate models where the network with the lowest error on the testing data set would be selected as as the surrogate model for the next generation. The genetic algorithm is then restarted with the newly trained surrogate models as the solvers and the top 80 patterns as the initial parent population. This optimization algorithm is presented in Figure 4.7. In this figure each step of the surrogate-based genetic algorithm is summarized, with numbers indicating the order in which operations are performed and arrows indicating the passing of data from one part of the algorithm to another.

Figure 4.8 presents the average parent objective and constraint functions as evaluated by MCNP. Many of the features found in the previous optimization are present in this optimization including the representativity constraint's effect on both representativity, total flux and on the sign of the insertion experiment delta k-eff.

Figure 4.9 shows the predicted versus the true values for the objective and constraint functions at the final generation. The k-eff, representativity and total neutron flux surrogate models' predictions compare well with the MCNP values as measured by the coefficient of determination, R^2. This value is a measure of how well a linear fit of the data is to the data. A perfect surrogate model would have an R^2 of 1.0. The R^2 values for three of the objective and constraint functions are in excess of 0.90.

Three of the surrogates reproduce the linear trend expected and show an ability to differentiate between the lowest and highest examples for each objective and constraint. All R^2 values, except for the insertion experiment delta k-eff are above 0.9, indicating that the models predict the values well.

The insertion experiment delta k-eff is not predictive of the MCNP calculated values. There may be many reasons for this poor fit including the combined uncertainty due to the combination two stochastically calculated k-eff values. Calculating the insertion and standard FNS k-eff to more precision may help resolve this. In addition, the process of finding the hyper-parameters for this network is also suspicious with the training loss on the testing

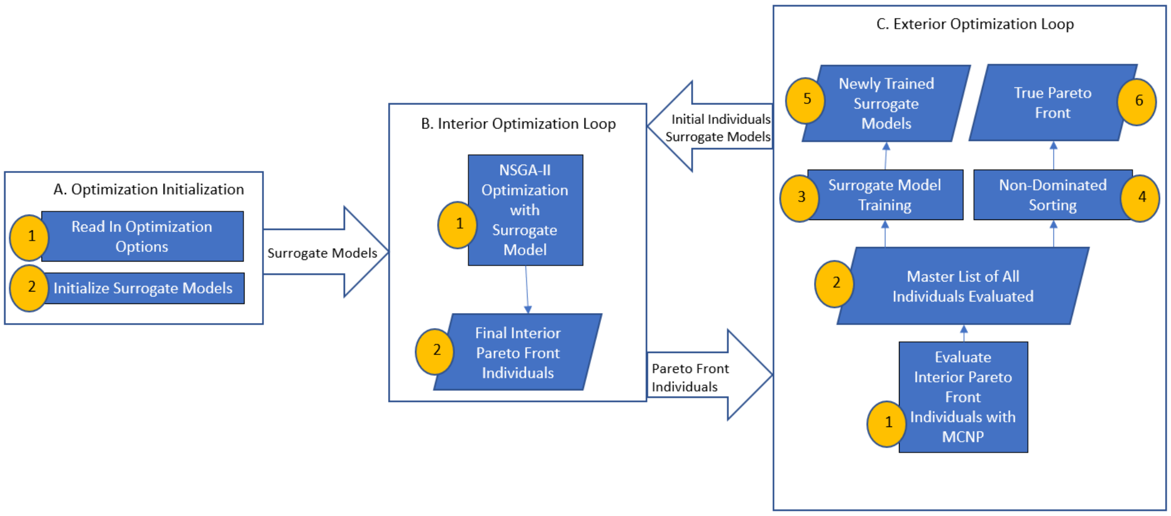

A. Optimization Initialization
1
Read In Optimization Options
2
Initialize Surrogate Models
Surrogate Models
B. Interior Optimization Loop
1
NSGA-II Optimization with Surrogate Model
2
Final Interior Pareto Front Individuals
Initial Individuals
Surrogate Models
Pareto Front Individuals
C. Exterior Optimization Loop
5
Newly Trained Surrogate Models
True Pareto Front
6
3
Surrogate Model Training
Non-Dominated Sorting
4
2
Master List of All Individuals Evaluated
Evaluate Interior Pareto Front Individuals with MCNP
1

Figure 4.7: Overview of FNS Surrogate-based GA Algorithm

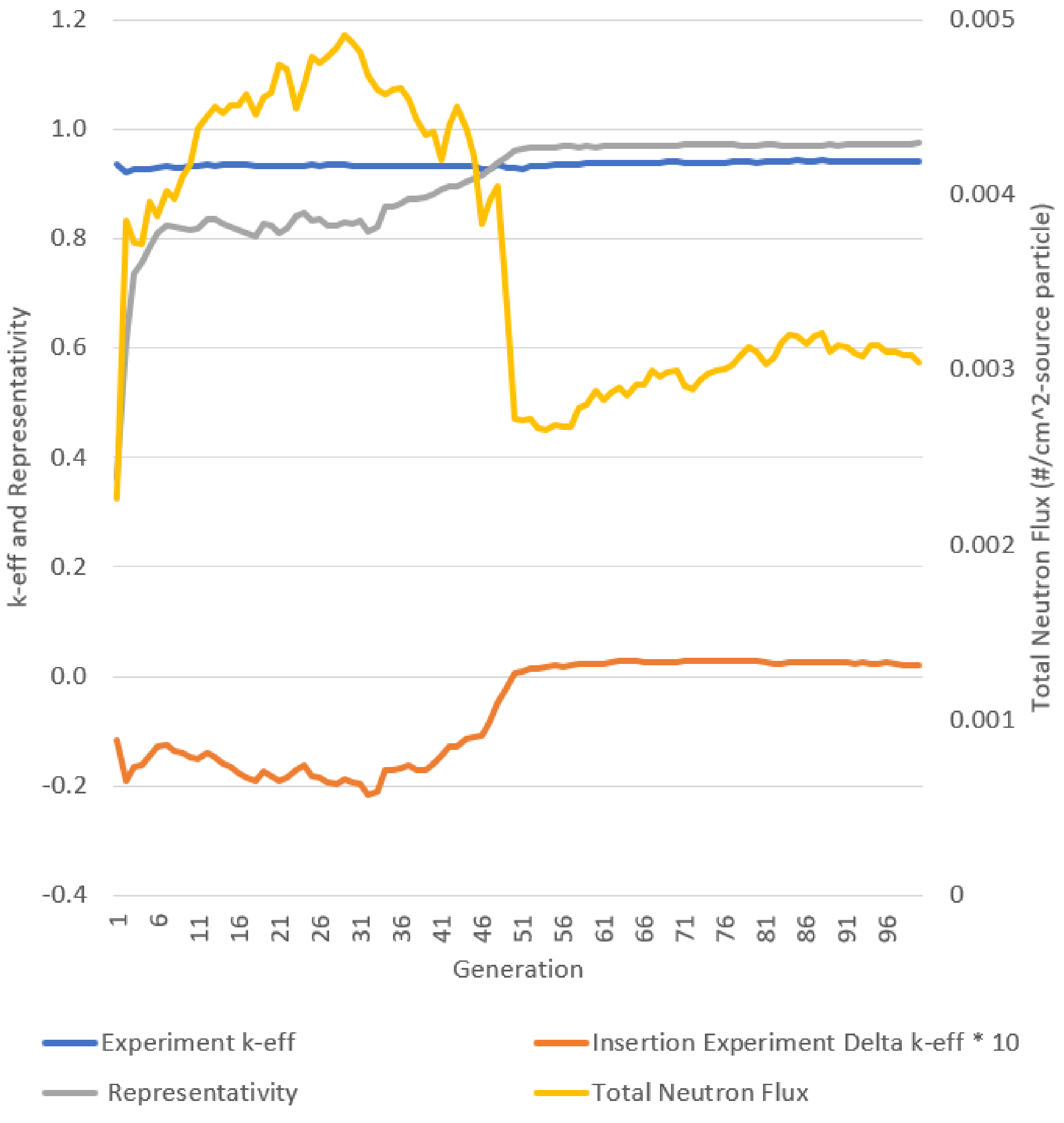


Figure 4.8: Accelerated Genetic Algorithm, Average Parent Objective Function vs. Generation

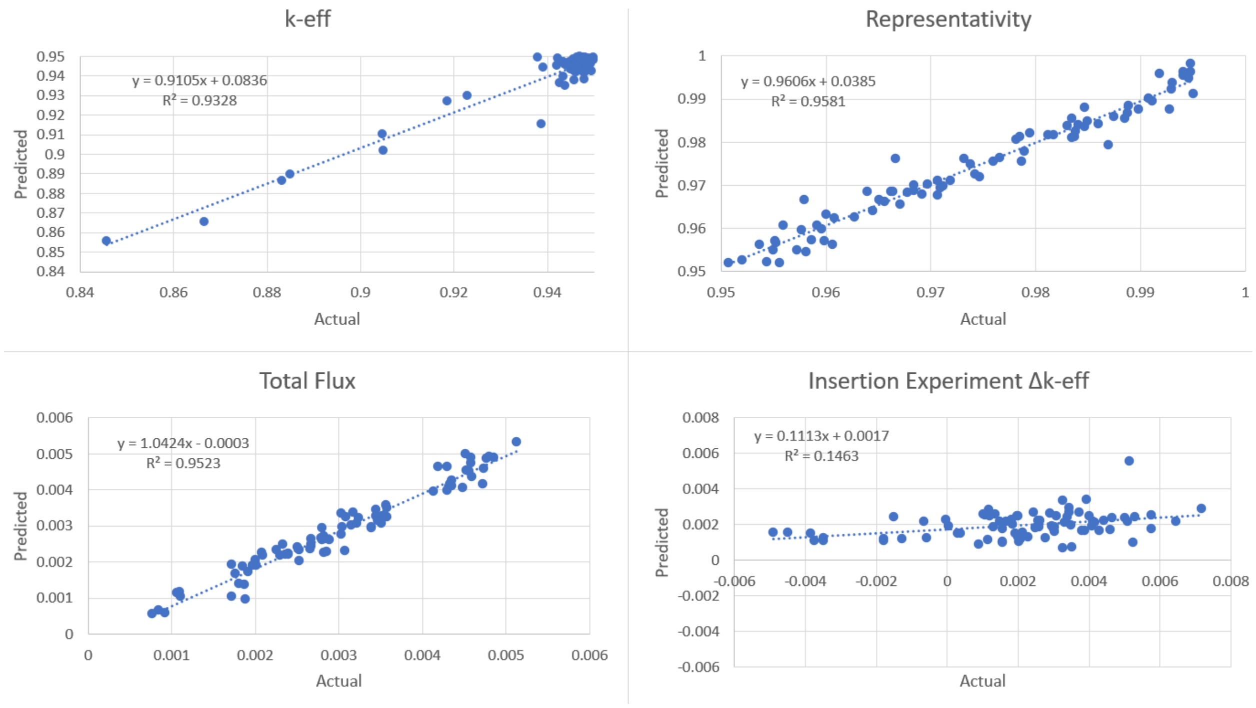


Figure 4.9: Generation 100 Surrogate Model Prediction vs. Actual

data being lower than the training data. This points to poorly selected training/testing data. Evaluating more network architectures and expanding the range of hyper-parameters may produce a more adequate network architecture. This objective function is the absolute value of the delta k-eff due to the insertion of the material. By taking the absolute value the optimization is maximizing the delta regardless of whether the experiment results in a positive or a negative insertion of reactivity.

The results of this work show that the CNN-based surrogate models are able to be trained to predict the objective and constraint functions given sufficient training data. The architectures found produced networks which were able to be trained to predict three of the four objective and constraint functions.

4.3.3 Comparison of Benchmark and Surrogate Model Genetic Algorithms

To begin we will examine the output of the algorithm and compare it to the standard NSGA-II algorithm. Both calculations were run for 100 generations with 80 individual patterns evaluated in each generation. In total there were 8,100 unique individuals evaluated in each calculation. The in-line optimization using the CNN surrogates produced a final Pareto front of individuals which outperform the standard NSGA-II genetic algorithm in all objectives. Figures 4.10 and 4.11 show the final parents of both the standard NSGA-II calculation and the NSGA-II calculation using CNN surrogate models. Compared to the standard NSGA-II calculation, the accelerated calculation produced a suite of individuals which increased the maximum representativity from 0.979 to 0.995, increased the maximum total flux from 0.0041 to 0.0051 and increased both the maximum and minimum insertion experiment delta k-eff (insertion experiment k-eff - experiment k-eff) from 0.0059 to 0.0072 and -0.0020 to -0.0049, respectively.

Only one individual in the final Pareto front of the accelerated calculation is fully dominated by the parents of the standard calculation. The individuals with the best objective evaluations in the standard calculation are all dominated by individuals in the accelerated

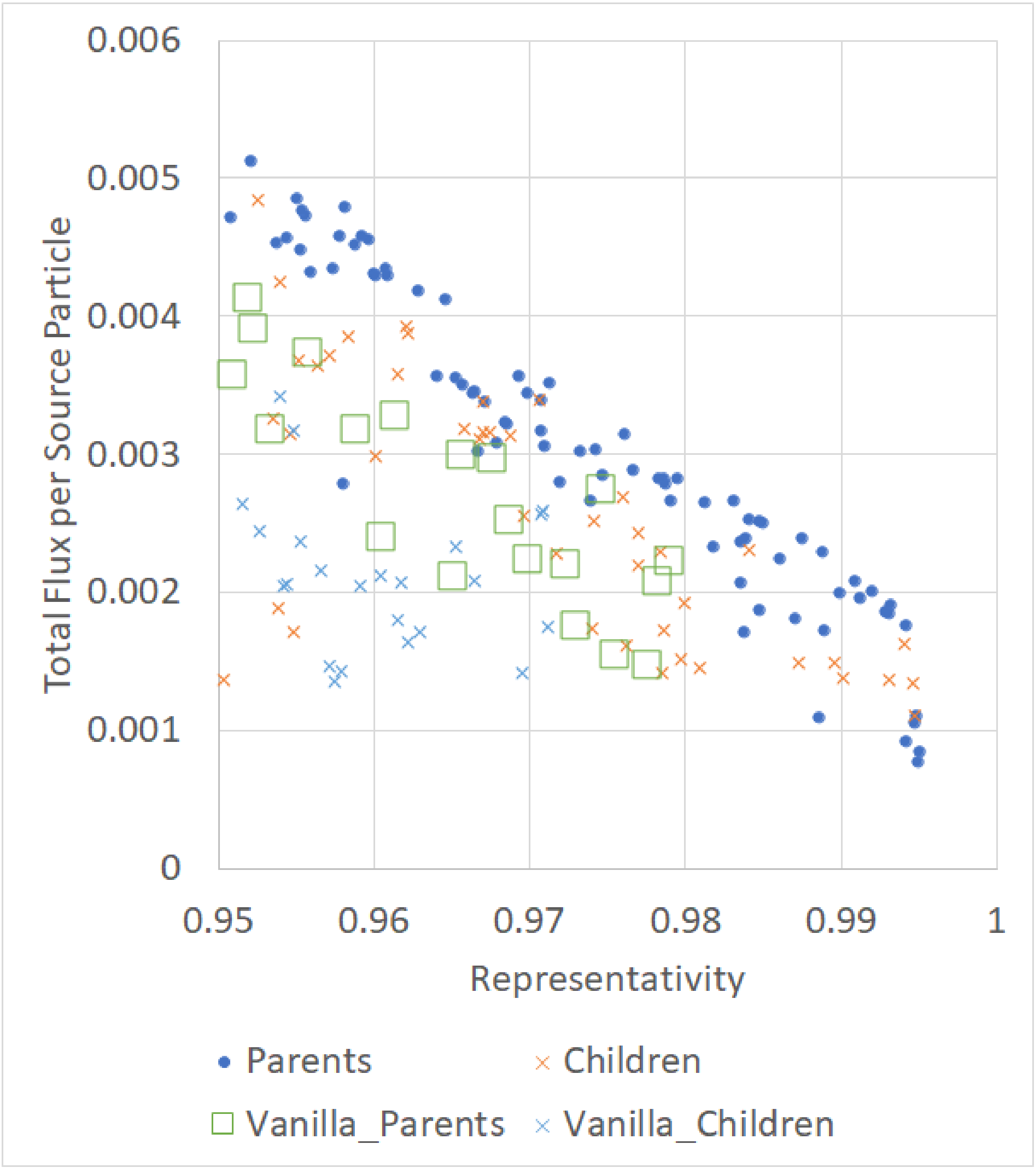


Figure 4.10: Total Flux vs. Representativity of Pareto front of Initial vs. Accelerated GA

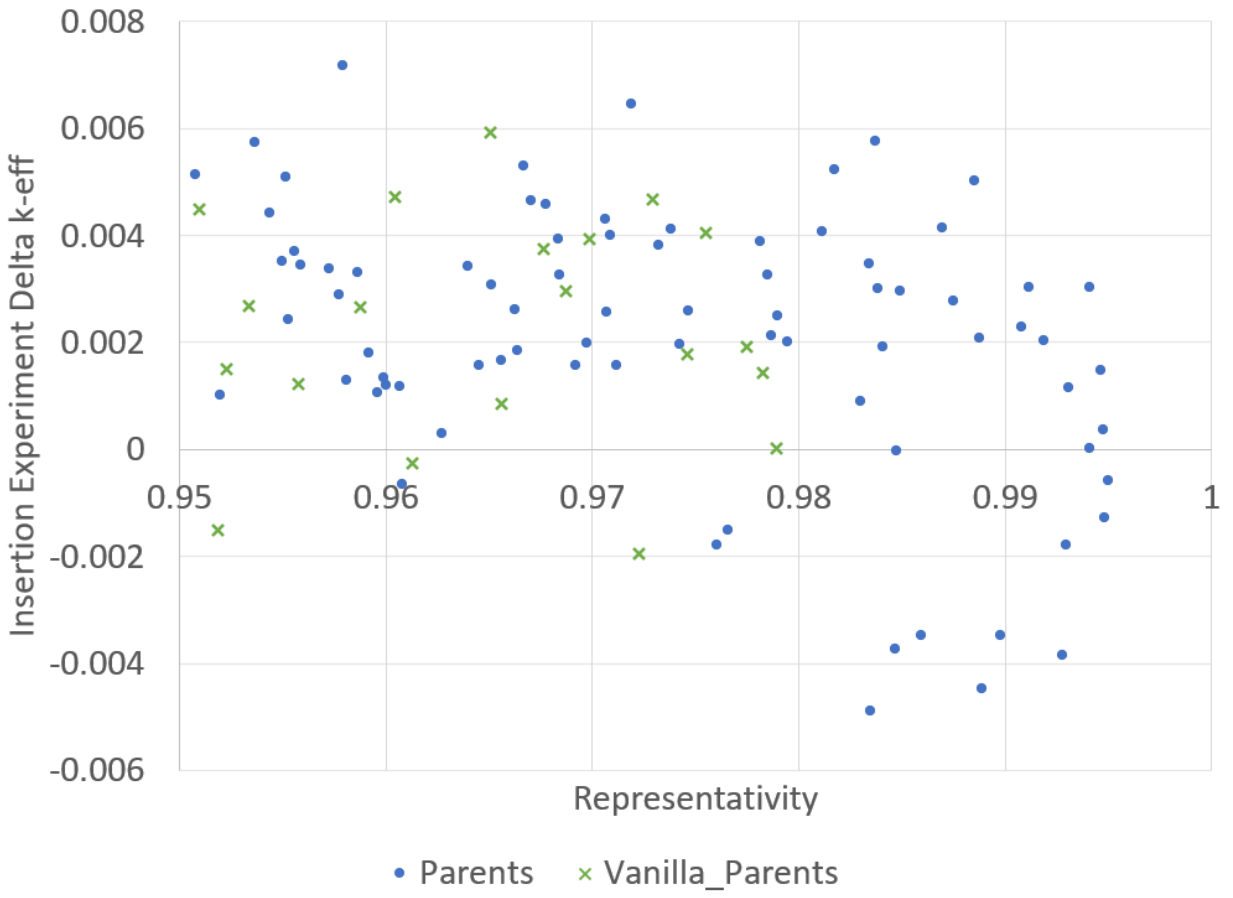


Figure 4.11: Total Flux vs. Representativity of Pareto front of Initial vs. Accelerated GA

calculation. The individuals in the standard calculation with the highest representativity, total flux and either most negative or most positive insertion experiment k-eff are dominated by 15, 12, 1 and 17 individuals in the accelerated calculation.

Even without an ineffective insertion experiment k-eff surrogate model, the surrogate-assisted genetic algorithm produced a set of FNS patterns which out-perform the standard genetic algorithm. This is due to the effectiveness of the other surrogate models.

4.4 Other Optimizations of the FNS

In this chapter other genetic algorithm based optimizations of the FNS are discussed. Unlike the previously discussed optimization, only the CNN-surrogate implementation of the genetic algorithm is used. These calculations were completed on the University of Tennessee Nuclear Engineering cluster.

4.4.1 Optimization of the Little Boy Neutron Spectra

This optimization uses the surrogate-based NSGA-II algorithm discussed in Chapter 4 to optimize the FNS with a target of the Little Boy atomic bomb neutron spectra [51]. This target spectra can be seen, along with a high and a low representativity examples from the final Pareto front in Figure 4.12. The plate materials used are beryllium metal, polyethylene, and 9.75% enriched uranium. The parameters used in the genetic algorithm can be found in Table 4.5. This optimization ran for a total of 100 generations.

In general, this was a poor optimization where the maximum representativity found was less than 0.85. This may be due to the target neutron spectra being from an atomic blast rather than one found in a reactor, leading to a less smooth neutron spectra that the apparently struggles to match. As seen in Figure 4.13, the average representativity plateaus at an average of 0.85 around generation 44. This is when the representativity constraint takes hold. After this generation the parent population is made by taking the

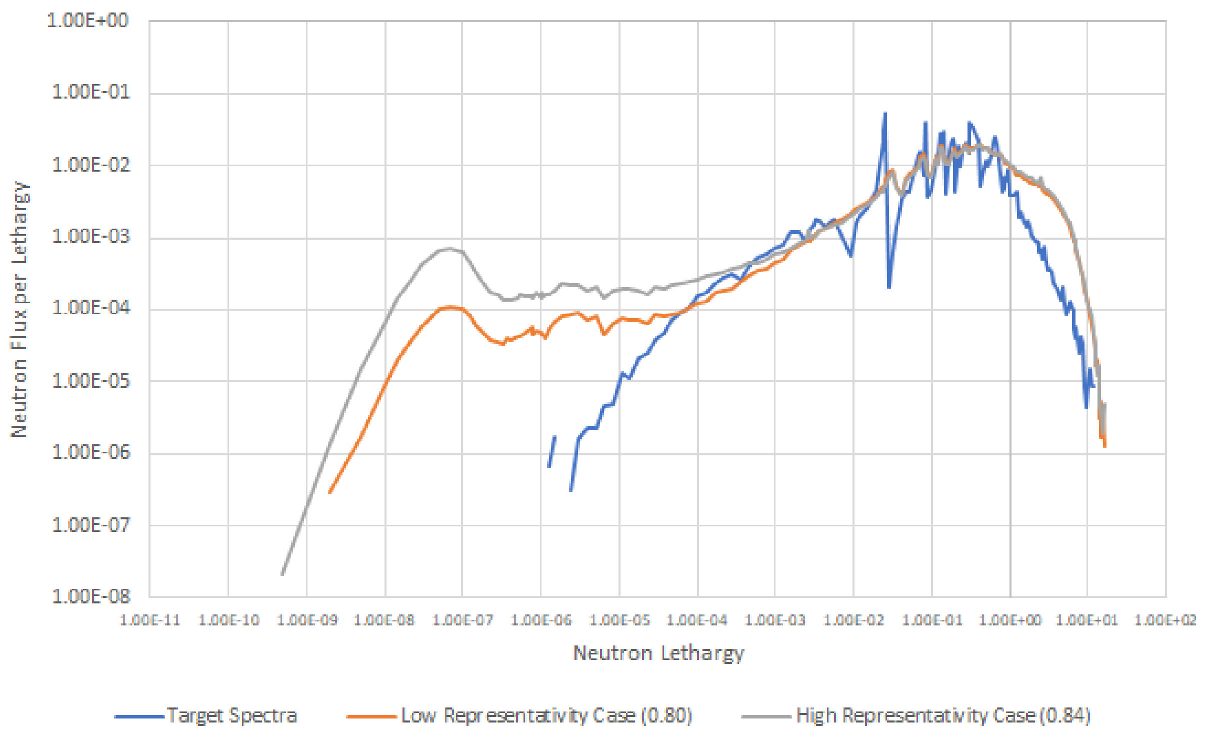


Figure 4.12: Neutron Spectra of Little Boy Spectra Optimization and the Highest and Lowest Representativity of the Final Pareto front, Normalized to an Equal Integral

Table 4.5: FNS Genetic Algorithm, Little Boy Neutron Spectra Optimization Hyper-parameters

Description	Hyper-parameter
Stopping Criteria	Generation Count (100)
Parent Population	20
Child Population	80
Crossover Rate	50%
Mutation Rate	10% per plate
Mutation Type	Single plate material change
Initial Population	100 (randomly created)
Objective/Constraint Solver	MCNP
Material Types	Polyethylene Beryllium Metal Enriched Uranium

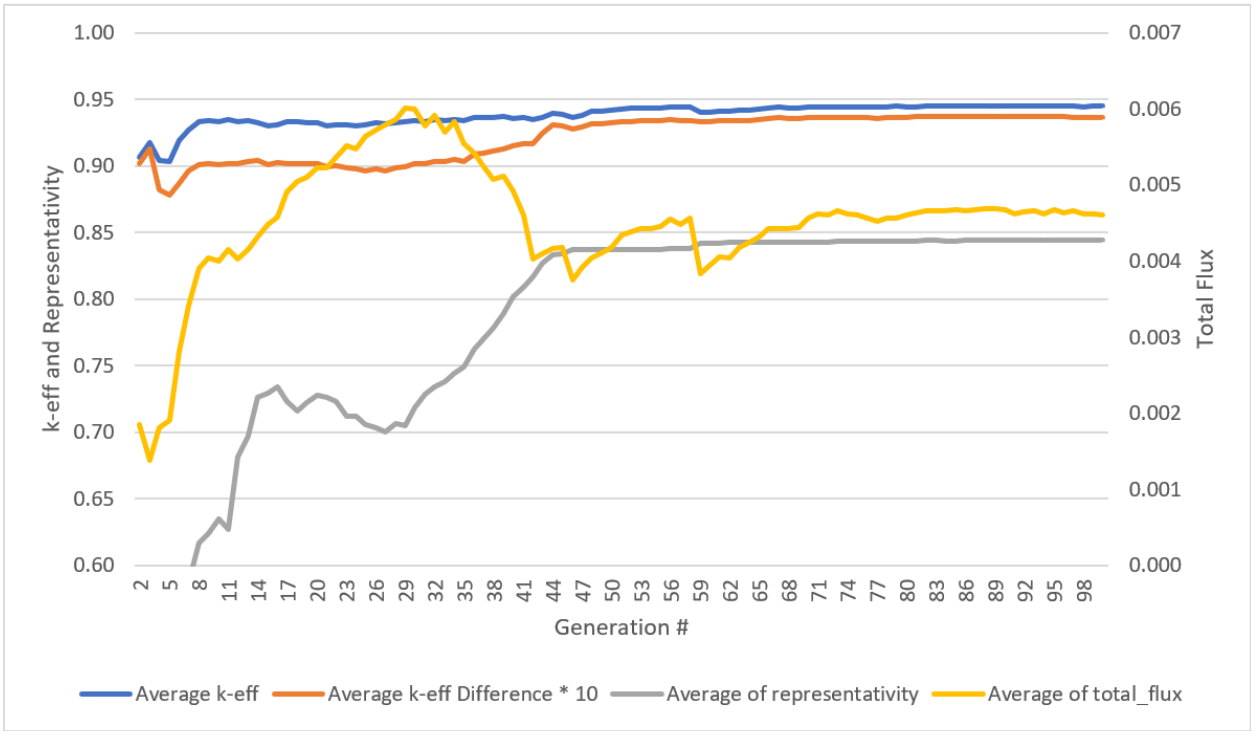


Figure 4.13: Genetic Algorithm Average Objective and Constraint Values for Little Boy Spectra Optimization

highest representativity cases regardless of any of the standard selection criteria such as non-dominated rank or crowding distance. Even with this pressure to increase representativity on the population, the optimization is stuck at this level with throughout the rest of the optimization.

Like the other optimizations, as the representativity constraint takes hold near generation 50, the total neutron flux in the experimental volume decreases and the total flux increases. This is expected behavior as the neutron spectrum shifts from soft to the harder target spectrum. The insertion experiment k-eff is shown in Figure 4.13 as well. It is multiplied by 10 for readability on this figure. It, and k-eff, plateaus as well at around generation 50.

Figure 4.14 shows the surrogate model prediction vs. actual values calculated with MCNP for all of the objectives and the constraints. From this plot the total flux and representativity show good agreement while the k-eff and insertion experiment k-eff show markedly bad agreement. This is because in generation 100 of the 80 individuals produced in the interior CNN-driven genetic algorithm, 48 patterns were already evaluated. When a pattern is already evaluated it is mutated. Of these 48 patterns that were mutated all of them had a k-eff above 0.95. The prediction of k-eff and insertion experiment k-eff are not updated, meaning that the CNN-predicted values and actual values are of different patterns.

Table 4.6 shows the minimum, maximum and average values of all of the individuals in the Pareto front in the final generation. Interestingly, the representativity of these individuals is relatively flat compare to other optimizations. This may be due to the choice of target spectra not being a reactor and beryllium being a poor choice for moderating material. The total flux is adequate, being within the bounds of the other optimizations.

In conclusion, this optimization of the FNS would need to be more thoroughly thought out as the choice of beryllium may not be the best material to use to match the spectra. In addition, the interior genetic algorithm is converged and no longer pushes out the Pareto front of individuals in this case. The addition of a fourth material or potentially replacing polyethylene with beryllium may be a viable path forward for this optimization.

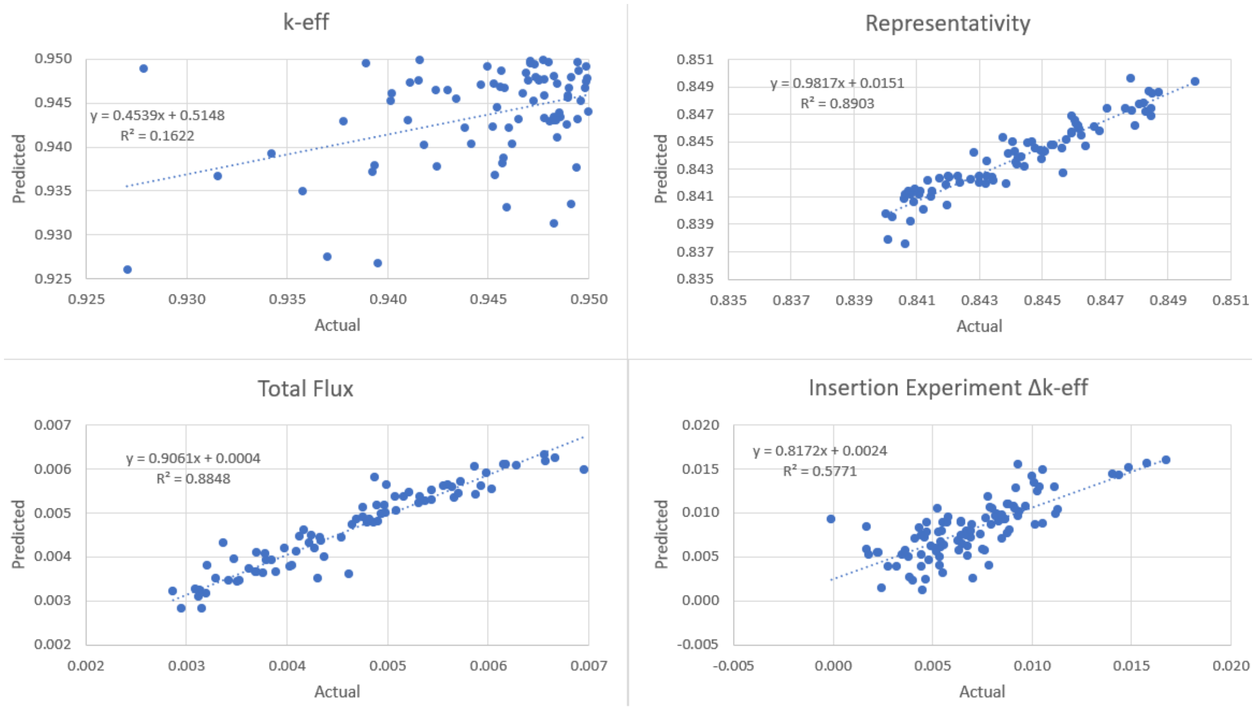


Figure 4.14: Little Boy Optimization CNN Surrogate Predicted vs. Actual Values, Generation 100

Table 4.6: Summary of Little Boy Optimization Objectives, Final Generation

	k-eff	Insertion Experiment delta k-eff	Representativity	Total Flux
Max	0.94999	-0.00126	0.84986	0.00695
Min	0.92668	-0.01599	0.84002	0.00287
Average	0.94498	-0.00837	0.84427	0.00461

4.4.2 Optimization of a Lead Reactor Spectra

The FNS could also be used to match a lead-cooled fast reactor, as was done in previously published FNS optimizations [52, 53]. Lead-cooled reactors feature a fast neutron spectra and are cooled by low pressure molten lead. These types of reactors have been designed and built around the world including in the United States, Europe and in the former Soviet Union [54]. Lead-cooled reactors face many challenges such related to corrosion and thermal-hydraulic performance. The uncertainty in the total neutron cross section for Pb-208, the most common isotope of natural lead, is above 5% in neutron energy ranges above 1 MeV in ENDF/VII.1. Better quantifying this uncertainty through nuclear data experiments would reduce the portion of uncertainty on these designs.

A genetic algorithm with CNN-surrogate acceleration was run on a target lead-cooled reactor spectrum. The three plate materials used were lead, polyethylene and 9.75% enriched uranium. The other options used in this optimization can be found in Table 4.7. Unlike the other optimizations, and due to a hardware error, this optimization was stopped at generation 61.

As seen in Figure 4.15, the optimization shows similar behavior to the other FNS optimizations with representativity increasing and total neutron flux decreasing near generation 50. Compared to the sodium metal optimization, the total flux decrease around generation 50 is not as pronounced. The ability for the surrogate models to predict their figures of merit in the last generation can be seen in Figure 4.16. The total neutron flux network shows the best predictive ability with an R^2 value of 0.91. The representativity and k-eff networks are still effective at their tasks, but less so, with R^2 values of 0.71 and 0.85, respectively. Unlike the sodium fast reactor spectrum optimization, the insertion experiment k-eff network is more effective at predicting the change in k-eff. The may be due to the relatively low absorption cross section of lead leading to the insertion of the material into the experimental volume being a reflector. Further study is needed to better understand this behavior.

Table 4.7: FNS Genetic Algorithm Hyper-Parameters, Limited Enriched Fuel Optimization Hyper-parameters, Lead Spectra Optimization

Description	Hyper-parameter
Stopping Criteria	Generation Count (61)
Parent Population	20
Child Population	80
Crossover Rate	50%
Mutation Rate	10% per plate
Mutation Type	Single plate material change
Initial Population	100 (randomly created)
Objective/Constraint Solver	MCNP
Material Types	Polyethylene Lead Enriched Uranium (Cassette 2A Only)

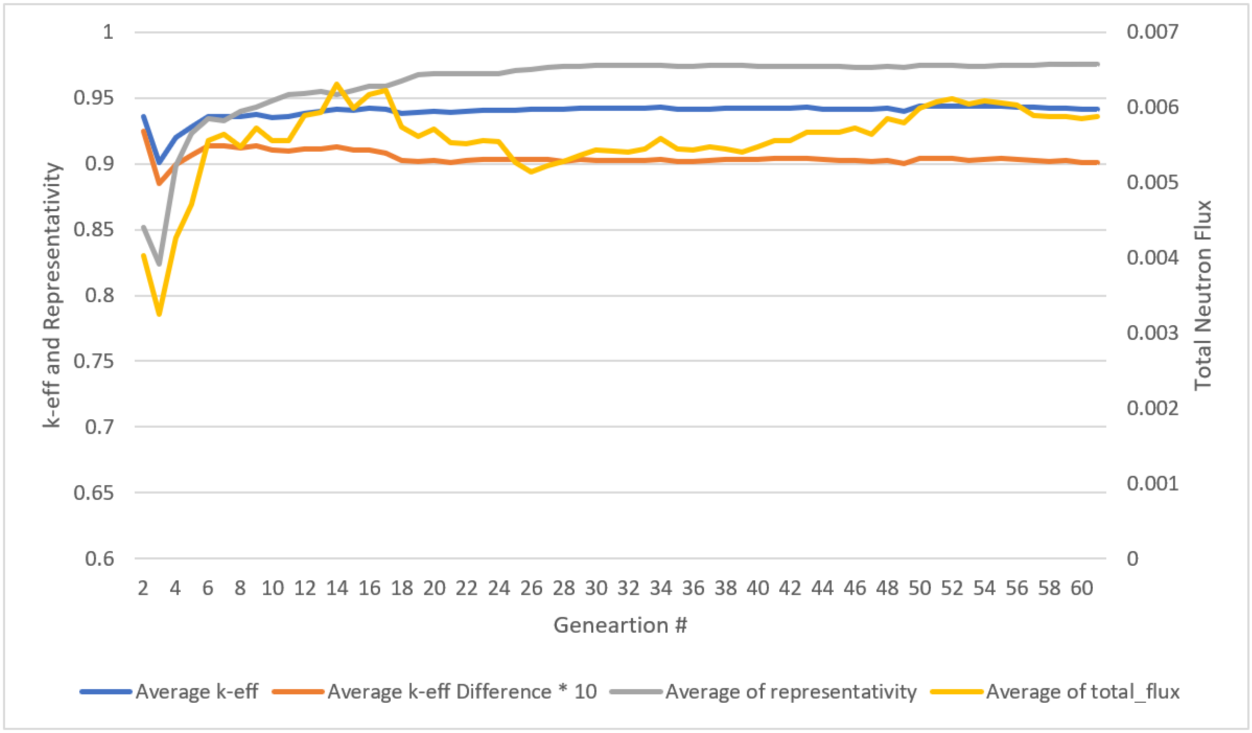


Figure 4.15: Lead Optimization Average Objective and Constraint Values

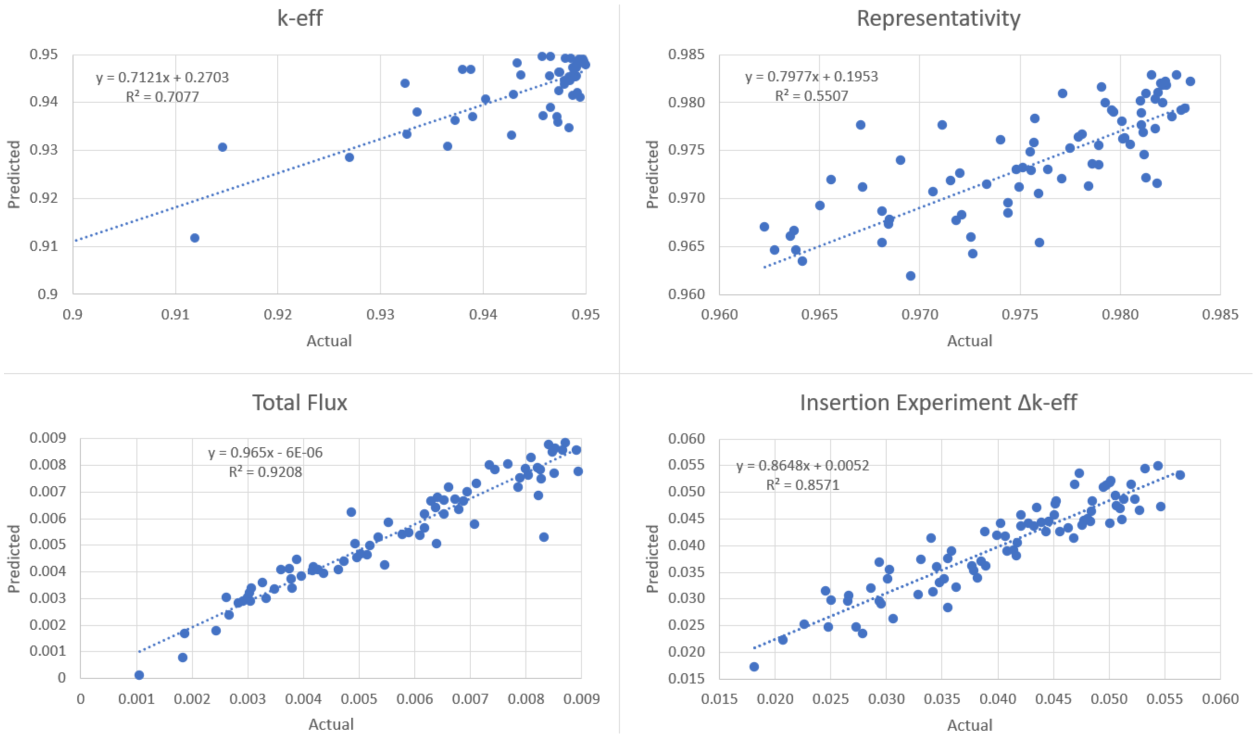


Figure 4.16: Lead Optimization CNN Surrogate Predicted vs. Actual Values, Generation 100

Despite the shorter optimization, the algorithm is able to produce FNS configuration comparable to the sodium plate optimization. The final minimum, maximum and average values in the Pareto front can be seen in Table 4.8. In comparison to the sodium metal optimization, maximum total neutron flux is significantly larger at 0.0095 neutrons per cm^2 per source particle. The representativity is similar. The insertion experiment delta k-eff is much more negative than in the sodium or Little Boy calculations with the most negative insertion experiment k-eff being over 50,000 pcm. The surrogate models are able to predict all of the objective and constraints effectively, as seen in Figure 4.16.

4.4.3 Optimization of a Lead Spectra with Sodium Metal, Natural Uranium, and Limited Fuel

In this optimization the FNS is optimized for a lead-cooled fast reactor spectra with constraints both on the number of enriched uranium plates and in where those plates can be located. Instead of using lead as a target material, sodium metal is used. This was done to explore some of the flexibility of the FNS to produce a target spectra with non-optimal plate materials. In addition, the 9.75% enriched uranium plates are restricted to only being placed in the center cassette (Pattern A) of zone B. This limit is enforced by requiring that plates 1-10 as counted from the neutron source side of the cassette to be alternating enriched fuel and polyethylene, as seen in Figure 4.17. If the cassette is beyond 10 plates in length other plate plate types can be added to the cassette in those locations. A further optimization of the final 80 parents by a CNN surrogate-based gradient descent algorithm will also be presented.

Genetic Algorithm Optimization

The options used for this genetic algorithm can be found in Table 4.9. The CNN surrogate models were trained in the same way as presented in Chapter 4 and used the same network architectures found during that optimization. The objectives for this optimization were fast neutron flux magnitude and representativity in the experimental volume. The k-eff

Table 4.8: Summary of Lead Optimization Objectives, Final Generation

	k-eff	Insertion Experiment delta k-eff	Representativity	Total Flux
Max	0.94993	-0.01812	0.98352	0.00950
Min	0.85212	-0.05646	0.96227	0.00105
Average	0.94154	-0.04046	0.97548	0.00588

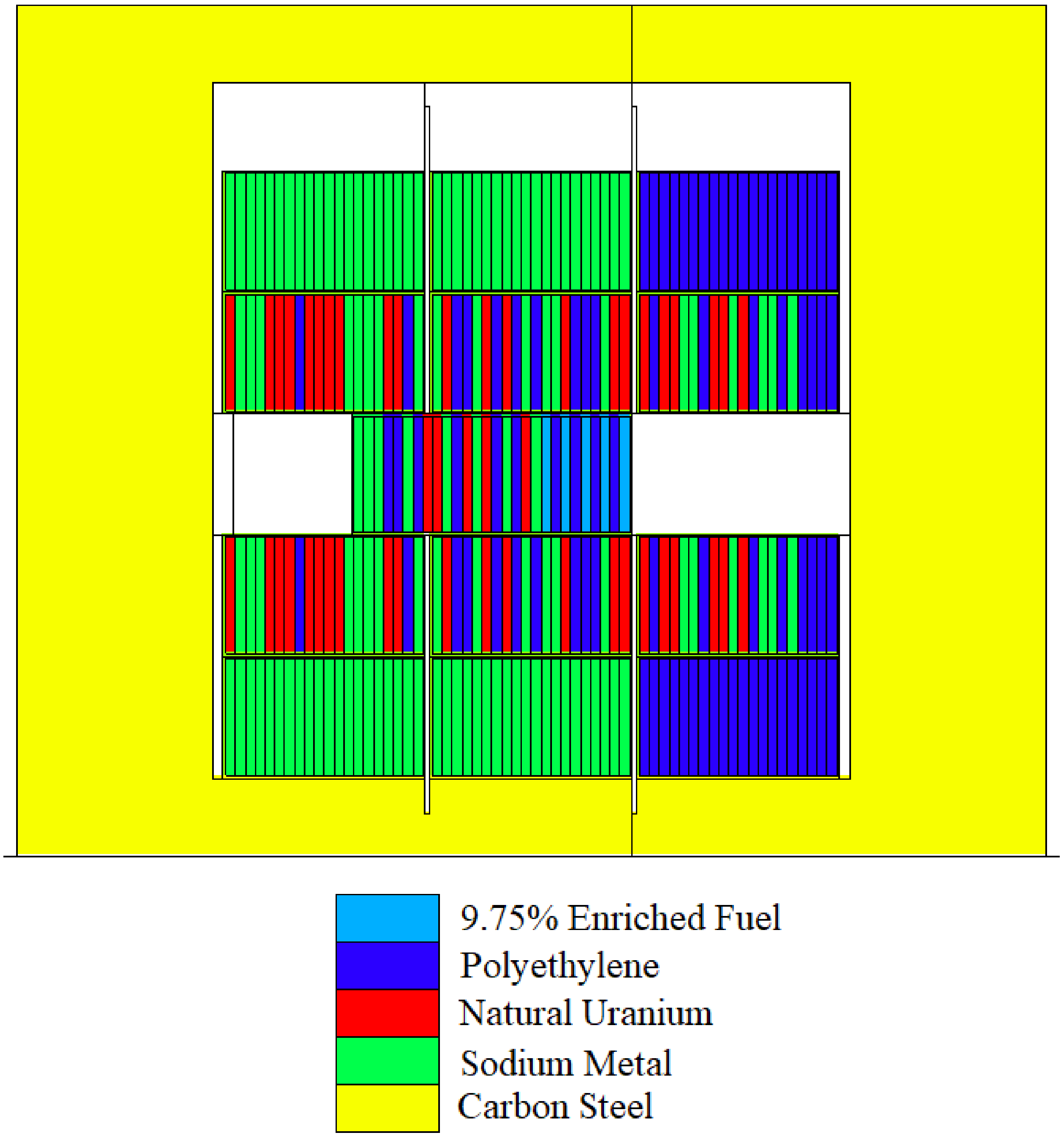


Figure 4.17: Limited Fuel Optimization Example FNS Configuration

Table 4.9: FNS Genetic Algorithm Hyper-parameters, Lead Spectra with Sodium Metal, Natural Uranium, and Limited Fuel

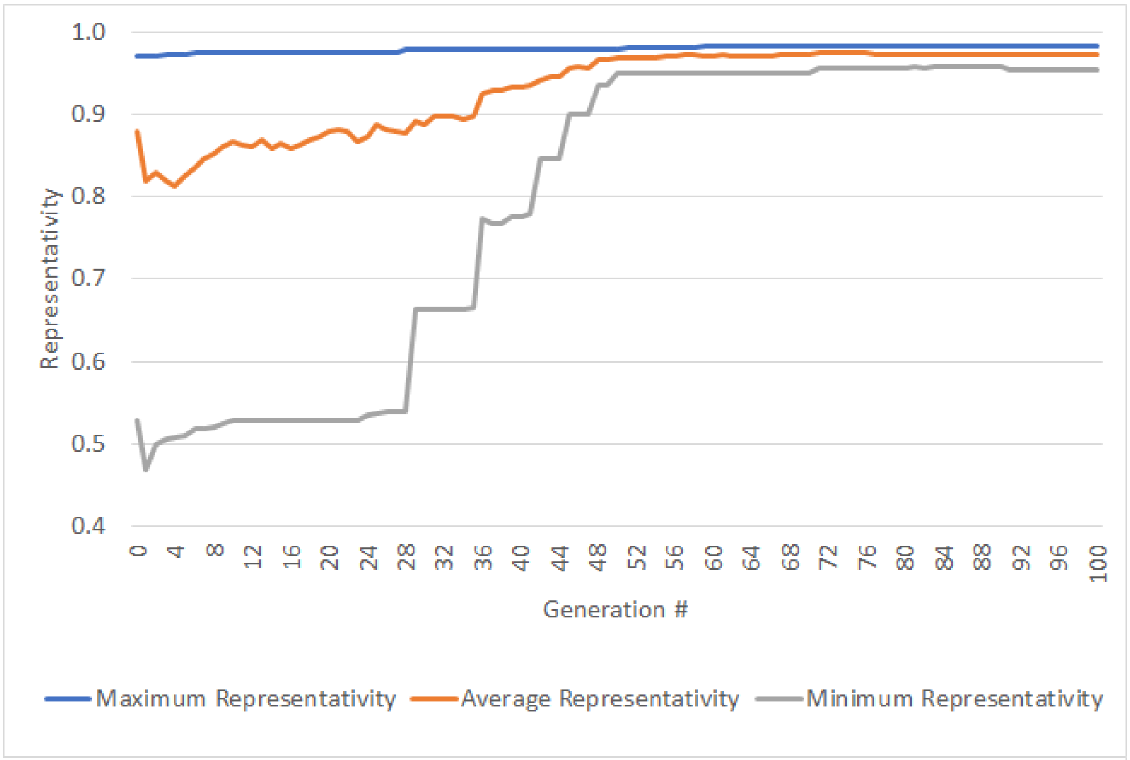
Description	Hyper-parameter
Stopping Criteria	Generation Count (100)
Parent Population	20
Child Population	80
Crossover Rate	50%
Mutation Rate	10% per plate
Mutation Type	Single plate material change
Initial Population	100 (randomly created)
Objective/Constraint Solver	MCNP
Material Types	Natural Uranium Polyethylene Sodium Metal Enriched Uranium (Cassette 2A Only)

constraint was not enforced nor calculated because with so few enriched uranium plates k-eff would not be near 0.95. Therefore, a single source-driven MCNP calculation was made for each potential pattern to produce the total flux and representativity objectives. A linearly increasing constraint on representativity was also enforced between generations 0 and 50. When maximally enforced, all parents are required to have a representativity above 0.95.

The CNN surrogate models were trained using individuals evaluated during the optimization. The CNN architectures used in Chapter 6 were used for this calculation. The only modification was a change to the first layer of the networks which allowed it to have 1 more material type (from 4 to 5). This was done to accommodate the inclusion of both natural and enriched uranium, along with polyethylene, sodium metal, and void.

The average, minimum and maximum representativity and total flux values from the parent population throughout the optimization can be found in Figure 4.18. These figures are presented together because plotting the data on one figure was not readable. Like the other optimizations, the trade-off between representativity and total flux is evident. As the representativity constraint takes hold leading up to generation 50, the minimum representativity of the parents is increased. At the same time, the maximum of the total flux of the parents is decreased. After the representativity constraint is enforced, the fast flux per source particle objective for all parents is increased by raising the minimum value while the maximum value is constant.

Figure 4.19 shows the total flux per source particle versus representativity for the parents of generations 0, 25, 50, 75 and 100. Initially the individuals are relatively sparse, with few individuals along the Pareto front. By generation 25 there is a clear Pareto front represented by the parents. There is also clearly grouping of the parents which is most prominent as the total flux increases/representativity decreases. These groupings are due to the limitation on the enriched fuel in cassette 2A. The largest flux/lowest representativity cases have the fewest number of enriched fuel plates at this generation. This behavior holds throughout the optimization and is likely driven by the relatively low neutron multiplication occurring in these configurations, making the gain in total flux from moving the experimental volume



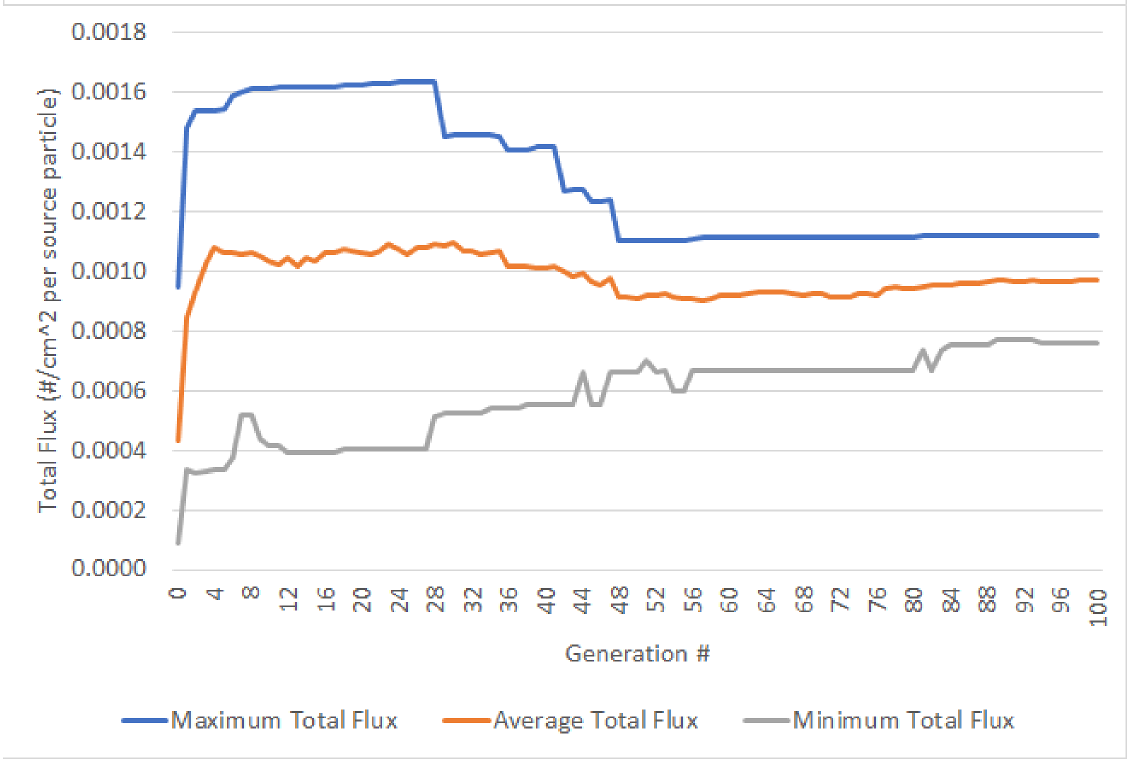


Figure 4.18: Total Flux and Representativity Average, Minimum and Maximum Parent Values for Limited Fuel Optimization

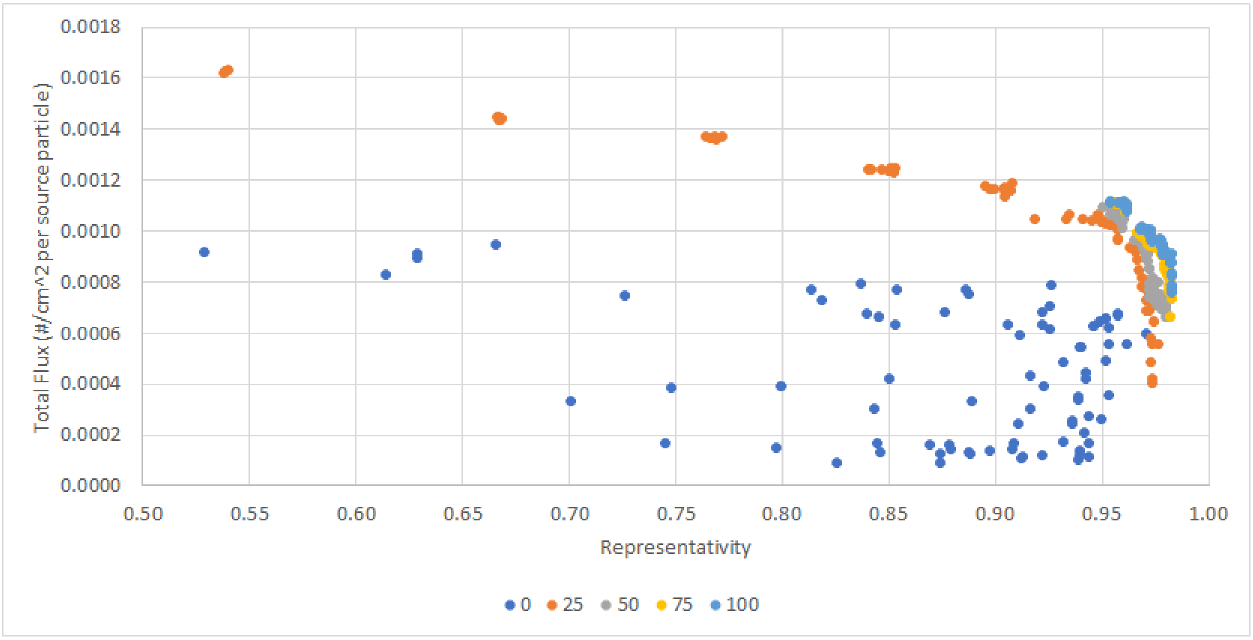


Figure 4.19: Limited Fuel Optimization Total Flux vs. Representativity for Selected Generations

closer to the source outweighing the potential multiplication. In later generations when the constraint is enforced on representativity, this effect is lessened.

This result shows that even with using a non-optimal material as the target plate material results in a neutron spectrum in the experimental volume which meets the representativity constraints. The next section details the addition of fully-enriched fuel back into the FNS design in order to bring the k-eff and total neutron flux in the experimental volume up.

Adding Enriched Fuel By Gradient Descent

This section discusses a further optimization of the final set of individuals by adding enriched fuel to the final set of Pareto front individuals. This was done by evaluating the effect of adding an enriched fuel plate to the pattern iteratively. Each possible plate swap is evaluated with the final generation (100) total flux surrogate model. Then all of the plate swaps for the pattern with the highest predicted total flux are evaluated, again with the CNN-based surrogate models. This process is repeated until either no plate swap produces a larger predicted total flux or until the total number of enriched uranium plates reaches 100. Due to the symmetry of the FNS, each plate in the pattern represents four plates. For each of the eighty final parents required 120 evaluations with the surrogate model in the first step. The second required 119, etc., until there were a total of 100 plates in the FNS pattern. In most cases this meant a total of 96 plates were added to the pattern.

The algorithm described above was run on the final Pareto front of parent individuals. The top twenty individuals sorted by the total neutron flux in the experimental volume can be seen in Table 4.10. Each of the patterns produced during this optimization were evaluated with MCNP eigenvalue and source-driven calculation to calculate k-eff, total neutron flux per source particle and representativity in the experimental volume.

The gradient descent optimization produced FNS patterns with an increased total flux compared to the Pareto front individuals which the optimization was based on. In the fuel restricted genetic algorithm optimization the maximum k-eff is 0.857 with a standard

Table 4.10: Top 20 Individuals Found by Gradient Descent

Individual Count	Grad. Desc. Step	k-eff	Total Flux	Representativity
5577	20	0.94900	0.00537	0.96026
5486	20	0.94904	0.00521	0.95253
6905	21	0.94698	0.00493	0.96525
6905	20	0.94674	0.00489	0.96521
7883	24	0.94793	0.00486	0.95410
5577	19	0.94772	0.00485	0.96024
7318	23	0.94792	0.00483	0.95628
5640	17	0.94757	0.00483	0.95721
7205	21	0.94525	0.00471	0.96882
7205	22	0.94439	0.00470	0.96866
6905	19	0.94473	0.00469	0.96631
5234	21	0.94545	0.00468	0.95719
5640	16	0.94370	0.00465	0.95655
7793	18	0.94552	0.00465	0.96202
7571	18	0.94871	0.00464	0.95037
7491	21	0.94932	0.00461	0.96573
7205	20	0.94427	0.00460	0.96976
7793	17	0.94289	0.00457	0.96229
5234	20	0.94284	0.00456	0.95766
6134	22	0.94730	0.00455	0.96204

uncertainty of 0.001. After the gradient descent algorithm there were twenty FNS patterns found which had k-eff values within 0.01 of the k-eff limit of 0.95.

With the increased k-eff of the FNS patterns comes increased neutron flux in the experiment volume. The maximum and minimum total neutron flux per source particle in the Pareto front is 0.00076 and 0.00112. After the gradient descent algorithm, the total neutron flux per second is 0.00537. This 4.8x increase in total flux would mean shorter experimental times. The original genetic algorithm produced a set of individuals with representativities ranging from 0.953 to 0.982. The maximum representativity of the FNS patterns found by the gradient descent is 0.96866 and the minimum is 0.95037.

When compared to the final Pareto front of the original CNN-surrogate optimization described in Section 4.4.2, the final individuals have lower representativity values, but similar or higher total neutron flux values. The optimization of the Pareto front by gradient descent is an effective method of optimization of the FNS patterns.

Chapter 5

Gradient Descent Optimization Results

5.1 Example Reactor System Optimization

This section summarizes the work previously published by the author [2]. In this work, a two dimensional reactor system is optimized using the Interior Point Method (IPM) [45]. The IPM is an optimization technique used on linear and non-linear functions which may be subject to both equality and/or non-equality constraints. The IPM takes as input the gradients with respect to the objective function (in this case, k-eff), and the constraint (in this case, the total mass of the fuel/moderator mixture). All Python and Matlab codes used in this analysis can be found here: https://github.com/jpevey/Dissertation-Work.

The geometry selected for this calculation is a 2 dimensional cuboid featuring a total side-length of approximately 1 meter that is discretization into 11 x 11 voxels. Void boundary conditions is enforced on the X and Y planes while a reflective boundary condition is used on the the Z planes. The materials of each of the 121 voxels is a mixture of either void or a mixture of polyethylene and uranium fuel. The fraction of the amount of void in each of the voxels varies between 0, for 100 percent moderator/fuel and 1.0 for 100 percent void. This value is defined as β.

This optimization comprises two parts. In the first, the IPM is used to maximize k-eff while respecting a constraint on the maximum amount of fuel/moderator mixture allowed in the system. This maximum is 57 total units of fuel/moderator mixture in the system. This maximum was found through a the discretization of a critical cylinder from a cylinder and solved for with SCALE to the 11 x 11 geometry. Both the objective (k-eff) and the gradients ((dk/k)/(dΣ/Σ)) are calculated with the SCALE module TSUNAMI. These sensitivities are converted into ((dk/k)/(dβ/β)) and fed into the IPM function. In the initial optimization, both the objective function and the sensitivities of the objective to the constraint values are used directly. The results of this optimization can be seen in figure 5.1. In this optimization, the k-eff of the system plateaus at approximately 0.977. The initial, two intermediate, and the final material compositions can be seen in figure 5.2. In the final step there is a total of 56.78 units of fuel/moderator mixture, slightly below the 57 unit maximum. This configuration is less than optimal both in total mass and in the inefficient placement of material throughout the geometry.

In the second step of this optimization and in order to push the geometry towards a discrete solution a penalty term is added both to the objective function and to the constraint sensitivities. This penalty function is defined as:

$$penalty = \gamma * \sum_{i=1}^{n} \beta_i(1 - \beta_i)$$

$$penalty' = \gamma * \frac{-2 * \beta_i + 1}{2\sqrt{penalty}}$$

In this penalty, the γ term is a hyper-parameter which governs the effectiveness that the penalty term has on the optimization. Four potential γ values were evaluated. In each case, the converged solution from the initial IPM optimization is used. Figure 5.3 shows the resulting geometries from values of 0.001, 0.01, 0.1 and 1.0. A penalty term of 0.1 produced a geometry both closest to the solution and the highest k-eff. Penalty terms of 0.01 and 0.001 produced inefficient solutions that are not well discretized. With a penalty term 0f 1.0, the

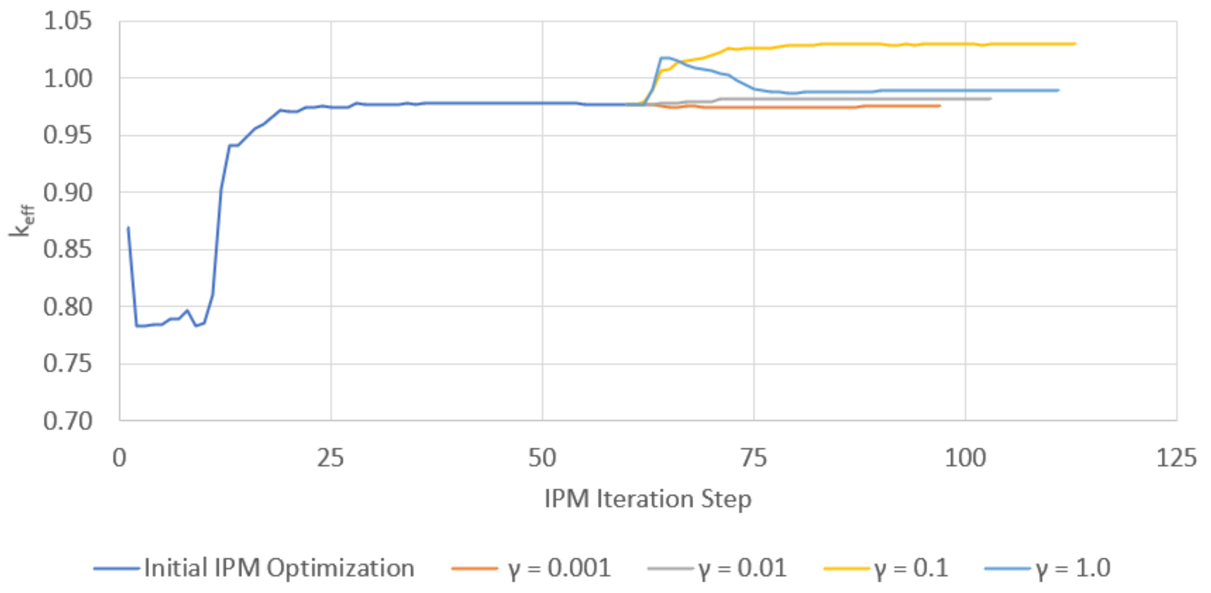
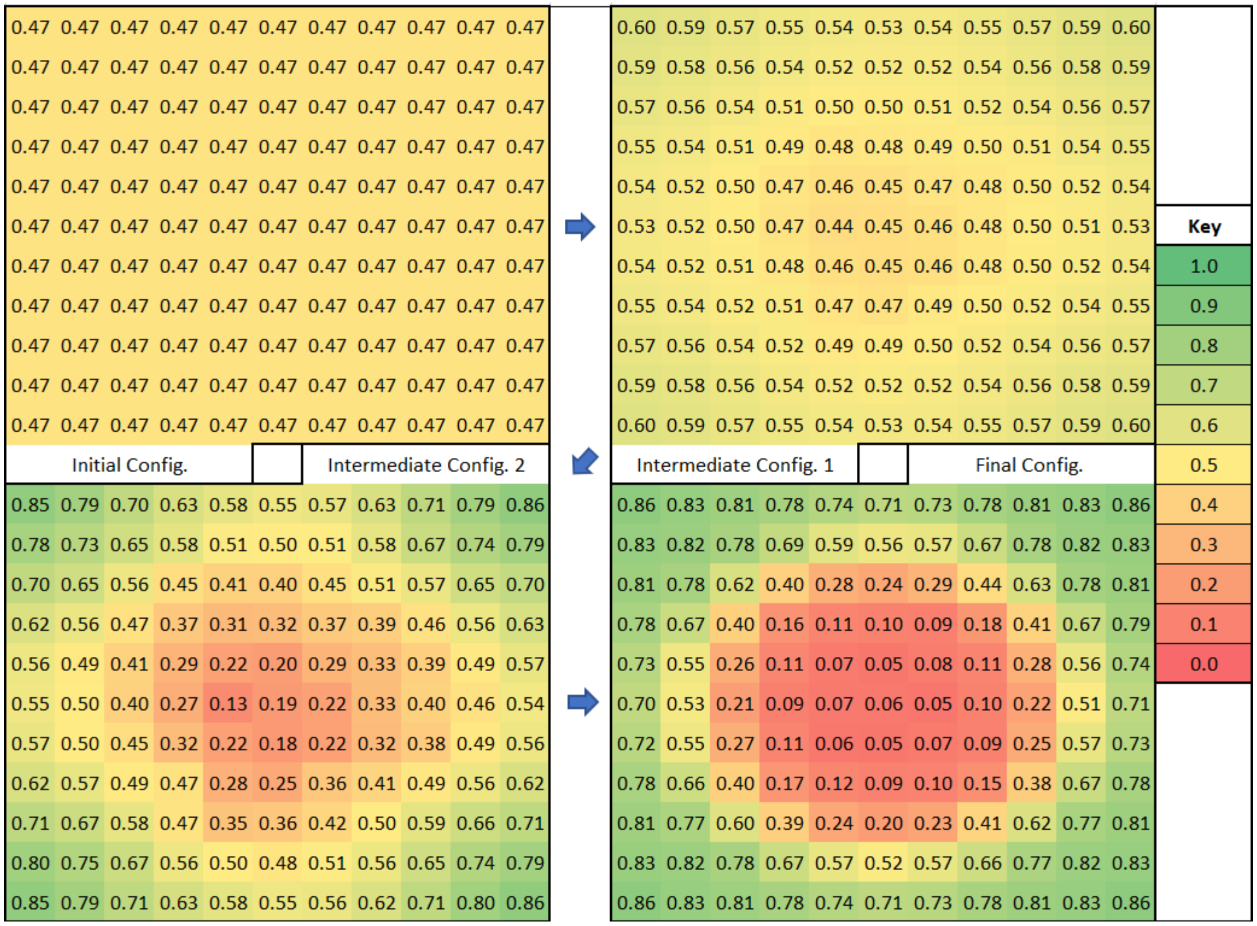


Figure 5.1: IPM Optimization of 11 x 11 Nuclear System k-eff vs. Optimization Step[2]



Initial Config.

0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47

Intermediate Config. 1

0.60	0.59	0.57	0.55	0.54	0.53	0.54	0.55	0.57	0.59	0.60
0.59	0.58	0.56	0.54	0.52	0.52	0.52	0.54	0.56	0.58	0.59
0.57	0.56	0.54	0.51	0.50	0.50	0.51	0.52	0.54	0.56	0.57
0.55	0.54	0.51	0.49	0.48	0.48	0.49	0.50	0.51	0.54	0.55
0.54	0.52	0.50	0.47	0.46	0.45	0.47	0.48	0.50	0.52	0.54
0.53	0.52	0.50	0.47	0.44	0.45	0.46	0.48	0.50	0.51	0.53
0.54	0.52	0.51	0.48	0.46	0.45	0.46	0.48	0.50	0.52	0.54
0.55	0.54	0.52	0.51	0.47	0.47	0.49	0.50	0.52	0.54	0.55
0.57	0.56	0.54	0.52	0.49	0.49	0.50	0.52	0.54	0.56	0.57
0.59	0.58	0.56	0.54	0.52	0.52	0.52	0.54	0.56	0.58	0.59
0.60	0.59	0.57	0.55	0.54	0.53	0.54	0.55	0.57	0.59	0.60

Intermediate Config. 2

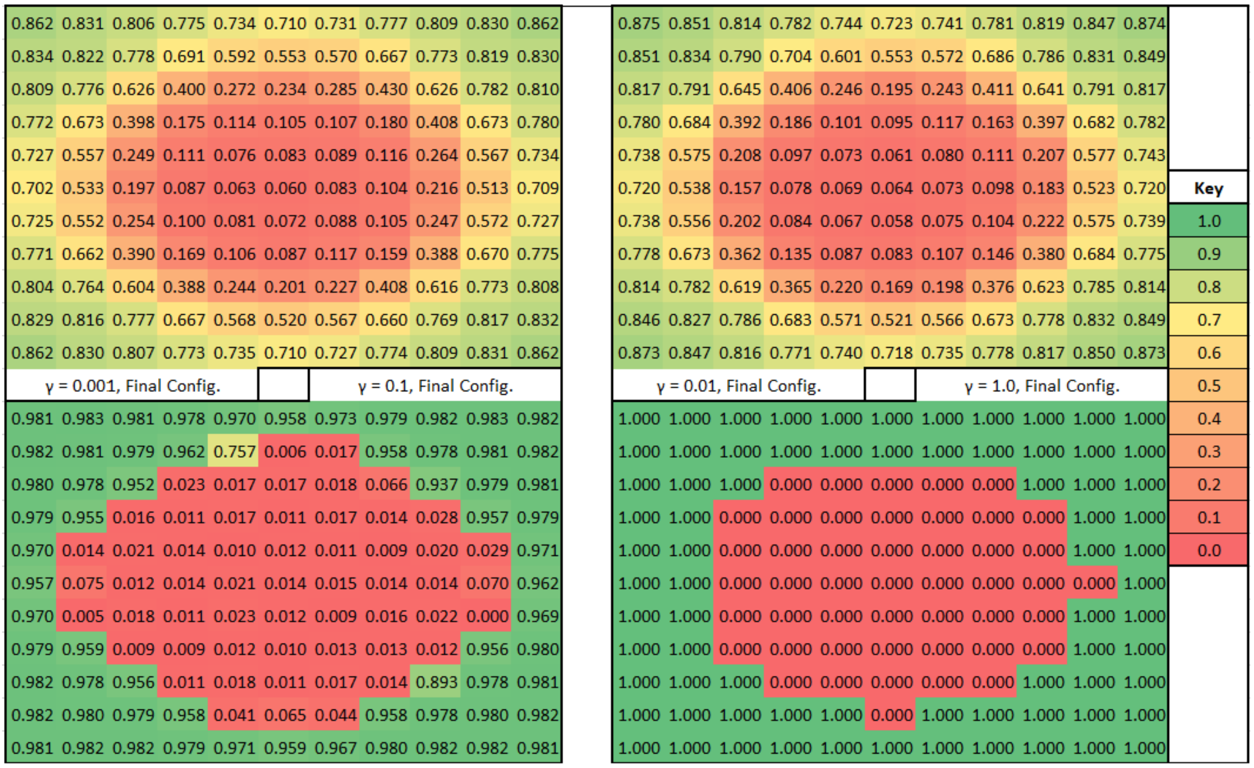
0.85	0.79	0.70	0.63	0.58	0.55	0.57	0.63	0.71	0.79	0.86
0.78	0.73	0.65	0.58	0.51	0.50	0.51	0.58	0.67	0.74	0.79
0.70	0.65	0.56	0.45	0.41	0.40	0.45	0.51	0.57	0.65	0.70
0.62	0.56	0.47	0.37	0.31	0.32	0.37	0.39	0.46	0.56	0.63
0.56	0.49	0.41	0.29	0.22	0.20	0.29	0.33	0.39	0.49	0.57
0.55	0.50	0.40	0.27	0.13	0.19	0.22	0.33	0.40	0.46	0.54
0.57	0.50	0.45	0.32	0.22	0.18	0.22	0.32	0.38	0.49	0.56
0.62	0.57	0.49	0.47	0.28	0.25	0.36	0.41	0.49	0.56	0.62
0.71	0.67	0.58	0.47	0.35	0.36	0.42	0.50	0.59	0.66	0.71
0.80	0.75	0.67	0.56	0.50	0.48	0.51	0.56	0.65	0.74	0.79
0.85	0.79	0.71	0.63	0.58	0.55	0.56	0.62	0.71	0.80	0.86

Final Config.

0.86	0.83	0.81	0.78	0.74	0.71	0.73	0.78	0.81	0.83	0.86
0.83	0.82	0.78	0.69	0.59	0.56	0.57	0.67	0.78	0.82	0.83
0.81	0.78	0.62	0.40	0.28	0.24	0.29	0.44	0.63	0.78	0.81
0.78	0.67	0.40	0.16	0.11	0.10	0.09	0.18	0.41	0.67	0.79
0.73	0.55	0.26	0.11	0.07	0.05	0.08	0.11	0.28	0.56	0.74
0.70	0.53	0.21	0.09	0.07	0.06	0.05	0.10	0.22	0.51	0.71
0.72	0.55	0.27	0.11	0.06	0.05	0.07	0.09	0.25	0.57	0.73
0.78	0.66	0.40	0.17	0.12	0.09	0.10	0.15	0.38	0.67	0.78
0.81	0.77	0.60	0.39	0.24	0.20	0.23	0.41	0.62	0.77	0.81
0.83	0.82	0.78	0.67	0.57	0.52	0.57	0.66	0.77	0.82	0.83
0.86	0.83	0.81	0.78	0.74	0.71	0.73	0.78	0.81	0.83	0.86

Key
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

Figure 5.2: Selected Geometries During Initial IPM Optimization[2]



0.862	0.831	0.806	0.775	0.734	0.710	0.731	0.777	0.809	0.830	0.862
0.834	0.822	0.778	0.691	0.592	0.553	0.570	0.667	0.773	0.819	0.830
0.809	0.776	0.626	0.400	0.272	0.234	0.285	0.430	0.626	0.782	0.810
0.772	0.673	0.398	0.175	0.114	0.105	0.107	0.180	0.408	0.673	0.780
0.727	0.557	0.249	0.111	0.076	0.083	0.089	0.116	0.264	0.567	0.734
0.702	0.533	0.197	0.087	0.063	0.060	0.083	0.104	0.216	0.513	0.709
0.725	0.552	0.254	0.100	0.081	0.072	0.088	0.105	0.247	0.572	0.727
0.771	0.662	0.390	0.169	0.106	0.087	0.117	0.159	0.388	0.670	0.775
0.804	0.764	0.604	0.388	0.244	0.201	0.227	0.408	0.616	0.773	0.808
0.829	0.816	0.777	0.667	0.568	0.520	0.567	0.660	0.769	0.817	0.832
0.862	0.830	0.807	0.773	0.735	0.710	0.727	0.774	0.809	0.831	0.862

γ = 0.001, Final Config.

γ = 0.1, Final Config.

0.981	0.983	0.981	0.978	0.970	0.958	0.973	0.979	0.982	0.983	0.982
0.982	0.981	0.979	0.962	0.757	0.006	0.017	0.958	0.978	0.981	0.982
0.980	0.978	0.952	0.023	0.017	0.017	0.018	0.066	0.937	0.979	0.981
0.979	0.955	0.016	0.011	0.017	0.011	0.017	0.014	0.028	0.957	0.979
0.970	0.014	0.021	0.014	0.010	0.012	0.011	0.009	0.020	0.029	0.971
0.957	0.075	0.012	0.014	0.021	0.014	0.015	0.014	0.014	0.070	0.962
0.970	0.005	0.018	0.011	0.023	0.012	0.009	0.016	0.022	0.000	0.969
0.979	0.959	0.009	0.009	0.012	0.010	0.013	0.013	0.012	0.956	0.980
0.982	0.978	0.956	0.011	0.018	0.011	0.017	0.014	0.893	0.978	0.981
0.982	0.980	0.979	0.958	0.041	0.065	0.044	0.958	0.978	0.980	0.982
0.981	0.982	0.982	0.979	0.971	0.959	0.967	0.980	0.982	0.982	0.981

0.875	0.851	0.814	0.782	0.744	0.723	0.741	0.781	0.819	0.847	0.874
0.851	0.834	0.790	0.704	0.601	0.553	0.572	0.686	0.786	0.831	0.849
0.817	0.791	0.645	0.406	0.246	0.195	0.243	0.411	0.641	0.791	0.817
0.780	0.684	0.392	0.186	0.101	0.095	0.117	0.163	0.397	0.682	0.782
0.738	0.575	0.208	0.097	0.073	0.061	0.080	0.111	0.207	0.577	0.743
0.720	0.538	0.157	0.078	0.069	0.064	0.073	0.098	0.183	0.523	0.720
0.738	0.556	0.202	0.084	0.067	0.058	0.075	0.104	0.222	0.575	0.739
0.778	0.673	0.362	0.135	0.087	0.083	0.107	0.146	0.380	0.684	0.775
0.814	0.782	0.619	0.365	0.220	0.169	0.198	0.376	0.623	0.785	0.814
0.846	0.827	0.786	0.683	0.571	0.521	0.566	0.673	0.778	0.832	0.849
0.873	0.847	0.816	0.771	0.740	0.718	0.735	0.778	0.817	0.850	0.873

γ = 0.01, Final Config.

γ = 1.0, Final Config.

1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000
1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000
1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000
1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000
1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000
1.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000
1.000	1.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000
1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Key
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

Figure 5.3: Selected Geometries During Second IPM Optimization[2]

geometry is well discretized but the penalty is over-weighted in the optimization versus the constraint derivatives. In this geometry there is only 47 units of fuel/moderator mixture.

5.2 TSUNAMI k-eff Sensitivities for the FNS

This section discusses the use of k-eff sensitivities calculated with TSUNAMI (SCALE) for FNS analysis. Previously published work by the author has shown the potential effectiveness of using TSUNAMI-calculated gradients for nuclear system optimization [2]. In this analysis, an FNS input from the final Pareto front of the surrogate-model optimization is used as an example comparing exploring the TSUNAMI-calculated sensitivities for optimizing the FNS.

5.2.1 The Calculation of TSUNAMI Sensitivities

The input selected for this analysis is source_calc_gen_94_ind_7549.inp. The nuclear parameters and geometry for this example is summarized in Table 5.1. This input has a k-eff of 0.84565 and was selected due to it being the lowest k-eff example in the final Pareto front.

TSUNAMI calculates sensitivities of k-eff to changes in the macroscopic cross section of materials throughout the model. The CLUTCH method that is used within TSUNAMI to calculated the sensitivities is discussed in Section 3.3.2. In this analysis all of the materials within each cassette are homogenized. The output of TSUNAMI is the sensitivity of k-eff to changes in the macroscopic cross section of the homogeneous materials in the cassettes. The plate materials in this model are sodium metal, polyethylene, and 9.75% enriched uranium.

The k-eff sensitivities are calculated by a TSUNAMI input created by modifying a KENO-V input with the options outlined in Table 5.2. The majority of these changes are in the parameter card with an additional grid geometry defined for the F* mesh. A KENO-V SCALE model of the FNS used in this analysis, detailed in Section 3.4.2. This model was modified with homogenized plate materials in each cassette. This simplification of the geometry decreases the run time of the calculation due to the increased volume at the expense

Table 5.1: FNS TSUNAMI-Clutch Example Input

Option	Value
k-eff	0.84565
Pattern Material Key:	2 - Polyethylene, 3 - 9.75% Enriched Fuel, 4 - Sodium Metal
Pattern 1B	4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4
Pattern 1C	3 3 3 4 4 3 3 3 4 4 4 4 3 3 4 4 3 3 4 4
Pattern 2B	4 4 4 4 4 4 4 4 4 3 3 4 3 4 4 3 4 3 3 3
Pattern 2C	3 4 4 3 3 4 4 3 3 4 3 4 4 4 4 3 3 3 3 3
Pattern 3B	3 3 3 3 2 3 3 4 4 3 2 4 4 4 3 4 3 3 2 4
Pattern 3C	3 4 3 2 4 4 2 3 4 4 4 4 4 4 2 4 3 4 2 3
Pattern 2A	3 3 4 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3 3

Table 5.2: FNS TSUNAMI-Clutch Options

Option	Value
Neutrons Per Generation (NPG)	10,000
Generations (NSK)	410
Specifies TSUNAMI Method (CET)	1
F* Mesh Latent Generations (CFP)	10
Specify F* Mesh Grid (CGD)	Yes
Specify F* Mesh Grid Number (MSH)	1
Mesh X Dimension (XLINEAR)	41 -1.27 80.7331
Mesh Y Dimension (YLINEAR)	41 -1.27 81.153
Mesh Z Dimension (ZLINEAR)	46 0 90.4875

of plate-by-plate resolution. These calculations were completed using SCALE version 6.2.4 on a Windows laptop. The calculation required approximately 2.5 hours of computer time on a single Intel i7-6700HQ CPU.

5.2.2 Discussion of TSUNAMI-calculated k-eff Sensitivities

The TSUNAMI calculation was run as described in the previous section. The output from this calculation are the standard SCALE output file, the message file (.msg) and the sensitivity data file (.sdf). The sensitivity data file contains all of the calculated sensitivities for all of the materials as defined in the model. If materials are used in multiple locations in the model then the calculated sensitivity would be the combination of them.

Figures 5.4, 5.5 and 5.6 are the group-wise sensitivities for the sodium, 9.75% enriched fuel and polyethylene in the FNS model, respectively. The mixture numbers in these figures correspond to zone materials in the following way. Mixtures 1 and 2 are for cassette patterns 1A and 1B, mixtures 3 and 4 are zone patterns 2B and 2C, and mixtures 5 and 6 are for cassette patterns 3B and 3C. Mixture 7 is the center cassette pattern, 2A. Each figure also includes the region-integrated sensitivity and, for the polyethylene and sodium, the sensitivities of the outer ring of cassette patterns. This outer ring of cassettes does not contain any fuel and is completely filled with either sodium metal in zones 1 and 2 or polyethylene in zone 3. The outer materials are mixtures 51 and 52 for zones 1/2 and 3, respectively.

The Na-23 sensitivity is shown in Figure 5.4. The largest sensitivities of the FNS in the examined configuration is in the sodium in cassette patterns D, E and F. The total sensitivity of this material is 0.0455 $(dk/k)/(d\Sigma/\Sigma)$. Of this total, the largest integrated sensitivity for this material is found in the elastic cross section at 0.463. This value is larger than the total due to the negative effects of absorption cross sections such as capture and (n,gamma) reactions. Increasing the absorption cross section decreases k-eff. Because these cassettes are on the periphery of the FNS, they act as a reflector and increasing the fast cross sections reduce the leakage of the system, increasing the neutron multiplication.

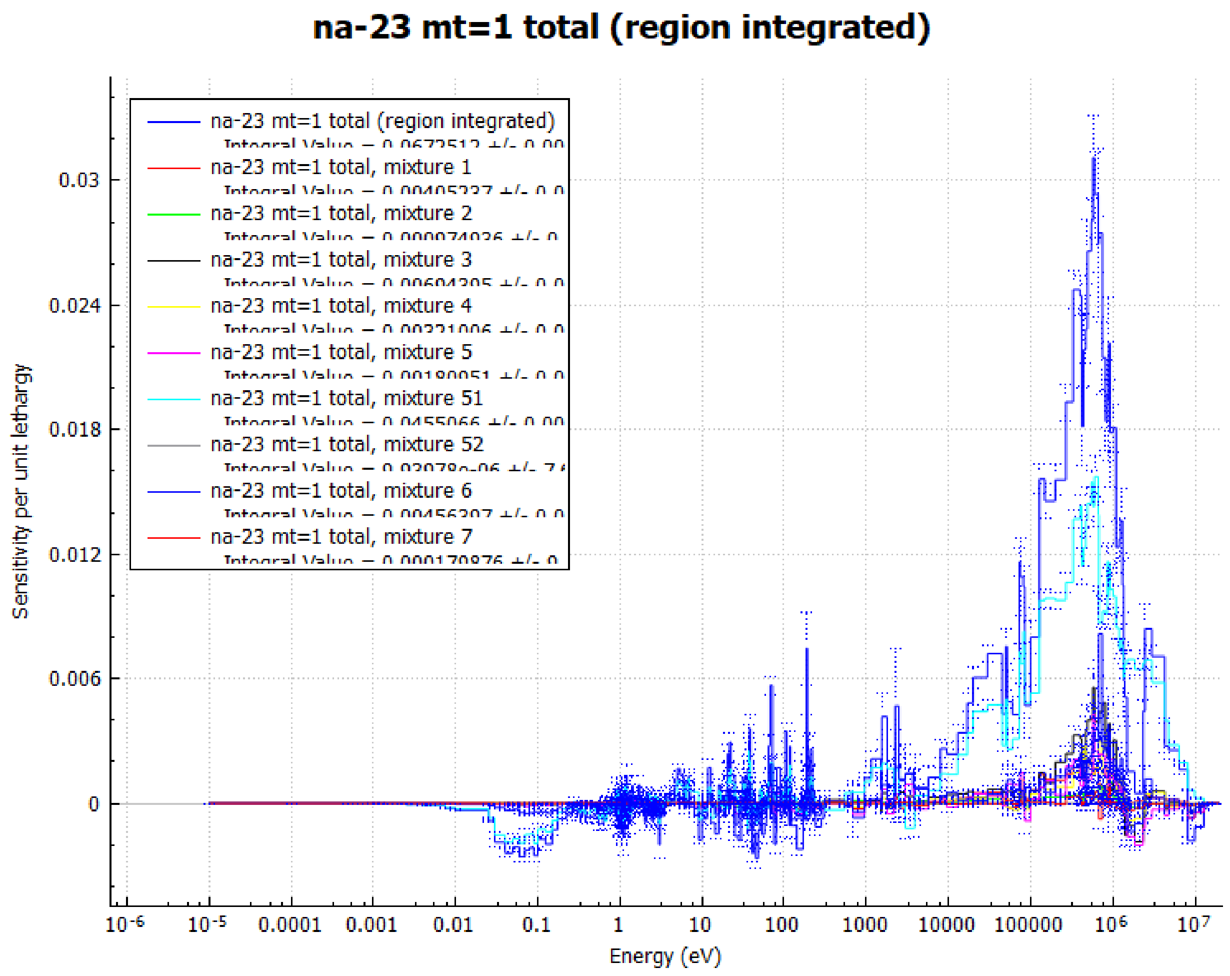


Figure 5.4: Sensitivity of Homogenized Sodium Metal in FNS Example

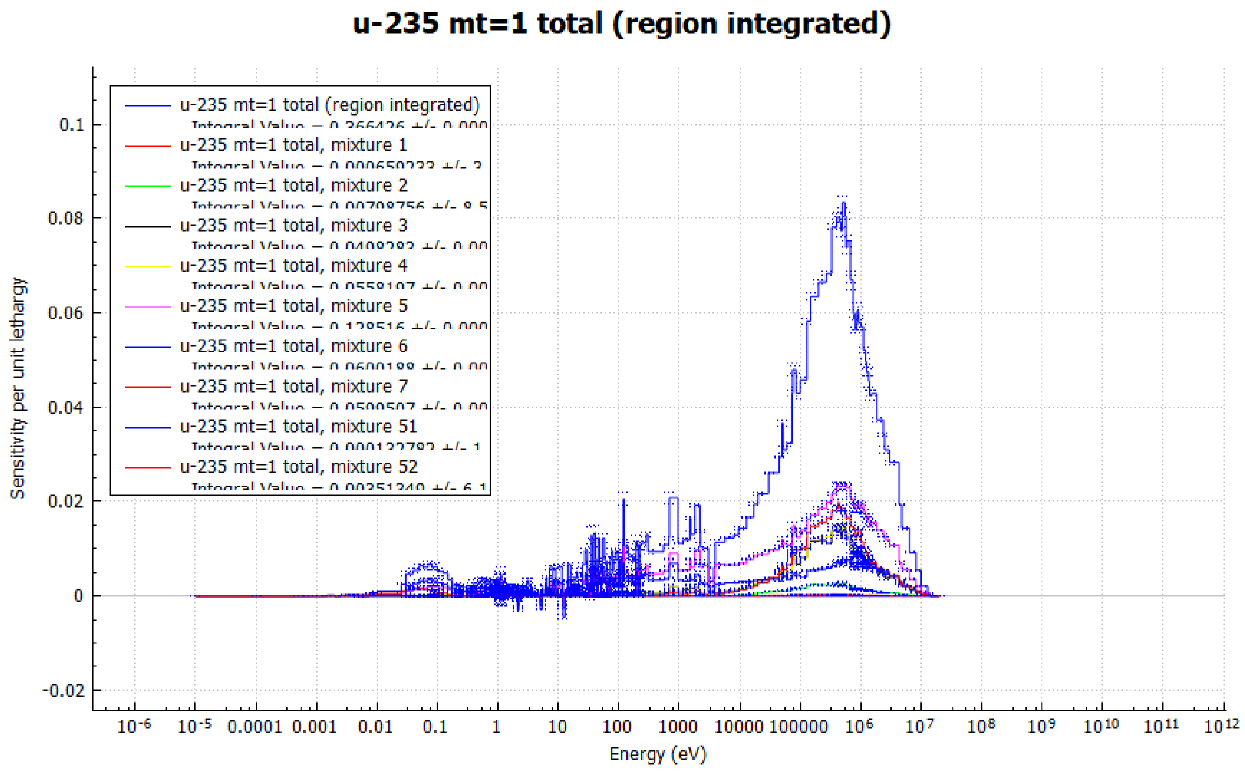


Figure 5.5: Sensitivity of Homogenized U-235 in FNS Example

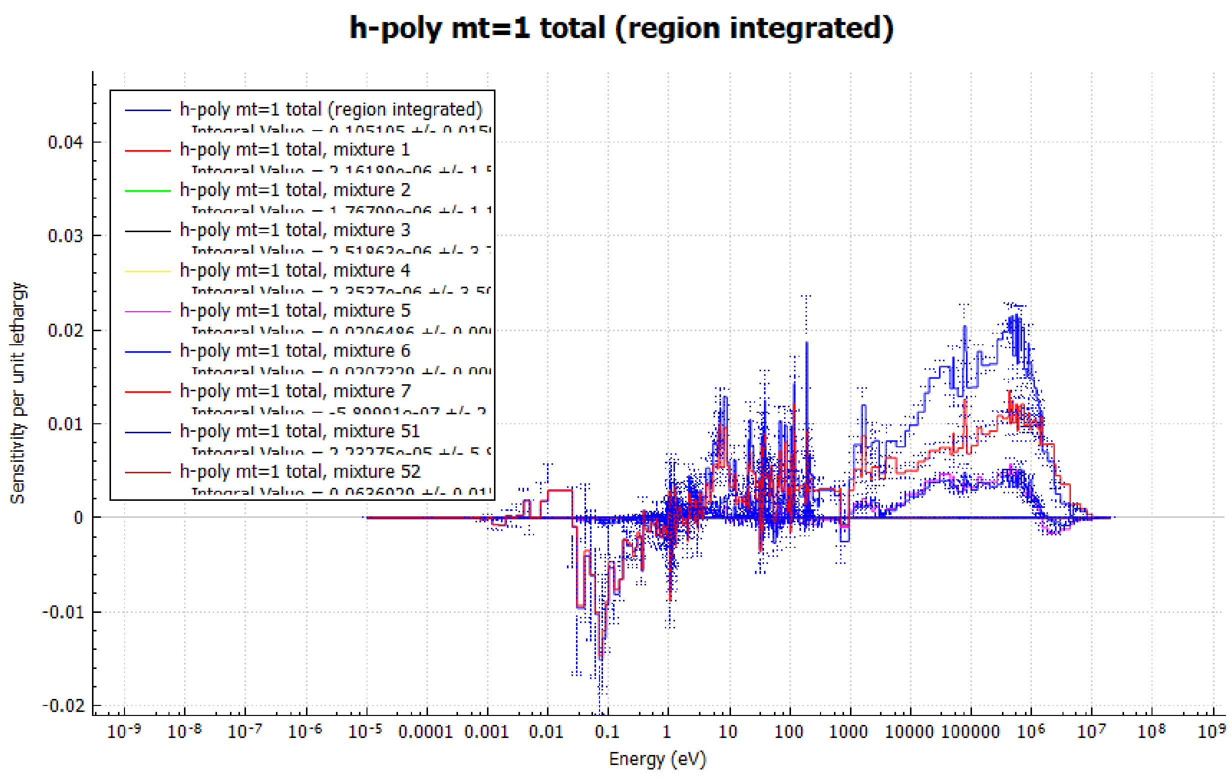


Figure 5.6: Sensitivity of Homogenized Polyethylene in FNS Example

The U-235 total (region integrated) sensitivity is shown in Figure 5.5. The total sensitivity for U-235 is, like the Na-23 total sensitivity, more prominent in the fast region above 10,000 eV. The total cross section sensitivity is positive with the most positive values from fission, which, when region-integrated, is +0.444 (dk/k)/(dΣ/Σ). This positive sensitivity is offset slightly by a negative sensitivity to (n, gamma) reactions at -0.080 (dk/k)/(dΣ)/(Σ).

The last sensitivity presented is for polyethylene and can be seen in Figure 5.6. Like the Na-23 sensitivity, the polyethylene sensitivity is largest in the periphery cassettes in zone 3. The largest positive contributor to the sensitivity are elastic scattering while the most negative sensitivity is (n,gamma) reactions. The n,gamma reaction sensitivities are most negative at energies less than 1 eV. The cassette patterns which have the second and third highest sensitivities are patterns 3B and 3C, which are around the neutron generator.

5.2.3 Comparison of TSUNAMI and Directly Calculated k-eff Sensitivities

The sensitivities calculated in the above section were compared with direct perturbations of the FNS pattern being evaluated. An MCNP model of the FNS was run for each potential plate material swap of materials which are already in the cassette. For example, the 2A cassette pattern which has 2 sodium plates and 17 fuel plate, a total of 19 separate k-eff values were calculated. Of these, 17 of those were with a sodium plate being replaced by a fuel plate and two with sodium plates replacing the fuel plates. In cassettes with all three materials, each plate location was replaced with the other plate types.

The TSUNAMI sensitivities were calculated with homogenized cassette materials while the MCNP perturbations were calculated with explicit plate materials. Using the TSUNAMI sensitivities an estimate for the effect of swapping one plate material with another was done. The sensitivities are in units of percent change in k-eff due to a percent change in the macroscopic cross section. Because the macroscopic cross section is the combination of the microscopic cross section and the number density of the isotope in question, the sensitivity can be interpreted as the sensitivity to changes in the atomic density of the plate materials in each cassette.

The TSUNAMI-calculated sensitivities were used to predict the effect that changes in the plate pattern of each cassette would have on k-eff. Figure 5.7 shows the results of the TSUNAMI predictions and the bounding MCNP-calculated k-eff values of those plate swaps. This figure shows the maximum error that would be expected in this iteration of the FNS if the TSUNAMI sensitivities were used to predict k-eff changes due to plate swaps. In some cases, such as in cassette patterns 1b and 2a, using the TSUNAMI sensitivities predicts relatively little change in k-eff due to plate swaps. But in both the fuel to sodium and sodium to fuel MCNP calculations in cassette pattern 2a there is a +/- 4,000 pcm swing in k-eff. The plates in the 2a, the center cassette show a high sensitivity to k-eff while, when homogenized, that sensitivity is diminished markedly. In ten of the plate swaps predicted did TSUNAMI predict a change in k-eff that was beyond the minimum or maximum calculated by MCNP.

This result shows that the homogenization of the cassettes may be too much of an approximation for the sensitivities to have much ability to predict actual changes in k-eff due to plate swaps.

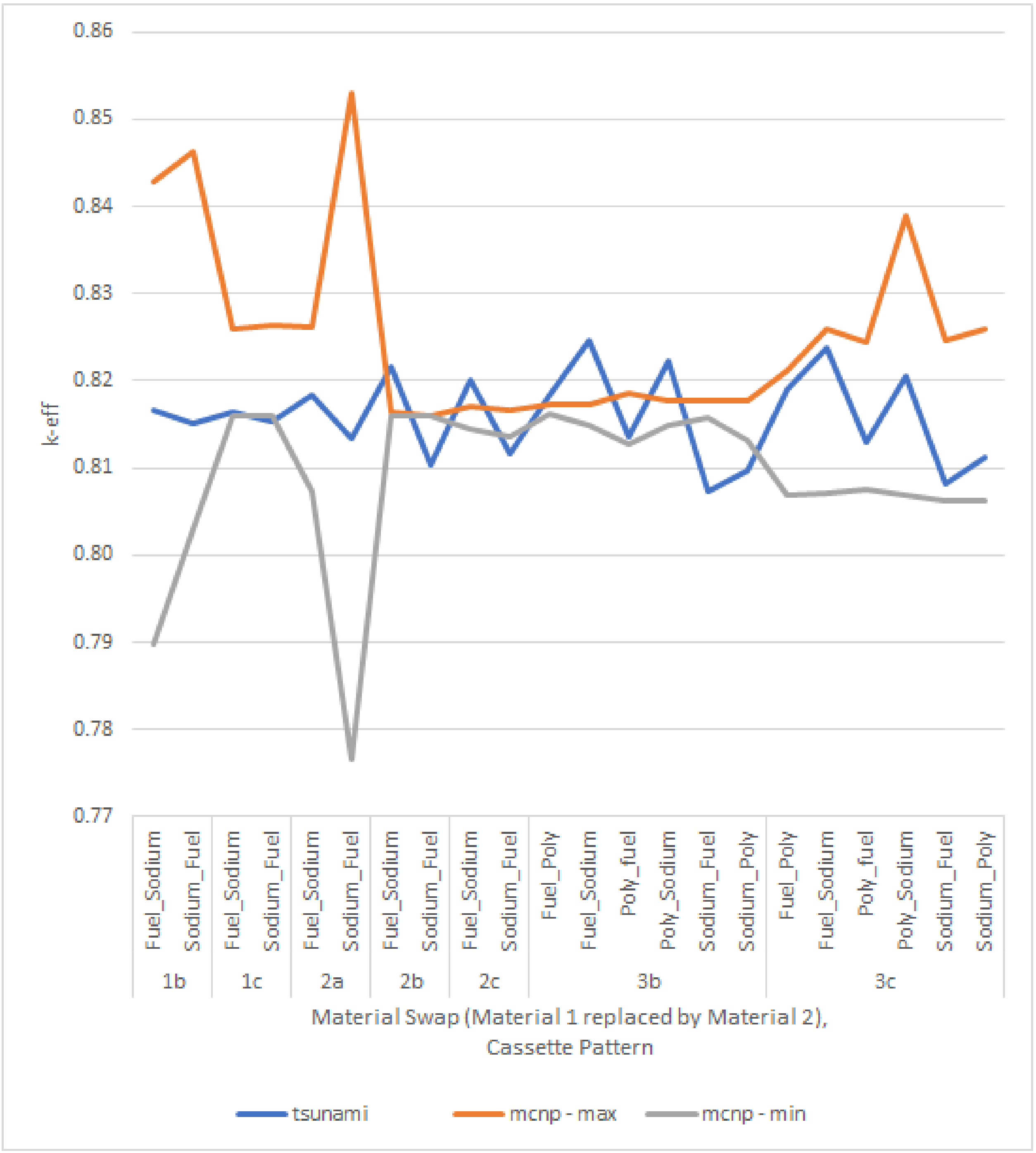


Figure 5.7: Calculated k-eff with MCNP and Predicted k-eff using TSUNAMI Sensitivities for Cassette-wise Homogenized FNS Model

Chapter 6

Conclusion and Future Work

The following sections provide a brief summary of the work presented in this dissertation, a summary of how the proposal objectives have been met and several avenues for future research based on this work.

6.1 Conclusion

The primary goal of this research is to implement a multi-objective genetic algorithm of the FNS. The objectives of the optimization are the total neutron flux per source particle in the experimental volume, the representativity of the flux spectrum in the experimental volume and the difference in k-eff of the FNS with and without filling the experimental volume with a target material. These objectives are evaluated using MCNP, a Monte Carlo neutronics solver. A constraint is also enforced on the k-eff of the FNS of 0.95. This constraint is enforced in a similar way as simulated annealing where the constraint is lax at the beginning of the optimization, but is gradually enforced through the optimization. This is done to ensure the safe operation of the FNS. The FNS presents a difficult optimization problem with more than 3.69988 x 10^{71} potential designs with a three material types.

The NSGA-II optimization of the three objectives related to the FNS design are presented. The most thoroughly discussed optimization is of a sodium metal-based

FNS design. These are compared to surrogate-model based genetic algorithms in which convolutional neural networks are trained on the patterns evaluated during the optimization. It was shown that the building of the surrogate models in-line with the genetic algorithm produces a set of surrogate models that are able to predict the objective functions around the design space of the parents.

The use of gradient descent algorithms for the optimization of nuclear systems, including the FNS, are also presented. The derivatives calculated by the TSUNAMI module in SCALE are shown to be useful for driving the optimization of a simple two dimensional nuclear system. In relation the the FNS, the derivatives calculated by a simplified TSUNAMI model are compared to the actual evaluations of k-eff with plate-swaps. Unsurprisingly, the TSUNAMI model produces average values which may be of use for future optimizations of the FNS.

In summary, the methods presented in this work push the boundary of the possible methods of optimizing nuclear systems. Because of this work, and the work of others in the design group, the construction of the FNS is on-track to be built and experiments will be run.

6.2 Meeting Proposal Objectives

1. Develop a genetic algorithm for the multi-objective optimization of nuclear experiments.

   Met in chapter 3 and in published journal articles [52, 53], where the methodology for the optimization of a nuclear experiment using genetic algorithms is presented.

2. Develop methodologies for the use of feature-extracting neural networks to be used as a surrogate models for neutronic calculations.

   Met in chapter 3 and published [52], where the CNN-surrogate optimization is presented and then applied to the optimization of the FNS.

3. Develop methodologies for the use of gradient descent with directly calculated sensitivities (dk-eff/k-eff/dΣ/dΣ) for nuclear systems.

   Met in chapters 3 and published [2], where gradient descent methods and the methods for producing the gradients are outlined.

4. Implement these methods (#1, #2, #3) into a Fast Neutron Source optimization (#1) as a proof of concept for their applicability to nuclear design.

   Met in chapters 4 and 5 where the methodologies outlined for #1, #2, #3 are applied to the FNS experiment design.

6.3 Future Work

There are many opportunities for research related to the continued optimization of the FNS. The methods laid out in this dissertation could be used to expand the number of objective functions, increase the complexity of the FNS, and, with the application of newer neural network architectures, further increase the effectiveness of the genetic algorithm. Other FNS experiments such as filter material optimization for isotope production and neutron detector design would also be possible. These experiments rely on well-characterized flux spectra which the FNS would produce.

The CNN-based surrogate models have been shown to be able to predict neutronic parameters of systems such as k-eff, neutron flux, and representativity. But those are not the only objectives that are relevant to a successful FNS configuration. Using a well-trained surrogate could open the door to objectives related to criticality safety such as introducing a preference for designs which have the lowest increase in reactivity from one or more plate or cassette loading mistakes. Although the CNN-surrogate prediction should not be relied upon for the final safety evaluation of the FNS, they have been shown to be able to accurately predict small perturbations of the FNS designs during the optimization.

The FNS as presented in this work is a complex nuclear system, but by leveraging surrogate models and/or TSUNAMI-based heuristics, the complexity could be increased.

Adding more material types would further increase the complexity but may allow the FNS to be designed to match more complex neutron spectra such as the Little Boy bomb spectra discussed in Section 4.4.1.

This work demonstrates the benefits of CNN-based surrogate models but other neural network architectures and methods may also be useful for predicting nuclear figures of merit such as k-eff or neutron flux. Other network architectures which have been shown to extract features from images are recurrent neural networks and long-short-term-memory networks. These may have advantages to the CNN-based networks. Other advantages may be realized by increasing the hyper parameter search space for the CNN networks and upgrading the GPU to one or more with larger memory. More memory translates into larger networks and also faster training using more data.

Lastly, this work shows the potential promise of augmenting nuclear optimizations with both gradient informed and un-informed methods. These methods could be used to optimize other nuclear systems with similar complexity such as power reactor systems, radiation shielding and isotope production, among others. These types of optimizations are common and may benefit from the advanced optimization methods presented here.

Bibliography

[1] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *International conference on parallel problem solving from nature*, pages 849–858. Springer, 2000. xii, 12, 40, 41, 42, 44

[2] John Pevey, Briana Hiscox, Austin Williams, Ondřej Chvála, Vladimir Sobes, and J Wesley Hines. Gradient-informed design optimization of select nuclear systems. *Nuclear Science and Engineering*, pages 1–13, 2022. xiii, 107, 109, 110, 111, 112, 124

[3] Bradley T Rearden, Don Mueller, Stephen M Bowman, Robert D Busch, and Scott Emerson. Tsunami primer: A primer for sensitivity/uncertainty calculations with scale. Technical report, Oak Ridge National Laboratory (United States). Funding organisation: NNSA . . . , 2009. 1, 46

[4] JH Vainionpaa, CK Gary, JL Harris, MA Piestrup, RH Pantell, and G Jones. Technology and applications of neutron generators developed by adelphi technology, inc. *Physics Procedia*, 60:203–211, 2014. 2, 52

[5] David A Brown, MB Chadwick, R Capote, AC Kahler, A Trkov, MW Herman, AA Sonzogni, Y Danon, AD Carlson, M Dunn, et al. Endf/b-viii. 0: the 8th major release of the nuclear reaction data library with cielo-project cross sections, new standards and thermal scattering data. *Nuclear Data Sheets*, 148:1–142, 2018. 7

[6] AJM Plompen, O Cabellos, C De Saint Jean, M Fleming, A Algora, M Angelone, P Archier, E Bauge, O Bersillon, A Blokhin, et al. The joint evaluated fission and fusion nuclear data library, jeff-3.3. *The European Physical Journal A*, 56(7):1–108, 2020. 7

[7] Keiichi Shibata, Osamu Iwamoto, Tsuneo Nakagawa, Nobuyuki Iwamoto, Akira Ichihara, Satoshi Kunieda, Satoshi Chiba, Kazuyoshi Furutaka, Naohiko Otuka, Takaaki Ohsawa, et al. Jendl-4.0: a new library for nuclear science and engineering. *Journal of Nuclear Science and Technology*, 48(1):1–30, 2011. 7

[8] John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press, 1992. 11

[9] Seyedali Mirjalili. Genetic algorithm. In *Evolutionary algorithms and neural networks*, pages 43–55. Springer, 2019. 11

[10] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 12

[11] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. 13

[12] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015. 13

[13] Marvin Minsky and Seymour Papert. Perceptrons. 1969. 13

[14] Jürgen Schmidhuber. Who invented backpropagation. *More in [DL2]*, 2014. 15

[15] Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pages 762–770. Springer, 1982. 15

[16] Mirza Cilimkovic. Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, 15(1), 2015. 15

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 15

[18] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982. 16

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 16

[20] Alexis Conneau, Holger Schwenk, Loıc Barrault, and Yann Lecun. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*, 2(1), 2016. 16

[21] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. 16

[22] Akansha Kumar and Pavel V Tsvetkov. A new approach to nuclear reactor design optimization using genetic algorithms and regression analysis. *Annals of Nuclear Energy*, 85:27–35, 2015. 19

[23] Jorge Luiz C Chapot, Fernando Carvalho Da Silva, and Roberto Schirru. 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, 1999. 20

[24] Sandra Bogetic. *Improvements, Validation, and Applications of a Metaheuristic Optimization Method for Neutron Spectra Tailoring at the National Ignition Facility.* University of California, Berkeley, 2020. 21

[25] Zhenping Chen, Zhenyu Zhang, Jinsen Xie, Qian Guo, Tao Yu, Pengcheng Zhao, Zijing Liu, and Chao Xie. Multi-objective optimization strategies for radiation shielding design with genetic algorithm. *Computer Physics Communications*, 260:107267, 2021. 23

[26] Cláudio MNA Pereira and Celso MF Lapa. Coarse-grained parallel genetic algorithm applied to a nuclear reactor core design optimization problem. *Annals of Nuclear Energy*, 30(5):555–565, 2003. 24

[27] Susan Hogle. Optimization of transcurium isotope production in the high flux isotope reactor. 2012. 26

[28] Hans D Gougar, Abderrafi M Ougouag, William K Terry, and KN Ivanov. Automated design and optimization of pebble-bed reactor cores. *Nuclear Science and Engineering*, 165(3):245–269, 2010. 27

[29] Binh Quang Do and Lan Phuoc Nguyen. Application of a genetic algorithm to the fuel reload optimization for a research reactor. *Applied Mathematics and Computation*, 187(2):977–988, 2007. 29

[30] Song Hyun Kim, Sung Gyun Shin, Sangsoo Han, Moo Hwan Kim, and Cheol Ho Pyeon. Feasibility study on application of an artificial neural network for automatic design of a reactor core at the kyoto university critical assembly. *Progress in Nuclear Energy*, 119:103183, 2020. 30

[31] Jin young Lee. *Convolutional Neural Network for Prediction of Two-Dimensional Core Power Distributions in PWRs*. PhD thesis, Seoul National University, 2019. 34

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 34

[33] Qian Zhang, Jinchao Zhang, Liang Liang, Zhuo Li, and Tengfei Zhang. A deep learning based surrogate model for estimating the flux and power distribution solved by diffusion

equation. In *EPJ Web of Conferences*, volume 247, page 03013. EDP Sciences, 2021. 35

[34] Abdullah Konak, David W Coit, and Alice E Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability engineering & system safety*, 91(9):992–1007, 2006. 40

[35] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013. 43

[36] T. Goorley, M. James, T. Booth, F. Brown, J. Bull, L. J. Cox, J. Durkee, J. Elson, M. Fensin, R. A. Forster, J. Hendricks, H. G. Hughes, R. Johns, B. Kiedrowski, R. Martz, S. Mashnik, G. McKinney, D. Pelowitz, R. Prael, J. Sweezy, L. Waters, T. Wilcox, and T. Zukaitis. Initial mcnp6 release overview. *Nuclear Technology*, 180(3):298–315, 2012. 45

[37] Mark B Chadwick, Michal Herman, P Obložinskỳ, Michael E Dunn, Yaron Danon, AC Kahler, Donald L Smith, Boris Pritychenko, Goran Arbanas, R Arcilla, et al. Endf/b-vii. 1 nuclear data for science and technology: cross sections, covariances, fission product yields and decay data. *Nuclear data sheets*, 112(12):2887–2996, 2011. 45

[38] Steven M Bowman. Scale 6: comprehensive nuclear safety analysis code system. *Nuclear technology*, 174(2):126–148, 2011. 45

[39] Bradley T Rearden, Michael E Dunn, Dorothea Wiarda, Cihangir Celik, Kursat B Bekar, Mark L Williams, Douglas E Peplow, Christopher M Perfetti, Ian C Gauld, William A Wieselquist, et al. Overview of scale 6.2. *Proceedings of ANS NCSD 2013, Wilmington, North Carolina, September 29–October*, 3, 2013. 45

[40] Christopher M Perfetti, Bradley T Rearden, and William R Martin. Scale continuous-energy eigenvalue sensitivity coefficient calculations. *Nuclear Science and Engineering*, 182(3):332–353, 2016. 46

[41] Chris Sheridan. *The Python language reference manual*. Lulu Press, Inc, 2016. 47

[42] Mark Lutz. *Programming Python, Rev. 4*. O'Reilly Media, Inc, 2015. 47

[43] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 47

[44] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows.*, volume 2016. 2016. 47

[45] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4):575–601, 1992. 48, 107

[46] The Mathworks, Inc., Natick, Massachusetts. *MATLAB version 9.6 (R2019a)*, R2019. 48

[47] Thomas Lawrence Sterling. *Beowulf cluster computing with Linux*. MIT press, 2002. 65

[48] Matthias Feurer and Frank Hutter. Hyperparameter optimization. In *Automated machine learning*, pages 3–33. Springer, Cham, 2019. 72

[49] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012. 72

[50] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. 72

[51] Sharon Hoots and Don Wadsworth. Neutron and gamma dose and spectra measurements on the little boy replica. Technical report, Lawrence Livermore National Lab., CA (USA), 1984. 86

[52] John Pevey, Ondřej Chvála, Sarah Davis, Vladimir Sobes, and J Wes Hines. Genetic algorithm design of a coupled fast and thermal subcritical assembly. *Nuclear Technology*, 206(4):609–619, 2020. 93, 123

[53] John L Pevey, Cameron Salyer, Ondrej Chvala, Vladimir Sobes, and J Wesley Hines. Multi-objective design optimization of a fast spectrum nuclear experiment facility using artificial intelligence. *Annals of Nuclear Energy*, 162:108476, 2021. 93, 123

[54] Alessandro Alemberti, Valery Smirnov, Craig F Smith, and Minoru Takahashi. Overview of lead-cooled fast reactor activities. *Progress in Nuclear Energy*, 77:300–307, 2014. 93

Appendix A

Model Inputs

A.1 MCNP Model of FNS

```
c This template file is of 3 zones with 3 separate cassette
c types per zone.
c Cassette Definitions
c *1 is center *2 is cardinal directions *3 is diagonal
c Zone 1
c 411 0 -301 302 -303 304 u=11 lat=1 $ROW 1
c  fill=0:9 0:00 0:00
c
c Zone 1
 412 0 -301 302 -303 304 u=12 lat=1 $ROW 1
       fill=0:19 0:00 0:00
       2 4 4 2 4 2 2 2 2
       3 2 2 4 3 2 3 3 3 3 4
c Zone 1
 413 0 -301 302 -303 304 u=13 lat=1 $ROW 1
       fill=0:19 0:00 0:00
```

```
      2 4 2 3 2 3 4 2 4 4 3 2 4 4 3 2 2 4 4 3
c Zone 2
 421 0 -301 302 -303 304 u=21 lat=1 $ROW 1
      fill=0:12 0:00 0:00
      3 4 4 3 4 4 3 3 5 2 5 2 5
c Zone 2
 422 0 -301 302 -303 304 u=22 lat=1 $ROW 1
      fill=0:19 0:00 0:00
      3 2 3 4 2 3 4 3 3 2 2 4 4 3 2 3 4 4 2 2
c Zone 2
 423 0 -301 302 -303 304 u=23 lat=1 $ROW 1
      fill=0:19 0:00 0:00
      2 3 3 2 2 4 4 2 2 2 4 4 3 3 2 4 3 2 2 3
c Zone 3
c comment: Center cell is unused in this model and is
c commented out
c 431 0 -301 302 -303 304 u=31 lat=1 $ROW 1
c fill=0:19 0:00 0:00
c $$$zone_33$$$
 432 0 -301 302 -303 304 u=32 lat=1 $ROW 1
      fill=0:19 0:00 0:00
      4 3 4 3 2 3 2 4 2 2 3 3 2 4 4 2 2 3 3 3
 433 0 -301 302 -303 304 u=33 lat=1 $ROW 1
      fill=0:19 0:00 0:00
      2 4 2 2 2 4 3 3 2 4 4 4 3 2 2 4 3 3 4 2
c Fast zone moderator cassette     ***************
  161  0 -301 302 -303 304  u=99 lat=1 $ROW 1
           fill=0:39 0:0 0:0
      4 4 4 4 4
      4 4 4 4 4
      4 4 4 4 4
      4 4 4 4 4
      4 4 4 4 4
      4 4 4 4 4
      4 4 4 4 4
      4 4 4 4 4
c Thermal full coolant     ***************
  163     0 -301 302 -303 304   u=98  lat=1 $ROW 1
            fill=0:39  0:0  0:0
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
c Plate definitions
c Zone 1 Plates
c ------------------------------$Void$---------------------
300 0           10   -11 u=1
c ------------------------------$Poly$---------------------
320 1    -0.93 10   -11 u=2
c ------------------------------$Natural Uranium$----------
340 13   -18.95 10   -11 u=3
c ------------------------------$Other Moderator$----------
```

```
360 3   −0.971 10   −11 u=4
c ————————————————————————$Enriched Fuel$——————————
360 2   −18.95 10   −11 u=5
c ——————————————————————————————CASSETTES——————————
c cassette box
    12      5   −2.7 1 −2  $ Cassette material in original case.
c Cassette: ('Zone, Row, Cassette', 1, 1, 1)
   13     6     −8.05 −501
 c fueled cassette innards
    1       0           −1         fill=99
c Cassette: ('Zone, Row, Cassette', 1, 1, 3)
   14       5   −2.7 1 −2  trcl=(0 16.002 0 )
   15       0           −1  trcl=(0 16.002 0 ) fill=99 $5
c Cassette: ('Zone, Row, Cassette', 1, 1, 4)
   16       5   −2.7 1 −2  trcl=(0 32.004 0 )
   17       0           −1  trcl=(0 32.004 0 ) fill=99
c Cassette: ('Zone, Row, Cassette', 1, 1, 5)
   18     6     −8.05 −502
c
c Cassette: ('Zone, Row, Cassette', 1, 2, 1)
   19       5   −2.7 1 −2  trcl=(0 −16.002 15.875 )
   20       0           −1  trcl=(0 −16.002 15.875 ) fill=99 $1
c Cassette: ('Zone, Row, Cassette', 1, 2, 2)
   21       5   −2.7 1 −2  trcl=(0 0 15.875 )
   22       0           −1  trcl=(0 0 15.875 ) fill=13 $5
c Cassette: ('Zone, Row, Cassette', 1, 2, 3)
   23       5   −2.7 1 −2  trcl=(0 16.002 15.875 )
   24       0           −1  trcl=(0 16.002 15.875 ) fill=12 $5
```

```
c Cassette: ('Zone, Row, Cassette', 1, 2, 4)
   25       5   −2.7 1 −2   trcl=(0 32.004 15.875 )
   26       0          −1   trcl=(0 32.004 15.875 ) fill=13 $5
c Cassette: ('Zone, Row, Cassette', 1, 2, 5)
   27       5   −2.7 1 −2   trcl=(0 48.006 15.875 )
   28       0          −1   trcl=(0 48.006 15.875 ) fill=99
c
c Cassette: ('Zone, Row, Cassette', 1, 3, 1)
   29       5   −2.7 1 −2   trcl=(0 −16.002 31.75 )
   30       0          −1   trcl=(0 −16.002 31.75 ) fill=99 $5
c Cassette: ('Zone, Row, Cassette', 1, 3, 2)
   31       5   −2.7 1 −2   trcl=(0 0 31.75 )
   32       0          −1   trcl=(0 0 31.75 ) fill=12 $5
c Cassette: ('Zone, Row, Cassette', 1, 3, 3)
c Cassette: ('Zone, Row, Cassette', 1, 3, 4)
   33       5   −2.7 1 −2   trcl=(0 32.004 31.75 )
   34       0          −1   trcl=(0 32.004 31.75 ) fill=12 $5
c Cassette: ('Zone, Row, Cassette', 1, 3, 5)
   35       5   −2.7 1 −2   trcl=(0 48.006 31.75 )
   36       0          −1   trcl=(0 48.006 31.75 ) fill=99
c
c Cassette: ('Zone, Row, Cassette', 1, 4, 1)
   37       5   −2.7 1 −2   trcl=(0 −16.002 47.625 )
   38       0          −1   trcl=(0 −16.002 47.625 ) fill=99
c Cassette: ('Zone, Row, Cassette', 1, 4, 2)
   39       5   −2.7 1 −2   trcl=(0 0 47.625 )
   40       0          −1   trcl=(0 0 47.625 ) fill=13 $5
c Cassette: ('Zone, Row, Cassette', 1, 4, 3)
```

```
    41         5    −2.7  1 −2   trcl=(0  16.002  47.625  )
    42         0              −1   trcl=(0  16.002  47.625  )  fill=12  $5
c  Cassette:  ('Zone,  Row,  Cassette',  1,  4,  4)
    43         5    −2.7  1 −2   trcl=(0  32.004  47.625  )
    44         0              −1   trcl=(0  32.004  47.625  )  fill=13  $5
c  Cassette:  ('Zone,  Row,  Cassette',  1,  4,  5)
    45         5    −2.7  1 −2   trcl=(0  48.006  47.625  )
    46         0              −1   trcl=(0  48.006  47.625  )  fill=99
c
c  Cassette:  ('Zone,  Row,  Cassette',  1,  5,  1)
    47       6      −8.05  −503
c  Cassette:  ('Zone,  Row,  Cassette',  1,  5,  2)
    48         5    −2.7  1 −2   trcl=(0  0  63.5  )
    49         0              −1   trcl=(0  0  63.5  )  fill=99
c  Cassette:  ('Zone,  Row,  Cassette',  1,  5,  3)
    50         5    −2.7  1 −2   trcl=(0  16.002  63.5  )
    51         0              −1   trcl=(0  16.002  63.5  )  fill=99
c  Cassette:  ('Zone,  Row,  Cassette',  1,  5,  4)
    52         5    −2.7  1 −2   trcl=(0  32.004  63.5  )
    53         0              −1   trcl=(0  32.004  63.5  )  fill=99
c  Cassette:  ('Zone,  Row,  Cassette',  1,  5,  5)
    54       6      −8.05  −504
c
c  Zone  2
c
c  Cassette:  ('Zone,  Row,  Cassette',  2,  1,  1)
    55       6      −8.05  −505
c  Cassette:  ('Zone,  Row,  Cassette',  2,  1,  2)
```

```
    56      5    −2.7  1 −2   trcl=(26.682  0  0  )
    57      0              −1   trcl=(26.682  0  0  )  fill=99
c  Cassette: ('Zone, Row,  Cassette', 2, 1, 3)
    58      5    −2.7  1 −2   trcl=(26.682  16.002  0  )
    59      0              −1   trcl=(26.682  16.002  0  )  fill=99
c  Cassette: ('Zone, Row,  Cassette', 2, 1, 4)
    60      5    −2.7  1 −2   trcl=(26.682  32.004  0  )
    61      0              −1   trcl=(26.682  32.004  0  )  fill=99
c  Cassette: ('Zone, Row,  Cassette', 2, 1, 5)
    62     6     −8.05  −506
c
c  Cassette: ('Zone, Row,  Cassette', 2, 2, 1)
    63      5    −2.7  1 −2   trcl=(26.682  −16.002  15.875  )
    64      0              −1   trcl=(26.682  −16.002  15.875  )  fill=99
c  Cassette: ('Zone, Row,  Cassette', 2, 2, 2)
    65      5    −2.7  1 −2   trcl=(26.682  0  15.875  )
    66      0              −1   trcl=(26.682  0  15.875  )  fill=23
c  Cassette: ('Zone, Row,  Cassette', 2, 2, 3)
    67      5    −2.7  1 −2   trcl=(26.682  16.002  15.875  )
    68      0              −1   trcl=(26.682  16.002  15.875  )  fill=22
c  Cassette: ('Zone, Row,  Cassette', 2, 2, 4)
    69      5    −2.7  1 −2   trcl=(26.682  32.004  15.875  )
    70      0              −1   trcl=(26.682  32.004  15.875  )  fill=23
c  Cassette: ('Zone, Row,  Cassette', 2, 2, 5)
    71      5    −2.7  1 −2   trcl=(26.682  48.006  15.875  )
    72      0              −1   trcl=(26.682  48.006  15.875  )  fill=99
c  Cassette: ('Zone, Row,  Cassette', 2, 3, 1)
    73      5    −2.7  1 −2   trcl=(26.682  −16.002  31.75  )
```

```
   74      0          −1  trcl=(26.682 −16.002 31.75 ) fill=99
c Cassette: ('Zone, Row, Cassette', 2, 3, 2)
   75      5  −2.7 1 −2  trcl=(26.682 0 31.75 )
   76      0          −1  trcl=(26.682 0 31.75 ) fill=22
c Cassette: ('Zone, Row, Cassette', 2, 3, 3)
77 5 −2.7 5 −6 trcl=( 35.5719999999999 16.002 31.75 )
78 0 −5 trcl=(  35.5719999999999 16.002 31.75 ) fill=21
c Cassette: ('Zone, Row, Cassette', 2, 3, 4)
   79      5  −2.7 1 −2  trcl=(26.682 32.004 31.75 )
   80      0          −1  trcl=(26.682 32.004 31.75 ) fill=22
c Cassette: ('Zone, Row, Cassette', 2, 3, 5)
   81      5  −2.7 1 −2  trcl=(26.682 48.006 31.75 )
   82      0          −1  trcl=(26.682 48.006 31.75 ) fill=99
c Cassette: ('Zone, Row, Cassette', 2, 4, 1)
   83      5  −2.7 1 −2  trcl=(26.682 −16.002 47.625 )
   84      0          −1  trcl=(26.682 −16.002 47.625 ) fill=99
c Cassette: ('Zone, Row, Cassette', 2, 4, 2)
   85      5  −2.7 1 −2  trcl=(26.682 0 47.625 )
   86      0          −1  trcl=(26.682 0 47.625 ) fill=23
c Cassette: ('Zone, Row, Cassette', 2, 4, 3)
   87      5  −2.7 1 −2  trcl=(26.682 16.002 47.625 )
   88      0          −1  trcl=(26.682 16.002 47.625 ) fill=22
c Cassette: ('Zone, Row, Cassette', 2, 4, 4)
   89      5  −2.7 1 −2  trcl=(26.682 32.004 47.625 )
   90      0          −1  trcl=(26.682 32.004 47.625 ) fill=23
c Cassette: ('Zone, Row, Cassette', 2, 4, 5)
   91      5  −2.7 1 −2  trcl=(26.682 48.006 47.625 )
   92      0          −1  trcl=(26.682 48.006 47.625 ) fill=99
```

```
c Cassette: ('Zone, Row, Cassette', 2, 5, 1)
   93     6    -8.05 -507
c Cassette: ('Zone, Row, Cassette', 2, 5, 2)
   94      5   -2.7 1 -2   trcl=(26.682 0 63.5 )
   95      0           -1   trcl=(26.682 0 63.5 ) fill=99
c Cassette: ('Zone, Row, Cassette', 2, 5, 3)
   96      5   -2.7 1 -2   trcl=(26.682 16.002 63.5 )
   97      0           -1   trcl=(26.682 16.002 63.5 ) fill=99
c Cassette: ('Zone, Row, Cassette', 2, 5, 4)
   98      5   -2.7 1 -2   trcl=(26.682 32.004 63.5 )
   99      0           -1   trcl=(26.682 32.004 63.5 ) fill=99
c Cassette: ('Zone, Row, Cassette', 2, 5, 5)
  100     6    -8.05 -508
c
c Zone 3
c
c Cassette: ('Zone, Row, Cassette', 3, 1, 1)
  101     6    -8.05 -509
c Cassette: ('Zone, Row, Cassette', 3, 1, 2)
  102      5   -2.7 1 -2   trcl=(53.35230     0 0 )
  103      0           -1   trcl=(53.35230     0 0 ) fill=98 $1
c Cassette: ('Zone, Row, Cassette', 3, 1, 3)
  104      5   -2.7 1 -2   trcl=(53.35230     16.002 0 )
  105      0           -1   trcl=(53.35230     16.002 0 ) fill=98
c Cassette: ('Zone, Row, Cassette', 3, 1, 4)
  106      5   -2.7 1 -2   trcl=(53.35230     32.004 0 )
  107      0           -1   trcl=(53.35230     32.004 0 ) fill=98 $6
c Cassette: ('Zone, Row, Cassette', 3, 1, 5)
```

```
  108      6     -8.05  -510
c
c Cassette: ('Zone, Row, Cassette', 3, 2, 1)
  109       5    -2.7  1 -2   trcl=(53.35230     -16.002  15.875  )
  110       0             -1   trcl=(53.35230     -16.002  15.875  )
          fill=98  $1
c Cassette: ('Zone, Row, Cassette', 3, 2, 2)
  111       5    -2.7  1 -2   trcl=(53.35230     0  15.875  )
  112       0             -1   trcl=(53.35230     0  15.875  )  fill=33
c Cassette: ('Zone, Row, Cassette', 3, 2, 3)
  113       5    -2.7  1 -2   trcl=(53.35230     16.002  15.875  )
  114       0             -1   trcl=(53.35230     16.002  15.875  )  fill=32
c Cassette: ('Zone, Row, Cassette', 3, 2, 4)
  115       5    -2.7  1 -2   trcl=(53.35230     32.004  15.875  )
  116       0             -1   trcl=(53.35230     32.004  15.875  )  fill=33
c Cassette: ('Zone, Row, Cassette', 3, 2, 5)
  117       5    -2.7  1 -2   trcl=(53.35230     48.006  15.875  )
  118       0             -1   trcl=(53.35230     48.006  15.875  )
          fill=98  $6
c
c Cassette: ('Zone, Row, Cassette', 3, 3, 1)
  119       5    -2.7  1 -2   trcl=(53.35230     -16.002  31.75  )
  120       0             -1   trcl=(53.35230     -16.002  31.75  )  fill=98
c Cassette: ('Zone, Row, Cassette', 3, 3, 2)
  121       5    -2.7  1 -2   trcl=(53.35230     0  31.75  )
  122       0             -1   trcl=(53.35230     0  31.75  )  fill=32
c Cassette: ('Zone, Row, Cassette', 3, 3, 3)
c   123       0             -511
```

```
c Cassette: ('Zone, Row, Cassette', 3, 3, 4)
  124      5   −2.7 1 −2  trcl=(53.35230    32.004 31.75 )
  125      0          −1  trcl=(53.35230    32.004 31.75 ) fill=32
c Cassette: ('Zone, Row, Cassette', 3, 3, 5)
  126      5   −2.7 1 −2  trcl=(53.35230    48.006 31.75 )
  127      0          −1  trcl=(53.35230    48.006 31.75 ) fill=98
c Cassette: ('Zone, Row, Cassette', 3, 4, 1)
  128      5   −2.7 1 −2  trcl=(53.35230    −16.002 47.625 )
  129      0          −1  trcl=(53.35230    −16.002 47.625 )
        fill=98 $7
c Cassette: ('Zone, Row, Cassette', 3, 4, 2)
  130      5   −2.7 1 −2  trcl=(53.35230    0 47.625 )
  131      0          −1  trcl=(53.35230    0 47.625 ) fill=33
c Cassette: ('Zone, Row, Cassette', 3, 4, 3)
  132      5   −2.7 1 −2  trcl=(53.35230    16.002 47.625 )
  133      0          −1  trcl=(53.35230    16.002 47.625 ) fill=32
c Cassette: ('Zone, Row, Cassette', 3, 4, 4)
  134      5   −2.7 1 −2  trcl=(53.35230    32.004 47.625 )
  135      0          −1  trcl=(53.35230    32.004 47.625 ) fill=33
c Cassette: ('Zone, Row, Cassette', 3, 4, 5)
  136      5   −2.7 1 −2  trcl=(53.35230    48.006 47.625 )
  137      0          −1  trcl=(53.35230    48.006 47.625 )
        fill=98 $8
c Cassette: ('Zone, Row, Cassette', 3, 5, 1)
  138     6    −8.05 −512
c Cassette: ('Zone, Row, Cassette', 3, 5, 2)
  139      5   −2.7 1 −2  trcl=(53.35230    0 63.5 )
  140      0          −1  trcl=(53.35230    0 63.5 ) fill=98 $7
```

```
c Cassette: ('Zone, Row, Cassette', 3, 5, 3)
  141      5   -2.7 1 -2  trcl=(53.35230    16.002 63.5 )
  142      0          -1  trcl=(53.35230    16.002 63.5 )
       fill=98
c Cassette: ('Zone, Row, Cassette', 3, 5, 4)
  143      5   -2.7 1 -2  trcl=(53.35230    32.004 63.5 )
  144      0          -1  trcl=(53.35230    32.004 63.5 )
       fill=98 $8
c Cassette: ('Zone, Row, Cassette', 3, 5, 5)
  145     6    -8.05 -513
c Cassette: Half cassette in fast zone 1
c 148 5 -2.7 3 -4 305 #150 trcl=( 38.862 16.002 31.75 )
c 149 0 -3 -305 #150 trcl=(  40.57628571 16.002 31.75 ) fill=11
c  150      5   -2.7 -306
c Cutout for Cassette patterns A, exp. vol and source
  200 0 -7 #998 #77 #78
  999      0          -500 -400 2 #13 #14 #15 #16 #17 #18 #19 #20
            #21 #22 #23 #24 #25 #26 #27
            #28 #29 #30 #31 #32 #33 #34 #35 #36 #37
            #38 #39 #40 #41 #42 #43 #44 #45 #46 #47
            #48 #49 #50 #51 #52 #53 #54
              7
 1000      0          -500 400 -401 #55 #56 #57 #58 #59
            #60 #61 #62 #155 #63 #64
            #65 #66 #67 #68 #69 #70 #71
            #72 #73 #74 #75 #76  #79 #80 #81
            #82 #83 #84 #85 #86 #87 #88 #89
            #90 #91 #92 #93 #94 #95 #96 #97 #98
```

```
                #99 #100 7
 1001      0          -500 401 #101 #102
                #103 #104 #105 #106 #107 #108
                #156 #109 #110 #111 #112 #113
                #114 #115 #116 #117 #118 #119
                #120 #121 #122 #124 #125 #126
                #127 #128 #129 #130 #131 #132 #133 #134
                #135 #136 #137 #138 #139 #140 #141 #142
                #143 #144 #145 7
  146      6     -8.05 402 403 500 -514 -401
c reflector on thermal side:
  157      6     -8.05 401 -514 402 403 500
  c shutdown rods 155, 156 boron: 8    -2.52
  155     0          -402 7
  156     0   -403 7 $ 8 -2.52
  c
  153     0 -515
c Exp. volume
  998 0 -516
c outside bounds of model:
  147      0          514 515

c SURFACE CARD
c All cassettes other than cassette 2A
    1         rpp -1.27 24.13 -1.27 13.97 -1.27 13.97
    2         rpp -1.5875 24.4475 -1.5875 14.2875 -1.5875 14.2875
c Variable Cassette A Zone 2
5 rpp -1.27 15.240000000000002 -1.27 13.97 -1.27 13.97
```

```
6 rpp −1.5875 15.557500000000003 −1.5875 14.2875 −1.5875 14.2875
c Inner FNS Volume (Around Cassette A, Exp. Vol and Source)
   7        rpp −2.8575 79.1456 14.4145 30.2895 30.1625 46.0375
c experiment volume
  516 rpp 18.744499999999998 33.984499999999998
       14.4145 30.2895 30.1625 46.0375
c
   10          px −1.2701
   11          px 0.00000000001
  301          px 0
  302          px −1.27
  303          py 100
  304          py −100
c These split the core into 3 parts, MCNP definition work−around
  400          px 24.45
  401          px 51.1297
c the shutdown rods     Fast/Fast and Fast/Thermal
  402   rpp 24.451 25.086 −21.5265 65.9765 −6.1275 86.1375
  403   rpp 51.1298 51.76479 −21.5265 65.9765 −6.1275 86.1375
c inner void of assembly,
c      the cutout for the entire assembly within the steel
  500        rpp −2.8575 79.1456 −18.9865 63.4365 −1.5875 89.2175
c
  501   rpp −1.5875 24.45 −17.5895 −1.7145 −1.5875 14.2875
  502   rpp −1.5875 24.45 46.4185 62.2935 −1.5875 14.2875
  503   rpp −1.5875 24.45 −17.5895 −1.7145 61.9125 77.7875
  504   rpp −1.5875 24.45 46.4185 62.2935 61.9125 77.7875
  505   rpp 25.086 51.1296 −17.5895 −1.7145 −1.5875 14.2875
  506    rpp 25.086 51.1296 46.4185 62.2935 −1.5875 14.2875
  507    rpp 25.086 51.1296 −17.5895 −1.7145 61.9125 77.7875
  508    rpp 25.086 51.1296 46.4185 62.2935 61.9125 77.7875
  509    rpp 51.7648 77.7988 −17.5895 −1.7145 −1.5875 14.2875
  510    rpp 51.7648 77.7988 46.4185 62.2935 −1.5875 14.2875
  511    rpp 51.7648 77.7988 14.6685 30.3 30.1625 45.9
  512    rpp 51.7648 77.7988 −17.5895 −1.7145 61.9125 77.7875
  513    rpp 51.7648 77.7988 46.4185 62.2935 61.9125 77.7875
c outer steel reflector dims
  514     rpp −28.2575 104.5456 −44.3865 88.8365 −11.7475 99.3775
c concrete pedestal
  515          rpp −43.4975 121.2228 −59.6265 104.0765
         −72.7075 −11.7475

c DATA CARD
c ————————————————SOURCE INFORMATION——
kcode 5000 1.000000 20 120 45000
ksrc    27.7246 22.6372 30.0729
c ————————————————MATERIALS——————
c Polyethylene, PNNL Doc −0.93 g/cc
m1      1001.70c −0.143716
        6000.70c −0.856284
MT1 poly.10
c FNS 9.75% Enrich. Fuel, −18.94 g/cc
m2      92232.70c −0.000000002
        92234.70c −0.0026
        92235.70c −0.0975
        92236.70c −0.0046
        92238.70c  −0.895299998
c Lead 11.35 pb
m3      82204.70c              −0.014   $
        82206.70c              −0.241  82207.70c  −0.221  82208.70c  −0.524
c Aluminum −2.7 g/cc
m5      13027.70c                   1
c Carbon Steel
m6      6000.70c          0.022831
        26054.70c         0.057164445
        26056.70c         0.896553475
        26057.70c         0.020716004
        26058.70c         0.002736076
c  Regular Concrete per pnnl pdf doc     −2.3 g/cc
m7      1001.70c         0.168038
        8016.70c         0.563183
        11023.70c        0.021365
        13027.70c        0.021343
        14028.70c        0.187378982
        14029.70c        0.009551857
        14030.70c        0.006300161
        20040.70c        0.018026179
        20042.70c        0.00012031
        20043.70c        2.51033E−05
        20044.70c        0.000387892
        20046.70c        7.438E−07
        20048.70c        3.47727E−05
        26054.70c        0.000248391
        26056.70c        0.003895705
```

```
      26057.70c       9.00152E-05
      26058.70c       1.18888E-05
c Natural Uranium, -18.95 g/cc
m13   92234.70c          -5.7e-005
      92235.70c          -0.007204
      92238.70c          -0.992739
c -------------------------------------------------OTHER---
c ---------------------------
imp:n   1 156r        0
c -------------------------------------------------TALLY
c DESC-------------------------
rand hist 865
c print
```

A.2 SCALE Model

```
=csas5 parm=()
fns geometry, half cassette in zone 1
ce_v7
read composition
' Polyethylene (PNNL Doc)
 h-poly     1 0 7.9863E-02  293   end
 c          1 0 3.9927E-02  293   end
' 9.75% Fuel
 u-232      2 0 9.8362E-11  293   end
 u-234      2 0 1.2678E-04  293   end
 u-235      2 0 4.7338E-03  293   end
 u-236      2 0 2.2239E-04  293   end
 u-238        2 0 4.2919E-02   293     end
' Sodium
 na-23        3 0 2.54348E-02    293     end
' Almunium
 al-27 4 0    6.02616E-02 293              end
' SS
 c            5 0   2.0181E-03 293     end
 fe-54        5 0   5.0528E-03 293     end
 fe-56        5 0   7.9247E-02 293     end
 fe-57        5 0   1.8311E-03 293     end
 fe-58        5 0   2.4184E-04 293     end
end composition
read parameter
 npg=10000
 htm=no
 plt=no
 sig=0.0010
end parameter
read geometry
unit 1
com='plate #1'
 cuboid 0 1  1.27  0  15.24  0        15.24            0
unit 2
com='plate #2'
 cuboid 1 1  1.27  0  15.24  0        15.24            0
unit 3
com='plate #3'
 cuboid 2 1  1.27  0  15.24  0        15.24            0
```

```
unit 4
com='plate #4'
 cuboid 3 1   1.27   0   15.24   0         15.24            0
unit 10
com='pattern 1A'
array 1 12.7 0 0
cuboid 4 1        25.7175 12.382  15.5575 −0.3175 15.5575 −0.3175
cuboid  0         1         25.7175 12.382  15.621   −0.381
    15.5575       −0.3175
cuboid  0         1         26.3525 −0.3175 15.621   −0.381
    15.5575       −0.3175
unit 11
com='pattern 1B'
array 2 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 26.3525        −0.3175 15.621   −0.381   15.5575 −0.3175
unit 12
com='pattern 1C'
array 3 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 26.3525        −0.3175 15.621   −0.381   15.5575 −0.3175
unit 13
com='pattern 1D'
array 4 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 26.3525        −0.3175 15.621   −0.381   15.5575 −0.3175
unit 14
com='pattern 1E'
```

```
array 5 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 26.3525       −0.3175 15.621   −0.381   15.5575 −0.3175
unit 15
com='pattern 1F'
array 6 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 26.3525       −0.3175 15.621   −0.381   15.5575 −0.3175
unit 16
com='pattern 2A'
array 7 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 26.3525       −0.3175 15.621   −0.381   15.5575 −0.3175
unit 17
com='pattern 2B'
array 8 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 26.3525       −0.3175 15.621   −0.381   15.5575 −0.3175
unit 18
com='pattern 2C'
array 9 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 26.3525       −0.3175 15.621   −0.381   15.5575 −0.3175
unit 19
com='pattern 2D'
array 10 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 26.3525       −0.3175 15.621   −0.381   15.5575 −0.3175
```

```
unit 20
com='pattern 2E'
array 11 0 0 0
cuboid 4 1 25.7175     −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 26.3525 −0.3175 15.621 −0.381 15.5575 −0.3175
unit 21
com='pattern 2F'
array 12 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 26.3525      −0.3175 15.621   −0.381   15.5575 −0.3175
unit 22
com='pattern 3A'
cuboid 0 1 25.7175      −0.3175 15.621   −0.381   15.5575 −0.3175
unit 23
com='pattern 3B'
array 13 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 25.7175      −0.3175 15.621   −0.381   15.5575 −0.3175
unit 24
com='pattern 3C'
array 14 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 25.7175      −0.3175 15.621   −0.381   15.5575 −0.3175
unit 25
com='pattern 3D'
array 15 0 0 0
cuboid 4 1 25.7175 −0.3175 15.5575 −0.3175 15.5575 −0.3175
cuboid 0 1 25.7175      −0.3175 15.621   −0.381   15.5575 −0.3175
```

```
unit 26
com='pattern 3E'
array 16 0 0 0
cuboid 4 1 25.7175 -0.3175 15.5575 -0.3175 15.5575 -0.3175
cuboid 0 1 25.7175       -0.3175 15.621   -0.381   15.5575 -0.3175
unit 27
com='pattern 3F'
array 17 0 0 0
cuboid 4 1 25.7175 -0.3175 15.5575 -0.3175 15.5575 -0.3175
cuboid 0 1 25.7175       -0.3175 15.621   -0.381   15.5575 -0.3175
unit 28
com='zone 1'
array 18 0 0 0
unit 29
com='zone 1'
array 19 0 0 0
unit 30
com='zone 1'
array 20 0 0 0
global unit 31
cuboid 0 1 80.7331 -1.27 81.153 -1.27 90.4875 0
hole 28 0 0 0
hole 29 26.67 0 0
hole 30 53.34 0 0
cuboid 5 1 106.1331 -26.67 106.553 -26.67 100.6475 -10.4775
end geometry
read array
ara= 1 nux=10 nuy=1 nuz=1
```

```
com='cassette 1A 10x1 plates'
fill
1 1 1 1 1 1 1 1 1 1
end fill
ara= 2 nux=20 nuy=1 nuz=1
com='cassette 1B 20x1 plates'
fill
4 4 4 4 4 3 4 4 4 4
4 4 4 4 4 4 4 4 4 4
end fill
ara= 3 nux=20 nuy=1 nuz=1
com='cassette 1C 20x1 plates'
fill
3 3 3 4 4 3 3 3 4 4
4 4 3 3 4 4 3 3 4 4
end fill
ara= 4 nux=20 nuy=1 nuz=1
com='cassette 1D 20x1 plates'
fill
4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
end fill
ara= 5 nux=20 nuy=1 nuz=1
com='cassette 1E 20x1 plates'
fill
4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
end fill
```

```
ara= 6 nux=20 nuy=1 nuz=1
com='cassette 1F 20x1 plates'
fill
4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
end fill
ara= 7 nux=20 nuy=1 nuz=1
com='cassette 2A 20x1 plates'
fill
1 3 3 4 3 3 3 3 3 3
3 3 3 3 3 4 3 3 3 3
end fill
ara= 8 nux=20 nuy=1 nuz=1
com='cassette 2B 20x1 plates'
fill
4 4 4 4 4 4 4 4 4 3
3 4 3 4 4 3 4 3 3 3
end fill
ara= 9 nux=20 nuy=1 nuz=1
com='cassette 2C 20x1 plates'
fill
3 4 4 3 3 4 4 3 3 4
3 4 4 4 4 3 3 3 3 3
end fill
ara= 10 nux=20 nuy=1 nuz=1
com='cassette 2D 20x1 plates'
fill
4 4 4 4 4 4 4 4 4 4
```

```
4 4 4 4 4 4 4 4 4 4
end fill
ara= 11 nux=20 nuy=1 nuz=1
com='cassette 2E 20x1 plates'
fill
4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
end fill
ara= 12 nux=20 nuy=1 nuz=1
com='cassette 2F 20x1 plates'
fill
4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
end fill
ara= 13 nux=20 nuy=1 nuz=1
com='cassette 3B 20x1 plates'
fill
3 3 3 3 2 3 3 4 4 3
2 4 4 4 3 4 3 3 2 4
end fill
ara= 14 nux=20 nuy=1 nuz=1
com='cassette 3C 20x1 plates'
fill
3 4 3 2 4 4 2 3 4 4
4 4 4 4 2 4 3 4 2 3
end fill
ara= 15 nux=20 nuy=1 nuz=1
com='cassette 3D 20x1 plates'
```

```
 fill
2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2
end fill
ara= 16 nux=20 nuy=1 nuz=1
com='cassette 3E 20x1 plates'
 fill
2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2
end fill
ara= 17 nux=20 nuy=1 nuz=1
com='cassette 3F 20x1 plates'
 fill
2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2
end fill
ara= 18 nux=1 nuy=5 nuz=5
com='zone 1'
 fill
15      14      13      14      15
14      12      11      12      14
13      11      10      11      13
14      12      11      12      14
15      14      13      14      15
end fill
ara= 19 nux=1 nuy=5 nuz=5
com='zone 2'
 fill
```

```
21 20 19 20 21
20 18 17 18 20
19 17 16 17 19
20 18 17 18 20
21 20 19 20 21
end fill
ara= 20 nux=1 nuy=5 nuz=5
com='zone 3'
fill
27 26 25 26 27
26 24 23 24 26
25 23 22 23 25
26 24 23 24 26
27 26 25 26 27
end fill
end array
read bnds
 +xb=vacuum
 -xb=vacuum
 +yb=vacuum
 -yb=vacuum
 +zb=vacuum
 -zb=vacuum
   end bnds
end data
end
```

Vita

John Pevey was born on August 17, 1987 in Augusta, GA to parents Nancy and Ronald Pevey. After moving to Knoxville, TN, he attended Farragut High School and graduated in 2006. He attended Louisiana State University and graduated with a Bachelor's degree in Construction Management in 2011. In 2012 he attended the University of Tennessee and graduated with a Master's Degree in nuclear engineering in 2015 under Dr. Lawrence Miller.

He then moved to upstate New York to work at Knolls Atomic Power Laboratory. In 2017 John moved back to Knoxville, TN to pursue a Ph.D. in nuclear engineering under the guidance of Dr. J. Wesley Hines and Dr. Vladimir Sobes. He will graduate with his Ph.D. in December 2022.