FLUORIDE-SALT-COOLED HIGH-TEMPERATURE REACTOR DESIGN OPTIMIZATION WITH EVOLUTIONARY ALGORITHMS

BY

GWENDOLYN J.Y. CHEE

PRELIMINARY EXAMINATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Nuclear, Plasma, and Radiological Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2021

Urbana, Illinois

Doctoral Committee:

Assistant Professor Kathryn D. Huff, Chair
Research Scientist Madicken Munk
Associate Professor Tomasz Kozlowski
Professor James F. Stubbins
Research Assistant Professor Huy Trong Tran

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Increased global average surface temperatures, sea levels, and severe weather events caused by elevated Greenhouse Gas (GHG) concentrations show the substantial negative impact of climate change on natural and human systems [5]. Energy use and production contribute two-thirds of total GHG emissions [5]. Furthermore, as the human population increases and previously underdeveloped nations rapidly industrialize, global energy demand will continue to rise. Because energy generation technology selection profoundly impacts climate change, large scale emissions-free nuclear power deployment could significantly reduce GHG production but faces both cost and perceived adverse safety challenges [5, 65]. The nuclear power industry must overcome these challenges to ensure continued global use and expansion of nuclear energy technology to provide low-carbon electricity worldwide. The Generation IV International Forum aims to enhance the role of nuclear energy in our global energy ecosystem by leading and planning research and development to support a new and innovative Generation IV nuclear energy systems [30]. Generation IV nuclear systems target goals in four areas: sustainability, economics, safety and reliability, and proliferation resistance and physical protection [30]. Table 1.1 summarizes the goals in each area.

The Generation IV International Forum's methodology working groups developed an evaluation and selection methodology based on these goals and correspondingly selected six Generation IV systems: Gas-Cooled Fast Reactors (GFRs), Lead-Cooled Fast Reactors (LFRs), Molten Salt Reactors (MSRs), Sodium-Cooled Fast Reactors (SFRs), Supercritical-

Table 1.1: Goals of Generation IV Nuclear Systems [30, 7]

Area	**Goals**
Sustainability	- Have a positive impact on the environment through the displacement of polluting energy and transportation sources by nuclear electricity generation and nuclear-produced hydrogen - Promote long-term availability of nuclear fuel - Minimize volume, lifetime, and toxicity of nuclear waste
Economics	- Have a life cycle and energy production cost advantage over other energy sources - Reduce economic risk to nuclear projects by developing plants using innovative fabrication and construction techniques
Safety and Reliability	- Increase the use of robust designs, and inherent and transparent safety features that can be understood by non-experts - Enhance public confidence in the safety of nuclear energy
Proliferation Resistance and Physical Protection	- Provide continued effective proliferation resistance of nuclear energy systems through improved design features and other measures - Increase the robustness of new facilities

Water-Cooled Reactors (SCWRs), and Very-High-Temperature Reactors (VHTRs) [30]. This proposed work will consider the MSR and VHTR systems.

MSR systems produce fission power in a circulating molten salt fuel mixture. It has a closed fuel cycle tailored to the efficient utilization of plutonium and minor actinides. Molten fluoride salts have very low vapor pressure, which reduces stress on the system. MSR systems also incorporate inherent system safety with fail-safe drainage, passive cooling, and a low inventory of volatile fission products in the fuel. MSR systems closed fuel cycle and excellent waste burndown performance make it top-ranked in sustainability. MSR systems are top-ranked in sustainability because of their closed fuel cycle and excellent waste burndown performance. They rate well in safety and proliferation resistance and physical protection, due to their inherent safety features, and rate neutrally in economics because of their large number of subsystems [30].

VHTR systems use a once-through uranium cycle and leverage their high outlet temperature for high-temperature heat applications, such as hydrogen production. Graphite-moderated and helium-cooled, VHTRs use Tristructural Isotropic (TRISO) fuel which easily

withstands high burnup and temperature. VHTR systems' high hydrogen production efficiency, high safety and reliability, and inherent fuel and reactor safety features make it highly ranked in economics. An open fuel cycle results in VHTR systems' ranking well in proliferation resistance and physical protection and neutrally in sustainability [30].

In the proposed PhD work, I will explore the Fluoride-Salt-Cooled High-Temperature Reactor (FHR) concept, which combines the best aspects of MSR and VHTR technologies. FHRs use high-temperature coated-particle fuel (similar to the VHTRs) and a low-pressure liquid fluoride-salt coolant (similar to the MSRs) [22, 20]. The proposed work focuses on a prismatic FHR design with hexagonal fuel assemblies consisting of TRISO fuel particles embedded in planks, i.e., the Advanced High Temperature Reactor (AHTR) design. To address and further understand the technical challenges associated with the AHTR design, such as the *multiply heterogeneous* AHTR fuel, I will participate in the Organisation for Economic Co-operation and Development (OECD)-Nuclear Energy Agency (NEA)'s FHR benchmarking exercise.

In the next section of the proposed PhD work, I will explore the impact of additive manufacturing technology advancements on reactor geometry optimization, specifically for the AHTR design. In recent years, additive manufacturing technology, popularly known as '3D printing', has advanced and altered the manufacturing and design of engineered hardware [79]. The automotive and aircraft industries have successfully fabricated car and airplane components with key additive manufacturing technologies that are relevant to nuclear reactor core structures [54]. For example, Boeing successfully used additive manufacturing to reduce weight in the 787 Dreamliner [4] and SES-15 spacecraft [1]. The highly-regulated aerospace industry's successful additive manufacturing applications shows promise for the also highly-regulated nuclear industry. Using additive manufacturing to fabricate nuclear reactor components could drastically reduce cost and timelines, and increase safety and performance by tailoring local material properties and redesigning for optimal geometries [79].

With further advancement of additive manufacturing technologies, a reactor core could be 3D printed in the near future. Oak Ridge National Laboratory (ORNL) leads this initiative through the 2019 Transformational Challenge Reactor (TCR) Demonstration Program. The TCR program will leverage recent scientific achievements in advanced manufacturing, nuclear materials, machine learning, and computational modeling and simulation to build a microreactor. The program aims to design, manufacture, and operate a demonstration reactor by 2023 [84]. Applying additive manufacturing to nuclear reactor design will free complex reactor geometries from previous manufacturing constraints, opening the door for a re-examination of nuclear reactor optimization [81]. Optimization efforts towards classically manufactured nuclear reactors, and now 3D printed nuclear reactors have focused on parameters such as core radius, cylinder height, fuel enrichment, etc [81, 73, 45, 61]. Leveraging additive manufacturing technology enables us to surpass classical manufacturing constraints, such as straight fuel channels or homogenous fuel enrichment, and optimize for arbitrary geometries and parameters such as non-uniform channel shapes, and inhomogeneous fuel distribution throughout the core.

Multi-objective design problems, such as reactor design optimization, inevitably require a trade-off between desirable attributes [11, 78]. For example, the neutron economy and fuel enrichment trade-off in nuclear reactor design. A reactor design should maximize neutron economy and minimize fuel enrichment to reduce proliferation risk and cost. Conflicting objectives results in no *one* perfect solution but a *set* of equally optimal solutions [11]. Multi-objective problems are challenging to optimize; therefore, they cannot be handled by classical optimization methods, such as gradient methods, which may find local optima while missing the global optimum [69]. Evolutionary algorithms have proven successful in optimizing multi-objective problems [41], as they can find solutions at the global optimum [69]. They also take advantage of parallel systems for reduced computational cost. Genetic algorithms are the most frequently used evolutionary algorithms for solving multi-objective problems [11, 41]. Genetic algorithms imitate natural selection to evolve solutions by [69]:

1. maintaining a population of solutions

2. allowing fitter solutions to reproduce

3. letting lesser fit solutions die off, resulting in final solutions that are better than the previous generations

To enable the use of evolutionary algorithms for reactor design optimization, in this preliminary work I designed an optimization tool that uses the evolutionary algorithm optimization technique with nuclear transport and thermal-hydraulics software. In this preliminary work, I demonstrate using the optimization tool to maximize k_{eff} in an AHTR fuel slab by varying fuel distribution. In the proposed PhD scope, I will further demonstrate using evolutionary algorithms for multi-objective AHTR optimization of arbitrary geometries and fuel distribution, which are now possible with additive manufacturing technology.

1.2 Objectives

I developed the proposed PhD scope's main objectives based on further understanding the AHTR design through neutronics and thermal-hydraulics modeling, and leveraging artificial intelligence tools with validated nuclear transport and thermal-hydraulics software to create an open-source tool which enables nuclear reactor design optimization with arbitrary parameters. The proposed work's objectives are:

I Model the FHR with established nuclear transport software. To demonstrate success in modeling the FHR with nuclear transport before using the optimization tool, I will participate in the OECD NEA's FHR benchmark [2].

II Develop a tool that applies evolutionary algorithms to optimize nuclear reactor design. This tool will not reinvent the wheel—it will utilize a well-documented and validated open-source evolutionary algorithm Python package with established

nuclear transport and thermal-hydraulics software. This tool will run parallel on high-performance computing (HPC) machines, be open-source, and follow the rules for ensuring reproducibility, effectiveness, and usability [51, 58, 74].

III **Demonstrate optimization tool with a single objective AHTR design optimization problem**. I will demonstrate successful implementation of the optimization tool with a nuclear transport software by optimizing an AHTR fuel slab's fuel distribution for a single objective function.

IV **Model the FHR with established thermal-hydraulics software**. To demonstrate success in modeling the FHR with thermal hydraulics before using the optimization tool, I will participate in the OECD NEA's FHR benchmark [2].

V **Demonstrate multi-objective nuclear reactor design optimizations for neutronics and thermal-hydraulics problems**. I will demonstrate successful use of the optimization tool for multi-objective AHTR optimization. I will vary parameters such as reactor geometry and fuel distribution with objectives such as maximizing k_{eff}, maximizing heat transfer, and minimizing power peaking factor.

This preliminary work completes objectives I, II, and III. These objectives provide the basis and proof of concept for objectives IV and V.

1.3 Outline

This document outlines the motivation, preliminary work, and future work proposed toward developing an open-source reactor evolutionary algorithm optimization tool to optimize nuclear reactor design beyond classical parameters. Chapter 1 describes the motivation and objectives of the proposed work. Chapter 2 presents a literature review that organizes and reports on previous relevant work. Chapter 3 describes the FHR benchmark specifications

and the results obtained thus far. Chapter 4 details the computational design of the developed optimization tool. Chapter 5 demonstrates nuclear reactor optimization with the optimization tool. Chapter 6 summarizes the remaining future work.

Chapter 2

Literature Review

This chapter provides a literature review of relevant past research efforts giving context to this proposed work. Recent advancements in additive manufacturing applications for nuclear reactor core components have removed traditional manufacturing constraints on reactor design, enabling reactor designers to reexamine optimization. The proposed work aims to provide a fresh perspective to nuclear reactor optimization by applying evolutionary algorithm methods to explore non-conventional reactor geometries and fuel distributions. With the many benefits of FHRs and growing interest in the nuclear science community with FHR designs, I chose to apply the optimization methods to this reactor type, and also participate in the Organisation for Economic Co-operation and Development (OECD) Nuclear Energy Agency's (NEA) FHR benchmarking exercise. Thus, I begin this literature review with an overview of the FHR concept, then go into detail about one specific FHR design: the AHTR, previous efforts and technical challenges of modeling the design, and a description of how these efforts led to the OECD NEA's initiation of the AHTR benchmark. Next, I outline additive manufacturing's history and describe the current research towards applying additive manufacturing to the fabrication of nuclear reactor core components. I also review previous efforts towards nuclear reactor design optimization, describe how additive manufacturing of nuclear reactor components enables optimization for less constrained reactor geometries, and types of optimization methods that can be leveraged in this expanded design space. Finally, I give a background of evolutionary algorithms and detail a specific evolutionary algorithm: the genetic algorithm and how it works to conduct global optimization robustly.

2.1 Fluoride-Salt-Cooled High-Temperature Reactor

The FHR concept introduced in 2003 uses high-temperature coated-particle fuel and a low-pressure liquid fluoride-salt coolant [23, 20]. However, the term Fluoride-Salt-Cooled High-Temperature Reactor was only introduced in 2010 to distinguish fluoride salt-cooled MSRs from other MSRs. FHR technology combines the best aspects of MSR and VHTR (or High Temperature Gas-Cooled Reactor (HTGR)) technologies. High-temperature performance and overall chemical stability make molten fluoride salts desirable as working fluids for nuclear reactors [76]. Using molten salts as reactor coolants introduces inherent safety due to the salts' high boiling temperature and high volumetric heat capacity [34]. One coolant salt is the fluoride salt Li_2BeF_4 (FLiBe), which remains liquid without pressurization up to 1400 °C and has a bigger heat capacity than water [34, 22]. FHRs' TRISO particles' solid fuel cladding adds an extra barrier to fission product release compared to MSRs with liquid fuel. [34].

VHTR technology delivers heat at substantially higher temperatures than Light Water Reactors (LWRs), resulting in the following advantages: increased power conversion efficiency, reduced waste heat generation, and co-generation and process heat capabilities [76]. VHTRs system helium coolant's high 100 atm pressurization requires an expensive thick concrete reactor vessel whereas, the FHR system room pressure FLiBe coolant does not. The molten salt coolant has superior cooling and moderating properties compared to helium coolant in VHTRs. Accordingly, FHRs operate at power densities two to six times higher than VHTRs [76, 22]. By combining the FLiBE coolant from MSR technology and TRISO particles from VHTR technology, the FHR benefits from a low operating pressure and large thermal margin enabled by the molten salt coolant and the thermal resilience of TRISO particle fuel.

Several types of FHR conceptual designs exist worldwide: the Pebble-Bed Fluoride-Salt-Cooled High-Temperature Reactor (PB-FHR) at University of California Berkeley (UCB)

with circulating pebble-fuel [77, 44], the Solid Fuel Thorium Molten Salt Reactor (SF-TMSR) at the Shanghai Institute of Applied Physics (SINAP) in China with static pebble-fuel [52], the large central-station AHTR at ORNL [36, 90] and the Small Modular AHTR (SmAHTR) at ORNL [33] with static, plate fuel.

2.1.1 Advanced High Temperature Reactor Design

This proposed work focuses on a prismatic FHR design with hexagonal fuel assemblies consisting of TRISO fuel particles embedded in planks, i.e., the AHTR design developed by ORNL. The AHTR has 3400 MWt thermal power and 1400 MW electric power with inlet/outlet temperatures of 650/700°C [90]. Figure 2.1 shows the prismatic AHTR's fuel assembly and core configuration. Each hexagonal fuel assembly features plate-type fuel consisting of eighteen planks arranged in three diamond-shaped sectors, with a central Y-shaped structure and external channel (wrapper). The fuel planks contain an isostatically pressed carbon with fuel stripes on each plank's outer side. Within each fuel stripe is a graphite matrix filled with a cubic lattice of TRISO particles. The core consists of 252 assemblies radially surrounded by reflectors [68]. Chapter 3 details the specifications of the AHTR geometry modeled in this proposed work.

2.1.2 Previous AHTR modeling efforts and challenges

AHTR Neutronics Modeling

The AHTR core design differs significantly from the present LWR-based nuclear power plants. These differences lead to modeling challenges and the need to verify and validate simulation methods [68]. Verification and validation of neutronics and thermal-hydraulics simulation capabilities in the context of the AHTR design crucially support licensure of the AHTR design towards the eventual goal of deployment [66, 67]. Several neutronic studies conducted along the way to the current AHTR design have shed light on the technical

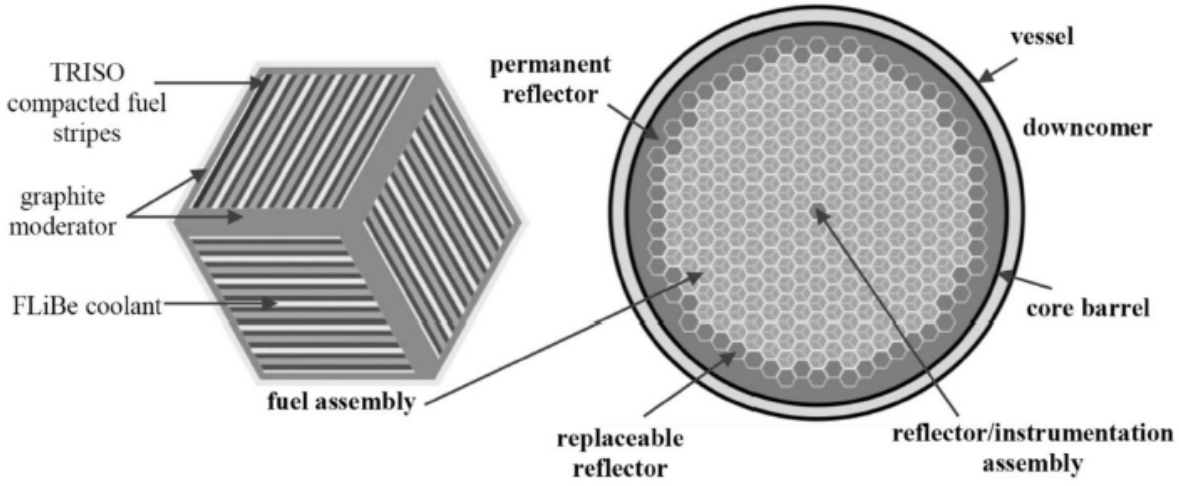


Figure 2.1: Advanced High Temperature Reactor fuel assembly (left) and core configuration (right) reproduced from [68].

challenges facing the design [68, 35, 33].

The Georgia Institute of Technology (Georgia Tech) led an Integrated Research Project to understand challenges in AHTR materials and modeling its neutronics and thermal-hydraulics [91]. During the research project, a panel of subject matter experts generated a Phenomena Identification and Ranking Table (PIRT). The PIRT identifies areas that need additional research to better understand important phenomena for adequate future modeling [66]. Table 2.1 lists the phenomena identified as requiring further research.

The *multiply heterogeneous* AHTR fuel, comprised of TRISO particles embedded in strategically arranged plates, presents simulation challenges. Researchers must obtain detailed reference power distributions with individual TRISO particle fidelity to best understand the AHTR's nuances. However, deterministic codes that use multigroup cross sections and traditional homogenization methods [68], insufficiently capture the correct physics in AHTRs due to these multiple heterogeneities [68]. In the AHTR, single and multiple slab homogenization decreased total neutron transport simulation time by an order of 10; however, the homogenization introduced a nontrivial error of ~3% [68, 15]. To determine the feasibility and safety of the AHTR design, researchers must calculate core physics parame-

Table 2.1: Phenomena Identification and Ranking Table identified Advanced High Temperature Reactor physical phenomena requiring further research [66].

Category	**Phenomena**
Fundamental cross section data	- Moderation in FliBe - Thermalization in FliBe - Absorption in FliBe - Thermalization in carbon - Absorption in carbon
Material Composition	- Fuel particle distribution
Computational Methodology	- Solution Convergence - Granularity of depletion regions - Multiple heterogeneity treatment for generating multigroup cross sections - Selection of multigroup structure - Boundary conditions for multigroup cross section generation
General Depletion	Spectral history

ters to an acceptable uncertainty. With Monte Carlo neutron transport, increasing neutron histories reduces statistical uncertainty but increases computational cost typically, requiring the use of supercomputers to run the simulations.

This AHTR presents another technical challenge: the uncertainty of graphite moderator material properties: densities, temperatures, and thermal scattering data. Problematically, the thermal scattering data ($S(\alpha, \beta)$ matrices) for the bound nuclei in Fluoride-Lithium-Beryllium (FLiBe) salts are lacking [68]. Mei et al. [53] and Zhu et al. [92] examined the thermal scattering behavior of solid and liquid FLiBe. They concluded that the bound and free atom cross section of FLiBe are identical above 0.1eV and diverges below 0.01eV, which means that the use or absence of thermal scattering data will impact the accuracy of the results [68].

AHTR Multiphysics Modeling

In past effort towards multiphysics modeling of the AHTR, Gentry et al [28] developed an adapted lattice physics-to-core simulator two-step procedure with Serpent [46] and Nodal Eigenvalue, Steady-state, Transient, Le core Evaluator (NESTLE) [88]. The adapted lattice

physics-to-core simulator two-step procedure proved to be successful for LWRs in which few group assembly homogenized group constants are generated by 2-D transport lattice calculation and then core analysis is performed by 3-D nodal simulation [39, 28]. NESTLE's thermal-hydraulics utilizes a Homogenous Equilibrium Mixture (HEM) model for two-phase flow and it solves the few-group neutron diffusion equation utilizing the Nodal Expansion Method (NEM) for cartesian and hexagonal reactor geometries. Lin [47] used RELAP5, a system-level code, to perform AHTR thermal hydraulics transient simulations to investigate the capability of the passive heat removal system. In this AHTR RELAP5 model, the 252 assemblies are separated into four concentric rings and a uniform power distribution is assigned to the fuel assemblies in each ring, and more fidelity is placed on the primary and Direct Reactor Auxiliary Cooling System (DRACS) system loops.

2.1.3 AHTR Benchmark

To address and further understand the technical challenges described in the previous section, in 2019, the OECD-NEA initiated a benchmark exercise to assess the modeling and simulation capabilities for AHTRs with TRISO fuel embedded in fuel planks of hexagonal fuel assemblies [2]. The benchmark plans to have three phases, starting from a single fuel assembly simulation without burnup, gradually extending to full core depletion and feedback. The benchmark's overarching objective is to identify the applicability, accuracy, and practicality of the latest methods and codes to assess the current state of the art of FHR simulation and modeling [64]. The benchmark also enables the cross-verification of software and methods for the challenging AHTR geometry, which is especially useful since applicable reactor physics experiments for code validation are scarce [63, 64]. Chapter 3 will provide a detailed description of the benchmark phases and results obtained so far.

2.2 Additive Manufacturing

Additive manufacturing is the formal term for what is popularly known as '3D printing' [29]. The basic principle of additive manufacturing is that a model is initially generated using a three-dimensional Computer-Aided Design (3D CAD) system and then fabricated directly without the need for process planning. Additive manufacturing, as the name implies, adds material in layers, such that each layer is a thin cross section of 3D CAD-designed part, as opposed to traditional machining which subtracts material instead [83]. All commercialized additive manufacturing machines to date use a layer-based approach, and the major ways that they differ are in materials, layer creation method, and how the layers are bonded to each other [29]. These major differences will determine the: accuracy of the final part, material and mechanical properties, the time required to manufacture the part, the need for post-processing, the size of additive manufacturing machine, and the overall cost of the machine and the process [29]. Initially, industries only utilized additive manufacturing for manufacturing prototypes. However, with improvements in material properties, accuracy, and overall quality of additive manufacturing output, the applications for additive manufacturing expanded to the point at which some industries build parts for direct assembly purposes [89]. Furthermore, using additive manufacturing in conjunction with other technologies, such as high-power lasers, has enabled additive manufacturing of parts made from various metals [29].

Additive manufacturing has progressed rapidly in the last 30 years, from rapid design prototyping with polymers in the automotive industry to scale production of metal components. Examples include Boeing using additive manufacturing to reduce the 979 Dreamliner's weight [4] and General Electric using additive manufacturing to produce fuel injection nozzles [6]. The most common metal additive manufacturing technologies, selective laser melting (SLM), electron beam melting (EBM), laser directed energy deposition (L-DED), and binder jetting, are not currently used to manufacture nuclear power plant parts.

The U.S. Department of Energy (DOE), National Laboratories, Electric Power Research Institute (EPRI) support research and development efforts towards deployment, testing, and qualification of additive manufacturing methods for nuclear components. However, the nuclear industry's efforts to incorporate additive manufacturing into the supply chain lags behind the auto and aerospace industries due to the lack of clarity on regulatory pathways. The aerospace and automotive industries benefit from long-standing and resourced regulatory and standards development activities [3]. Thus, in 2019 the Nuclear Regulatory Commission (NRC) addressed these regulatory challenges by issuing a draft action plan to prepare the agency to review applications for additive manufacturing of nuclear components and clarify the industry's expectations of their use [3].

2.2.1 Benefits of Additive Manufacturing for Nuclear Reactor Core Components

Wide-spread adoption of these methods in the nuclear industry could drastically reduce fabrication costs and timelines, combine multiple systems and assembled components into single parts, increase safety and performance by tailoring local material properties, and enable geometry redesign for optimal load paths [79]. Many Generation IV advanced reactor concepts have complex geometries, such as hex-ducts for sodium-cooled fast reactors, that are costly and difficult to fabricate using standard processing techniques. Traditional manufacturing routes also restrict many viable geometries for reactor designers to explore [82]. In summary, reactor core component fabrication with additive manufacturing enables further optimization and improvement of fuel geometries to enhance fuel performance at lower costs [8].

2.2.2 Demonstration of Additive Manufacturing for Nuclear Reactor Core Components

Recent nuclear materials experiments have demonstrated the application of additive manufacturing to nuclear fuel and structural core material fabrication. Rosales et al. [71] conducted a feasibility study of direct routes to fabricate dense uranium silicide (U_3Si_2) fuel pellets using the Idaho National Laboratory (INL) approach known as Additive Manufacturing as an Alternative Fabrication Technique (AMAFT). U_3Si_2 demonstrates desirable accident-tolerant nuclear fuel properties such as high uranium density and improved thermal properties, however, it has an expensive and long metallurgical fabrication process. Thus, using AMAFT to fabricate U_3Si_2 will lower cost and ensure a timely and commercially-reliable fabrication process [71]. Sridharan et al. [82] demonstrated the application of the laser-blown-powder additive manufacturing process to fabricate ferritic/martensitic (FM) steel, a type of steel commonly used for cladding and structural components in nuclear reactors. Koyanagi et al. [40] presented the latest additive manufacturing technology for manufacturing nuclear-grade silicon carbide (SiC) materials. They demonstrated that combinations of additive manufacturing techniques and traditional SiC densification methods enabled new designs of SiC components with complex shapes. SiC demonstrates excellent strength at elevated temperatures, chemical inertness, relatively low neutron absorption, and stability under neutron irradiation up to high doses [75, 80, 40]. These qualities make SiC suitable for many applications in nuclear systems such as fuel cladding, constituents of fuel particles [80] and pellets [85], and core structural components in fission reactors [75]. Trammel et al [87] conducted a preliminary investigation to assess the possibilities of fabricating a fuel element with embedded TRISO fuel using additive manufacturing techniques, such as binderjet printing and chemical vapor infiltration (CVI). They successfully demonstrated a fabrication method using the following steps (depicted in Figure 2.2):

1. A SiC fuel element structure is first printed with binderjet technology.

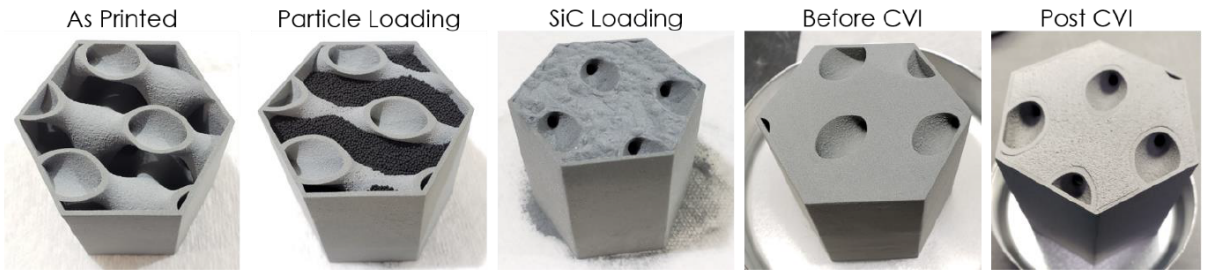


Figure 2.2: Stages of additive manufacturing fabrication conducted at Oak Ridge National Laboratory to produce a fuel element with non-homogenously shaped coolant channels and TRISO particles embedded in a SiC matrix [87].

2. The designated fueled region of the element is loaded with surrogate TRISO particles and additional SiC powder to fill interstitial spaces between particles.

3. The loaded fuel element is densified in a CVI process to achieve microencapsulation of TRISO particles in a SiC matrix.

Many of the materials and fabrication methods discussed are applicable for FHR-part manufacturing. Therefore, this reiterates the possibility of leveraging additive manufacturing to 3D print a FHR-type reactor with non-conventional geometry.

2.3 Nuclear Reactor Design Optimization

With the conception of nuclear reactors came along the practice of nuclear reactor optimization. Nuclear engineering sub-fields such as reactor design, reactor reloading patterns, and the nuclear fuel cycle utilize optimization. Traditional manufacturing constraints limit nuclear reactor core design optimization. In the proposed work, I will reexamine nuclear reactor core design optimization for arbitrary reactor geometries and fuel distributions enabled by additive manufacturing.

2.3.1 Previous Efforts for Reactor Design Optimization

Previous efforts towards reactor design optimization processes utilized deterministic and stochastic optimization techniques, coupled with surrogate models. Deterministic optimization methods usually start from a guess solution. Then, the algorithm suggests a search direction by applying local information to a pre-specified transition rule. Any better solution becomes the new solution, and the above procedure continues several times [18]. Drawbacks of deterministic methods include: algorithms tend to get stuck at suboptimal solutions, and an algorithm efficient in solving one type of problem may not solve a different problem efficiently [18]. Stochastic optimization methods, such as evolutionary algorithms amd simulated annealing, minimize or maximize an objective function when randomness is present. Stochasticity enables them to find globally optimal solutions more reliably than deterministic methods.

A nuclear reactor's complexity results in reactor design optimization being a multi-objective design problem requiring a tradeoff between desirable attributes [11, 78]. When multiple conflicting objectives compete no single optimum solution simultaneously optimizes all objectives. Instead, multi-objective optimization returns multiple optimal solutions that meets each objective to varying degrees [18]. For a multi-objective problem like reactor design, an ideal optimization method should find widely spread solutions in the obtained non-dominated front [18]. For each solution in a set of non-dominated front solutions, none of the objective functions can be improved in value without degrading some of the other objective values.

Recent efforts towards nuclear reactor optimization have relied heavily on the aforementioned stochastic methods, with the occasional addition of stochastic-deterministic hybrid methods.

Stochastic Optimization: Simulated Annealing Method

Simulated annealing iteratively updates one candidate solution until it reaches the termination criteria. At each iteration, the simulated annealing algorithm selects a random move. If the selected move improves the solution, it is always accepted; however, if it does not improve the solution, the algorithm updates the solution with some probability of less than 1.

Sacco et al. [73, 72] used stochastic simulated annealing and deterministic-stochastic hybrid optimization techniques to optimize reactor dimensions, enrichment, materials, etc., to minimize the average peaking factor in a three-enrichment-zone reactor. Odeh et al. [57] used the simulated annealing stochastic algorithm coupled with neutronics and thermal-hydraulics simulation tools, Purdue Advanced Reactor Core Simulator (PARCS) and RELAP5 [21], to develop an optimal Purdue Novel Modular Reactor (NMR-50) core design with a 10-year cycle length and minimal fissile loading. Kropaczek et al. [43] demonstrated the constraint annealing method: a highly scalable method based on the method of parallel simulated annealing with mixing of states [42] for the solution of large-scale, multiconstrained problems in LWR fuel cycle optimization. These papers demonstrate the simulated annealing optimization method's success in reactor design optimization problems.

Nuclear reactor optimization problems require computationally expensive neutronics and thermal-hydraulics software to compute the objective function and constraints. Multiple papers utilized stochastic optimization methods with surrogate models and replace, high-fidelity neutronics or thermal hydraulics simulations to reduce computational cost. Kumar et al. [45] combined genetic algorithm optimization with a surrogate model to optimize for high breeding of ^{233}U and ^{239}Pu in desired power peaking limits and keff by varying: fuel pin radius, fissile material isotopic enrichment, coolant mass flow rate, and core inlet coolant temperature. Betzler et al. [9] developed a systematic approach to build a surrogate model to serve in place of high-fidelity computational analyses. They leveraged the surrogate model

with a simulated annealing optimization algorithm to generate optimized designs at a lower computational cost and understand the impact of design decisions on desired metrics for High Flux Isotope Reactor (HFIR) low-enriched uranium (LEU) core designs.

The simulated annealing method uses a point-by-point approach: one solution gets updated to a new solution in one iteration, which does not exploit parallel systems' advantages. Finding an optimal solution with simulated annealing methods takes very long if high-fidelity computationally expensive codes are used to compute the objective function and constraints. Using a simulated annealing method is only practical if a surrogate evaluation model is used, as described in Betzler et al. [9] and Kumar et al. [45]. Therefore, in the proposed PhD scope, the stochastic evolutionary algorithm optimization method is used.

Stochastic Optimization: Evolutionary Algorithm Method

Peireira et al. [60, 61] used a coarse-grained parallel genetic algorithm and a niching genetic algorithm to optimize the same problem as Sacco et al. [73]. Kamalpour et al. [38] utilized the imperialist competitive algorithm, a type of evolutionary algorithm, to optimize a fully ceramic microencapsulated (FCM) fuelled Pressurized Water Reactor (PWR) to extend the reactor core cycle length.

Contrary to a single solution per iteration in deterministic and stochastic simulated annealing methods, evolutionary algorithms use a population of solutions in each iteration [18]. Evolutionary algorithm methods mimic nature's evolutionary principles to drive the search towards an optimal solution. With the affordability and availability of parallel computing systems, the evolutionary algorithm optimization method stands out as a method that easily and conveniently exploits parallel systems. Further, evolutionary algorithms have proved amenable to HPC solutions and scalable to tens of thousands of processors [42]. Thus, for optimization problems that require high-fidelity evaluation software, the evolutionary algorithm method can leverage parallel computing to find a solution faster than the simulated annealing method. Therefore, in this proposed work, I will utilize the evolutionary algorithm

optimization method.

2.3.2 Impact of Additive Manufacturing on Nuclear Reactor Design Optimization

Previous efforts toward nuclear reactor optimization, as discussed in the above section, focused on optimizing classical reactor parameters such as radius of fuel pellet and clad, enrichment of fuel, pin pitch, etc. Additive manufacturing advancements for reactor core components remove conventional fuel manufacturing geometric constraints, allowing the reactor designers to optimize beyond classical parameters to enhance fuel performance and safety further. Reactor design objectives remain consistent with past objectives, such as minimizing fuel amount and minimizing the maximum fuel temperature for a given power level. However, reactor designers can now approach the nuclear design problems with truly arbitrary geometries, no longer limited by traditional geometric shapes that are easy to manufacture with traditional processes: slabs as fuel planks, cylinders as fuel rods, spheres as fuel pebbles, axis-aligned coolant channels, etc [81]. This has opened the door for a re-examination of reactor core optimization in a completely new way, determining the optimal arbitrary geometry for a given objective function with a much smaller set of constraints [81].

With a substantial increase and change in an arbitrary geometry's design space, it becomes time consuming for a human reactor designer to thoroughly explore and find optimal geometries in the expanded design space. Instead, we can leverage artificial intelligence (AI) optimization methods (such as evolutionary algorithms) to promptly explore the large design space to find global optimal designs. AI does not replace the human reactor designer but shifts the human designer's focus away from conjecturing suitable geometries to defining design criteria to find optimal designs [81]. Thus, when the human designer changes the reactor criteria, the AI model will quickly adapt and produce new global optimal designs to fit the new criteria.

2.4 Evolutionary Algorithms

AI research studies 'intelligent agents': any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals [17]. Evolutionary algorithms, a subset of AI, create a population of individual solutions, inspired by biological evolution, and induce goals by using a 'fitness function' to mutate and preferentially replicate high-scoring individuals to reach an optimal solution. Evolutionary algorithms often perform well at approximating solutions to many problem types because they do not make any assumptions about the underlying fitness landscape. Genetic algorithms are the most popular evolutionary algorithms for solving multi-objective problems [11, 41].

2.4.1 Genetic Algorithms

Genetic algorithms imitate natural genetics and selection to evolve solutions by maintaining a population of solutions, allowing fitter solutions to reproduce and letting lesser fit solutions die off, resulting in final solutions that are better than the previous generations [69]. From here, we will refer to a solution as an individual within the population. Genetic algorithms efficiently exploit historical information to speculate new search points, improving each subsequent population's performance [31]. They are theoretically and empirically proven to provide robust search in complex spaces and are computationally simple yet powerful in their search for improvement [31]. Genetic algorithms trounce deterministic and stochastic simulated annealing optimization methods, because they:

1. search from a population of points
2. use objective function information, not derivatives or other auxiliary knowledge of the problem
3. tuse probabilistic transition rules, not deterministic rules

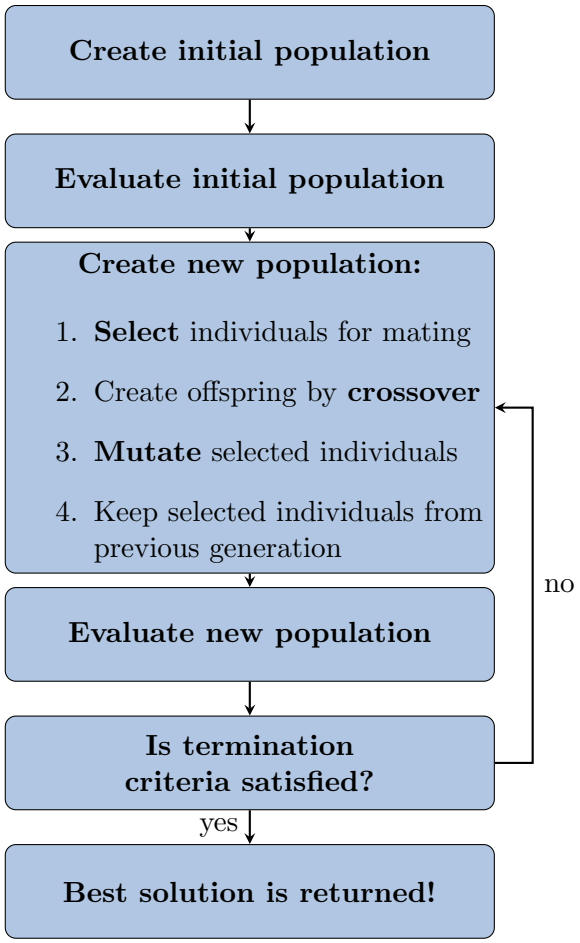


Figure 2.3: Process of finding optimal solutions for a problem with a evolutionary algorithm [69].

Figure 2.3 depicts the iterative process of using a genetic algorithm to solve a problem. The genetic algorithm generates new populations iteratively until it meets the termination criteria.

Genetic algorithms use mechanisms inspired by biological evolution such as selection, crossover, and mutation. The three operators are simple and straightforward. The selection operator selects good individuals. The crossover operator recombines good individuals to form a better individual. The mutation operator alters individuals to create better individuals [18]. Next, we provide more description and common methods for each operator.

Selection Operator

The selection operator duplicates good individuals and eliminates bad individuals while keeping the population constant [18]. It achieves this by identifying above-average individuals in a population, eliminating bad individuals from the population, and replacing them with copies of good individuals. Selection operator methods utilized in the proposed work include tournament selection, best selection, and Non-dominated sorting genetic algorithm II (NSGA-II) selection. In *tournament selection*, a user-defined number of individuals play in tournaments, and the best individual proceeds to the next population. This repeats until all the population's spots are filled. In *best selection*, the operator selects a user-defined number of best individuals, and copies are made to keep the population size constant. In *NSGA-II selection*, the operator selects the best individuals from the combination of parent and offspring populations [19]. The operator maintains population size by adding copies of the best individuals.

The selection operator does not create any new individuals in the population and only makes more copies of good individuals at the expense of not-so-good individuals. Instead, crossover and mutation operators perform the creation of new solutions.

Crossover/Mating Operator

In most crossover operators, the operator picks two individuals from the population at random. The operator exchanges some portion of each individuals' attributes with one another to create two new individuals [18]. Crossover operator methods utilized in the proposed work include *single-point crossover*, *uniform crossover*, and *blend crossover*. In the *single-point crossover*, the operator randomly selects two individuals from the population and a site along the individual's definition. For example, if the individual is a list, the operator randomly chooses an element in the list and attributes on this cross site's right side are exchanged between the two individuals, creating two new offspring individuals. In a *uniform crossover*, the user defines an independent probability for each individual's

attribute to be exchanged; usually, $p = 0.5$ is used. In *blend crossover*, the operator creates two offspring (O) individuals based on a linear combination of two-parent (P) individuals using the following equations:

$$O_1 = P_1 - \alpha(P_1 - P_2) \tag{2.1}$$

$$O_2 = P_2 + \alpha(P_1 - P_2) \tag{2.2}$$

where

α = Extent of the interval in which the new values can be drawn
for each attribute on both side of the parents attributes (user-defined)

The user defines a crossover probability (p_c) to preserve some good individuals selected during the selection operator stage. Therefore, the crossover operator only operates on $100p_c\%$ of the population; the rest proceed to the new population [18]. The crossover operator covers the search aspect of the genetic algorithms, whereas the mutation operator keeps diversity in the population [18].

Mutation Operator

The mutation operator alters one or more attributes of an individual within a population. Mutation occurs in the genetic algorithm based on a user-defined mutation probability (p_m). A low p_m prevents a primitive random search. Mutation operator methods utilized in the proposed work include *polynomial bounded mutation*, in which each attribute in each individual is mutated based on a polynomial distribution. The user also defines each attribute's upper and lower bounds and the crowding-degree of the mutation, η (a large η will produce a mutant resembling its parent, while a small η will produce the opposite).

2.4.2 Genetic Algorithm Hyperparameter Tuning

Hyperparameters refer to parameters whose value controls the genetic algorithm's process, such as the population size. A well-performing genetic algorithm needs to balance the extent of exploration and exploitation; by finding a balance between the conservation of valuable individuals obtained until the current generation while exploring new individuals. With over exploitation of previously obtained individuals, the population loses its diversity, resulting in premature convergence to a suboptimal solution. Alternatively, if too much stress is given on exploration, the algorithm did not appropriately utilize the information obtained thus far, and the genetic algorithm's search procedure behaves like a random search process [18]. A quantitative balance between these two issues, exploitation and exploration, is challenging to achieve. Deb et al. [18] and Goldberg et al. [32] quantified the relationship between exploitation and exploration. They found that for the one-max test problem, in which the objective seeks to maximize the number of 1s in a string, a genetic algorithm with any arbitrary hyperparameter setting does not work well even on a simple problem. Only genetic algorithms with a selection pressure (s) and crossover probability (p_c) falling inside the control map (Figure 2.4) will find the desired optimum. Another consideration is the population size. A function with considerable variability in objective function values demands a large population size to find a global optimum [18].

Therefore, finding an optimized solution with genetic algorithms requires the user to conduct a hyperparameter search. Ng et al. [56] suggest that a coarse-to-fine sampling scheme is the best way to perform a systematic hyperparameter search. For a two-dimensional example of a coarse-to-fine sampling scheme, the user first does a coarse sample of the entire square, then a fine search on the coarse search's best-performing region. Ng et al. also suggests to use random sampling over grid sampling because of the former's efficiency in high-dimensional spaces. Figure 2.5 illustrates how grid sampling gives even coverage in the original 2-d space, but provides inefficient coverage in projections onto either the x1 or x2

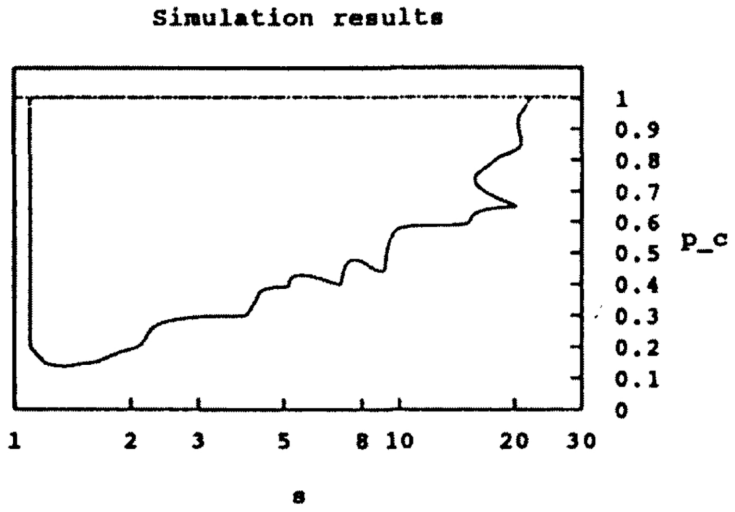


Figure 2.4: Figure reproduced from [32, 18] shows a control map region of selection pressure (s) and crossover probability (p_c) values in which the genetic algorithm will find the desired optimum for the one-max problem.

subspace. In contrast, random sampling produces a less even distribution in the original space, but a far more even distribution in the subspaces.

2.5 Summary

This chapter provided a literature review of relevant past research efforts that give context to the proposed PhD scope. In summary, participation in the OECD-NEA FHR benchmarking exercise contributes to the assessment of the current neutron transport and thermal-hydraulics modeling and simulation capabilities for the AHTR design. Also, additive manufacturing of nuclear reactor components is a quickly developing field thanks to the aerospace and auto industries, which led to breakthroughs in additive manufacturing fabrication of metal components. The promise of cheaper and faster manufacturing of reactor components with additive manufacturing frees complex reactor geometries from previous manufacturing constraints and allows reactor designers to reexamine reactor design optimization. Stochastic optimization methods such as evolutionary algorithms have proven to work well for finding global optimums in multi-objective design problems such as nuclear reactor optimization

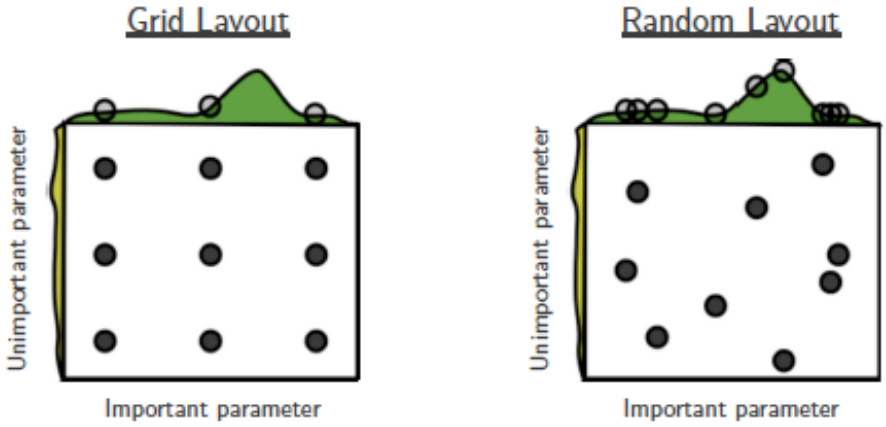


Figure 2.5: The impact of grid sampling vs random sampling on coverage of projections into subspaces (reproduced from [37]). Random sampling has better coverage in the subspaces.

and can be leveraged to explore the vast exploration design space enabled by additive manufacturing.

Chapter 3

Fluoride-Salt-Cooled High-Temperature Reactor Benchmark

Fluoride-Salt-Cooled High-Temperature Reactors (FHRs) use TRISO fuel and a low-pressure liquid fluoride-salt coolant. FHR technology combines FLiBe coolant from MSRs and TRISO particles from VHTRs to enable a reactor with low operating pressure, large thermal margin, and accident-tolerant qualities. Within the FHR reactor class, AHTRs have plate-based fuel in a hexagonal fuel assembly. To address the AHTR modeling challenges, described in Chapter 2, such as multiple heterogeneity and material cross-section data, the OECD-NEA and Georgia Tech initiated the FHR benchmark for the AHTR design in 2019 [2]. In section 2.1, I gave an FHR concept overview, an AHTR design description, a review of previous efforts towards modeling these designs, and how these efforts led to the benchmark initiation.

The three-phase FHR benchmark begins with a single fuel assembly simulation without burnup and gradually extends to full core depletion. Table 3.1 outlines the complete and incomplete benchmark phases. In the subsequent sections, I will describe the benchmark's

Table 3.1: Organisation for Economic Co-operation and Development (OECD) Nuclear Energy Agency (NEA) Fluoride-Salt-Cooled High-Temperature Reactor (FHR) benchmark Phases [2].

Phases	**Sub-phases**	**Description**	**Completed?**
Phase I: fuel assembly	I-A	2D model, steady-state	✓
	I-B	2D model depletion	✓
	I-C	3D model depletion	
Phase II: 3D full core	II-A	Steady-state	
	II-B	Depletion	
Phase III: 3D full core with feedback & multicycle analysis	III-A	Full core depletion with feedback	
	III-B	Multicycle analysis	

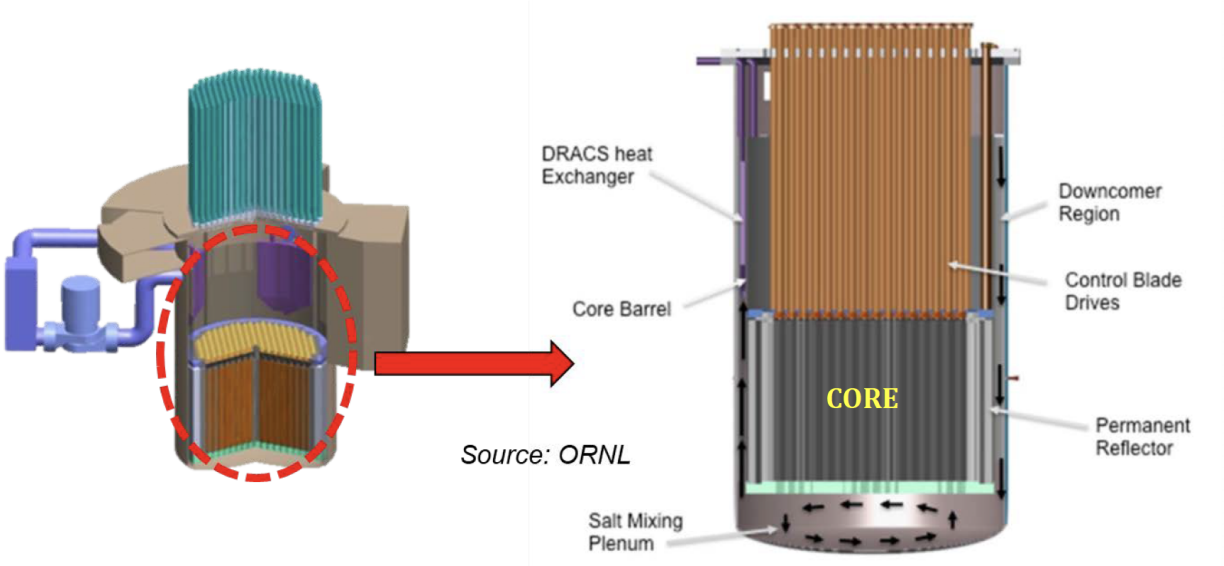


Figure 3.1: Advanced High Temperature Reactor (AHTR) schematic (left) and vessel (right) reproduced from [2].

specifications for the AHTR design and Phase I. Then, I will share our Phase I-A and I-B results generated with the OpenMC neutronics code [70].

3.1 FHR Benchmark Advanced High Temperature Reactor Design

Figure 3.1 shows the Advanced High Temperature Reactor (AHTR) schematic and a vertical cut of the reactor vessel. The AHTR operates at 3400 MWt thermal power and 1400 MWe electric power [90]. The $10m$-diameter exterior reactor vessel contains an $8m$-diameter reactor core that which in turn contains 252 hexagonal fuel assemblies.

Each $6m$ high fuel assembly comprises a $5.5m$ active core region, which contains TRISO particles, and $0.25m$ top and bottom non-fuelled reflector regions. Figure 2.1, from Chapter 2, shows a single hexagonal fuel assembly geometry and the arrangement of all assemblies in the core. All dimensions specified are at room temperature. The benchmark's phases I and II use room temperature dimensions while phase III will address dimensional changes brought

about by temperature expansion. Figure 3.2 shows a detailed 2D view of the AHTR's hexagonal fuel assembly. The hexagonal fuel assembly consists of eighteen fuel-containing graphite planks arranged in three diamond-shaped sectors, with a external channel wrapper and structural Y-shape, made of C-C composite with extra notches to hold the fuel planks in place. The diamond-shaped sections have 120° rotational symmetry with each other [90, 68, 2]. Semi-cylindrical spacers attach to the fuel planks with radius equalling to coolant channel thickness. FLiBe coolant fills the gaps between the fuel planks, and between assemblies (note: FliBe layer around the single assembly). The Y-shaped control rod slot at the center of the Y-shape structure contains FLiBe coolant when the control blade is not in the slot (as seen in Figure 3.2) [90, 68, 2]. For a single fuel assembly, the internal 120° rotational symmetry is represented by periodic boundary conditions, as seen in Figure 3.3.

Figure 3.4 magnifies a single fuel plank. Fuel stripes line the upper and lower sides of each graphite fuel plank. Each fuel stripe contains a graphite matrix filled with a cubic lattice of TRISO particles with 40% packing fraction. The lattice is 210 TRISO particles wide in the x-direction, four particles deep in the y-direction, and 5936 particles tall in the z-direction. Figure 3.5 shows the TRISO particle's cross section which consists of five layers: oxycarbide fuel kernel, porous carbon buffer, inner pyrolytic carbon, silicon carbide layer, and the outer pyrolitic carbon.

To control reactivity, the FHR benchmark includes AHTR configurations with burnable poisons and control rods. The burnable poisons consist of europium oxide, Eu_2O_3, and have a discrete or integral (dispersed) option. Figure 3.6 shows the discrete option with z-direction axially stacked small spherical Eu_2O_3 particles at five XY locations in each fuel plank. The integral options consists of Eu_2O_3 homogenously mixed with the graphite fuel plank (including the graphite in fuel stripes matrix and plank ends indented to structural sides, but excluding the graphite in spacers and graphite in TRISO particles).

The molybdenumhafnium carbide alloy (MHC) control rod inserts into the Y-shaped control rod slot where it displaces the FLiBe that occupies the slot (shown in Figure 3.2).

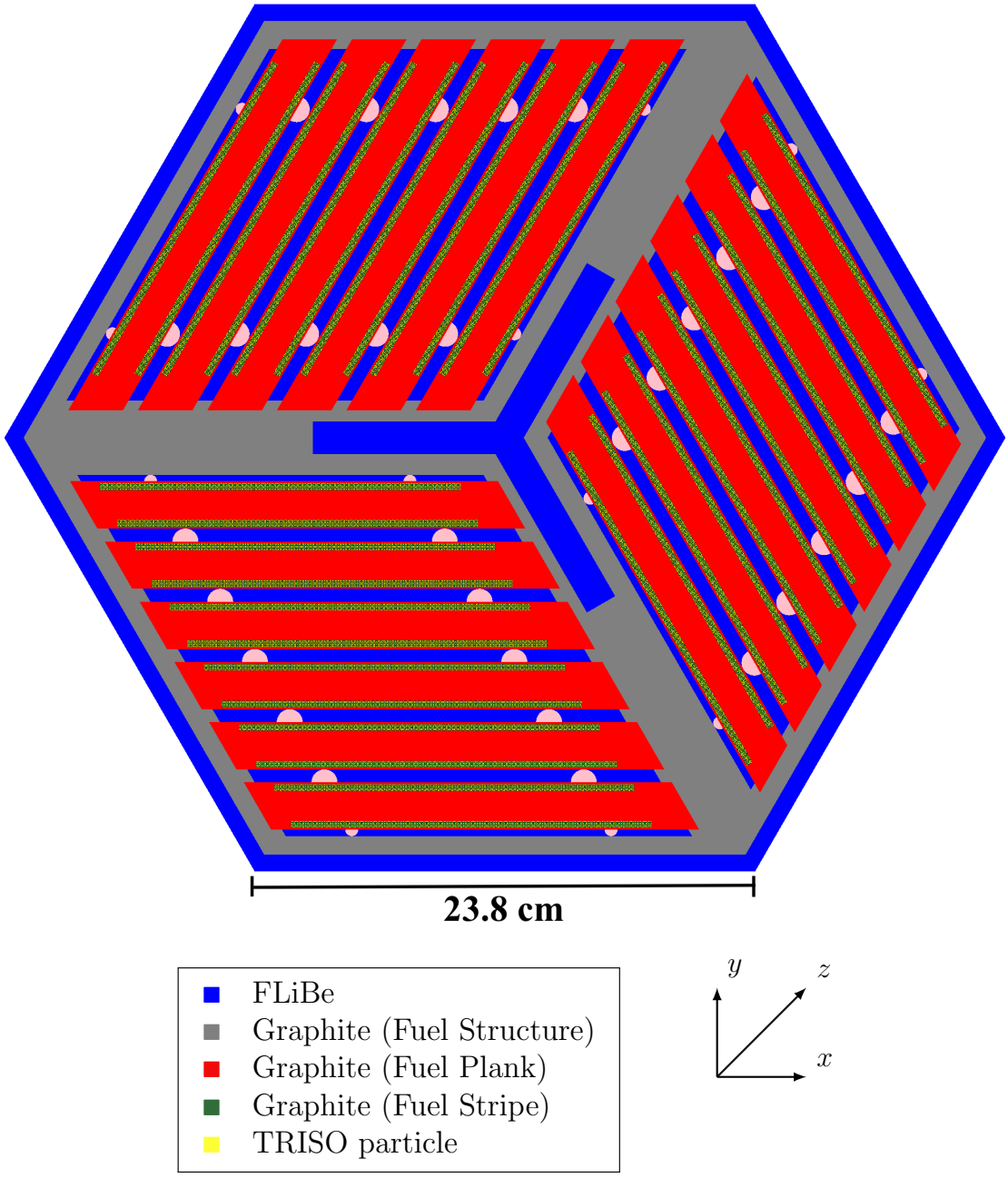


Figure 3.2: Advanced High Temperature Reactor (AHTR) fuel assembly with 18 fuel plates arranged in three diamond-shaped sectors, with a central Y-shaped and external channel graphite structure.

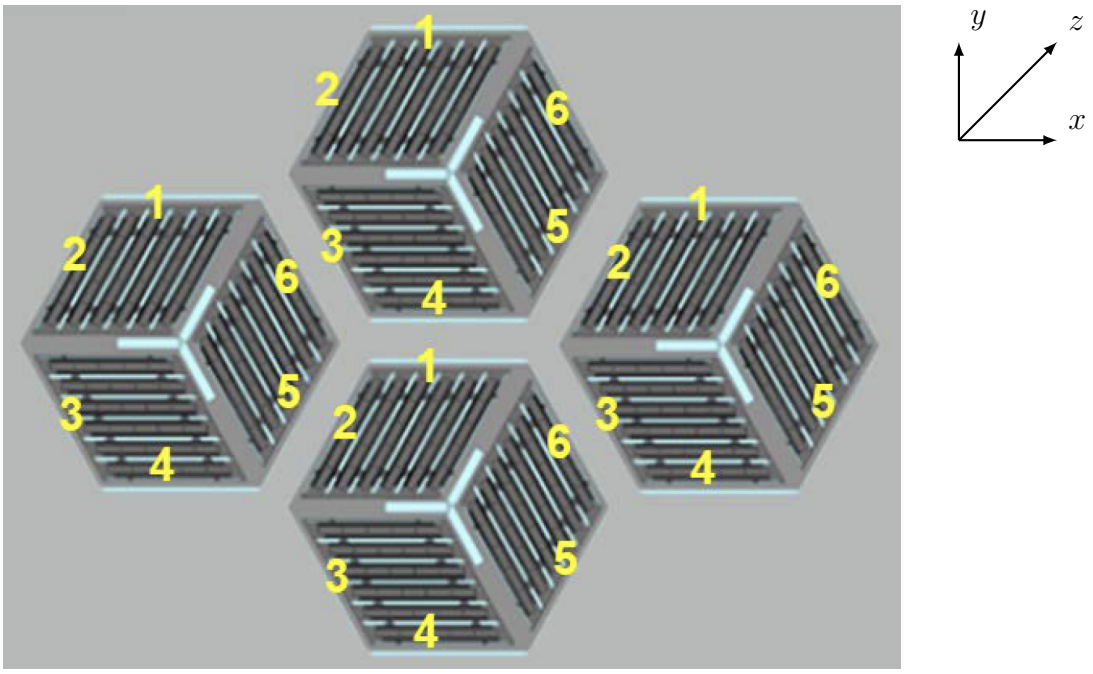


Figure 3.3: Visualization of periodic boundary conditions for a single fuel assembly in the Advanced High Temperature Reactor (AHTR), reproduced from [2].

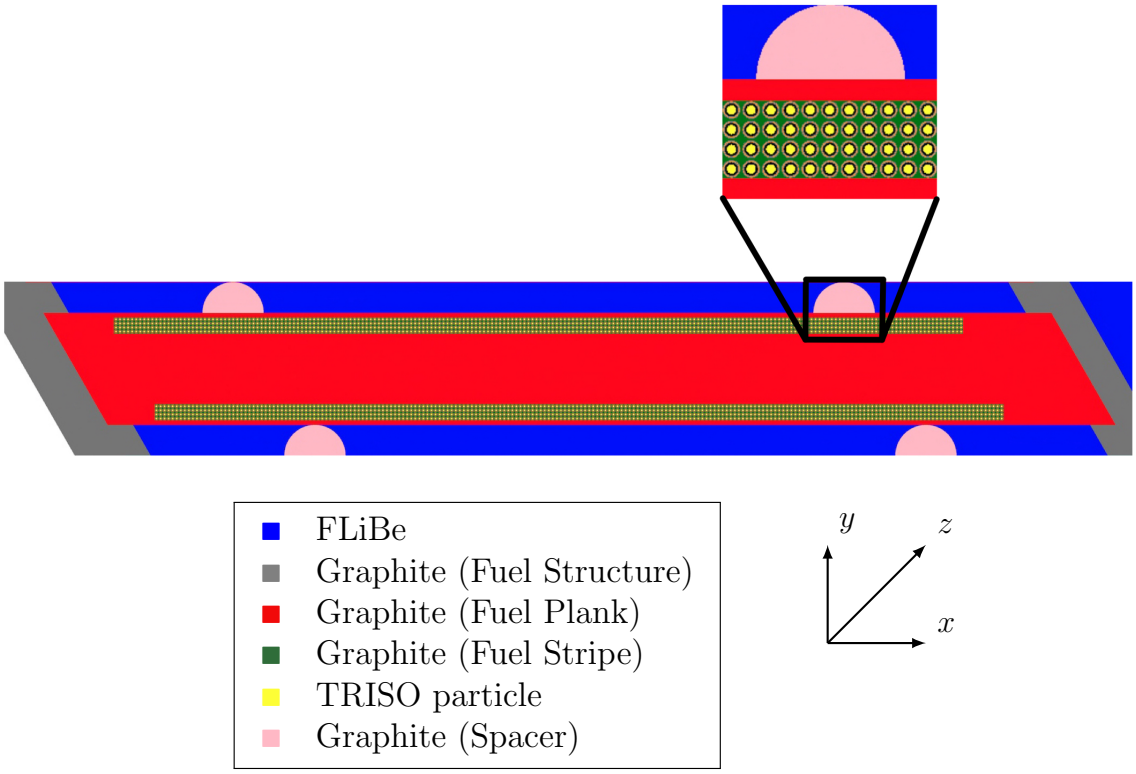


Figure 3.4: Advanced High Temperature Reactor (AHTR)'s fuel plank, with the magnification of a spacer and segment of the fuel stripe with embedded TRISO particles.

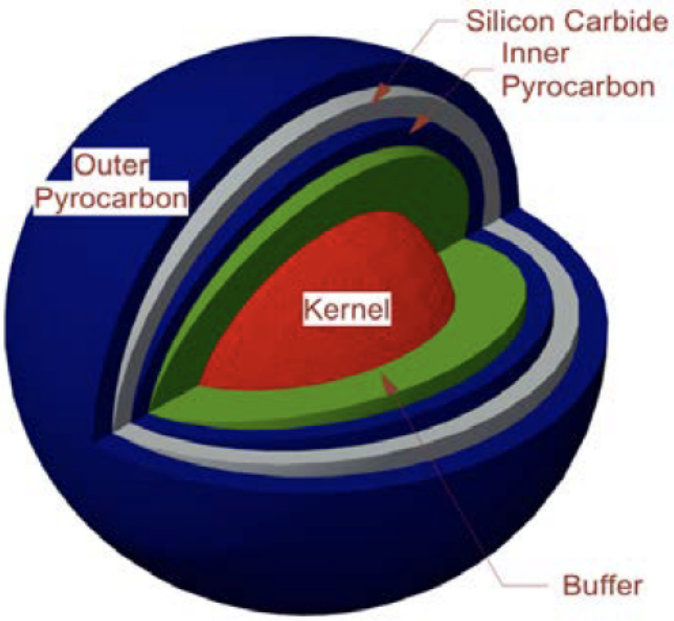


Figure 3.5: Advanced High Temperature Reactor's TRISO particle schematic reproduced from [2].

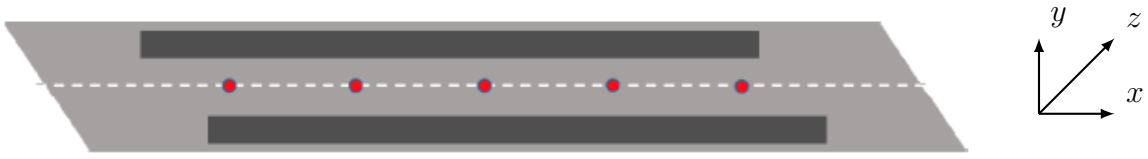


Figure 3.6: XY Placement of z-direction axial burnable poisons stacks in the Advanced High Temperature Reactor [2].

3.2 Benchmark Phase I Specifications

The FHR benchmark Phase I consists of three subphases. First, a steady-state 2D model (Phase I-A), depletion of one 2D FHR fuel assembly (Phase I-B), and depletion of one 3D FHR fuel assembly (Phase I-C). Benchmark organizers have only released Phase I-A and I-B's specifications, thus Phase I-C's specifications will be omitted in this section.

The benchmark requires the following results for Phases I-A and I-B:

(a) k_{eff} (effective multiplication factor)

(b) reactivity coefficients: β_{eff}, α_D (doppler coefficient), $\alpha_{T,FliBe}$ (FLiBe temperature coefficient), α_M (moderator temperature coefficient)

(c) tabulated fission source distribution by one-fifth fuel stripe

(d) $\bar{\phi_1}, \bar{\phi_2}, \bar{\phi_3}$ (neutron flux averaged over the whole model tabulated in three coarse energy groups)

(e) $\phi_1(\vec{x}, \vec{y}), \phi_2(\vec{x}, \vec{y}), \phi_3(\vec{x}, \vec{y})$ (neutron flux distribution in three coarse energy groups)

(f) fuel assembly averaged neutron spectrum

Next, I report the equations used to calculate these required results.

Reactivity Coefficients (b)

I assumed one energy group and six delayed groups for β_{eff}. Reactivity coefficient, α, is the change in reactivity (ρ) of the material per degree change in the material's temperature (T). I calculated each reactivity coefficient and its corresponding uncertainty with these equations:

$$\frac{\Delta\rho}{\Delta T} = \frac{\rho_{T_{high}} - \rho_{T_{low}}}{T_{high} - T_{low}} \left[\frac{pcm}{K}\right] \tag{3.1}$$

$$\delta\frac{\Delta\rho}{\Delta T} = \frac{\sqrt{\delta(\rho_{T_{high}})^2 + (\delta\rho_{T_{low}})^2}}{T_{high} - T_{low}} \left[\frac{pcm}{K}\right] \tag{3.2}$$

Fission Source Distribution / Fission Density (c)

I calculated fission density (FD) with OpenMC's `fission` tally score (f) for each region divided by the average `fission` tally score of all the regions:

$$FD_i = \frac{f_i}{f_{ave}} \ [-] \tag{3.3}$$

where

f_i = fission reaction rate in a single region i [reactions/src]

f_{ave} = average of all f_i [reactions/src]

The uncertainty calculations for FD_i and f_{ave}:

$$\delta FD_i = |FD_i| \sqrt{\left(\frac{\delta f_i}{f_i}\right)^2 + \left(\frac{\delta f_{ave}}{f_{ave}}\right)^2} \tag{3.4}$$

$$\delta f_{ave} = \frac{1}{N} \sqrt{\sum_i^N f_i^2} \tag{3.5}$$

where

N = No. of fission score values

Neutron Flux (d, e, f)

OpenMC's `flux` score is normalized per source particle simulated, resulting in $[\frac{neutrons\ cm}{src}]$ units. This is an unnatural unit for system analysis, and thus to better compare OpenMC results with other software results in the benchmark, I converted flux to $[\frac{neutrons}{cm^2 s}]$ units using

the following equations:

$$\Phi_c = \frac{N \cdot \Phi_o}{V} \tag{3.6}$$

$$N = \frac{P \cdot \nu}{Q \cdot k} \tag{3.7}$$

where

Φ_c = converted flux $[\frac{neutrons}{cm^2 s}]$

Φ_o = original flux $[\frac{neutrons\ cm}{src}]$

N = normalization factor $[\frac{src}{s}]$

V = volume of fuel assembly $[cm^3]$

P = power $[\frac{J}{s}]$

$\nu = \frac{\nu_f}{f}\ [\frac{neutrons}{fission}]$

Q = Energy produced per fission $[\frac{J}{fission}]$

$= 3.2044 \times 10^{-11}$ J per U_{235} fission

$k = k_{eff}\ [\frac{neutrons}{src}]$

The flux standard deviation is:

$$\delta\Phi_c = \Phi_c \times \sqrt{(\frac{\delta\Phi_o}{\Phi_o})^2 + (\frac{\delta\nu_f}{\nu_f})^2 + (\frac{\delta k}{k})^2 + (\frac{\delta f}{f})^2} \tag{3.8}$$

I calculated reactor power based on the given reference specific power (P_{sp}) of 200 $\frac{W}{gU}$:

$$P = P_{sp} \times V_F \times \rho_F \times \frac{wt\%_U}{100} \tag{3.9}$$

where

$$
\begin{aligned}
V_F &= \text{volume of fuel } [cm^3] \\
&= \frac{4}{3}\pi r_f^3 \times N_{total} \\
r_f &= \text{radius of fuel kernel within TRISO particle} \\
N_{total} &= \text{total no. of TRISO particles in fuel assembly} \\
&= 101 \times 210 \times 4 \times 2 \times 6 \times 3 \\
\rho_F &= \text{density of fuel } [g/cc] \\
wt\%_U &= \frac{at\%_{U235} \times m_{U235} + at\%_{U238} \times m_{U238}}{\sum(at\%_i \times m_i)} \times 100 \\
m &= \text{atomic mass}
\end{aligned}
$$

3.2.1 Phase I-A Specifications

For Phase I-A, the benchmark specifies that each participant must produce a steady-state 2D model of one fresh fuel assembly for nine cases and report the required results listed in Section 3.2. Table 3.2 describes each case.

3.2.2 Phase I-B Specifications

For Phase I-B, the benchmark specifies that each participant must produce depletion results for three cases: 1B, 4B, and 7B. These are the same as cases 1A, 4A, and 7A, but with depletion steps added. The benchmark assumes that depletion occurs only in the fuel and burnable poisons and that the depletion performs under the critical spectrum assumption.

Table 3.2: Description of the Fluoride-Salt-Cooled High-Temperature Reactor benchmark Phase I-A cases [2].

Case	Description
1A	Reference case. Hot full power (HFP), with temperatures of 1110K for fuel kernel and 948K for coolant and all other materials (including TRISO particle layers other than fuel kernel). Nominal (cold) dimensions, 9 wt% enrichment, no burnable poison, control rod out.
2AH	Hot zero power (HZP) with uniform temperature of 948 K, otherwise same as Case 1A. Comparison with Case 1A provides HZP-to-HFP power defect.
2AC	Cold zero power (CZP). Same as Case 2AH, but with uniform temperature of 773 K. Comparison with Case 2AH provides isothermal temperature coefficient.
3A	Control rod inserted, otherwise same as Case 1A.
4A	Discrete europia burnable poison, otherwise same as Case 1A.
4AR	Discrete europia burnable poison and control rod inserted, otherwise same as Case 1A.
5A	Integral (dispersed) europia burnable poison, otherwise same as Case 1A.
6A	Increased heavy metal (HM) loading (4 to 8 layers of TRISO) decreased C/HM ratio (from about 400 to about 200) and decreased specific power to 100 W/gU, otherwise same as Case 1A.
7A	Fuel enrichment 19.75 wt%, otherwise same as Case 1A.

3.3 Benchmark Phase I Results

Several organizations participated in the benchmark with various Monte Carlo and Deterministic neutronics software, such as Serpent [46], OpenMC [70], and WIMS [48]. The University of Illinois at Urbana-Champaign (UIUC) participated in the benchmark with the OpenMC Monte Carlo code [70] and the ENDF/B-VII.1 material library [12]. The UIUC team consists of myself and my advisor, Dr. Kathryn Huff. I contributed Methodology, Software, Validation, Formal Analysis, and Visualization, while Dr. Huff contributed Resources, Supervision, and Funding acquisition. The `fhr-benchmark` Github repository contains all the results submitted by UIUC for the FHR benchmark [13]. The benchmark used a phased blind approach – participants were asked to submit Phase I-A and I-B results without knowledge of other submissions. Petrovic et al. [64] describes the preliminary results of the benchmark results across several institutions and concludes that the overall observed agreement is satisfactory. In the subsequent sections, I will share the results obtained by UIUC.

Table 3.3: University of Illinois at Urbana-Champaign's Fluoride-Salt-Cooled High-Temperature Reactor Benchmark Phase I-A results [13].

Case	**Summary**	**WCT [hr]**	k_{eff}*	β_{eff}**	**Fuel** $\frac{\Delta\rho}{\Delta T}$	**FliBe** $\frac{\Delta\rho}{\Delta T}$	**Graphite** $\frac{\Delta\rho}{\Delta T}$
1A	Reference	2.82	1.39389	0.006534	-2.24±0.15	-0.15±0.15	-0.68±0.15
2AH	HZP	2.82	1.40395	0.006534	-3.14±0.15	-0.20±0.14	-0.85±0.14
2AC	CZP	2.75	1.41891	0.006534	-3.36±0.14	-0.11±0.14	0.07±0.14
3A	CR	2.49	1.03147	0.006534	-4.03±0.28	-0.83±0.27	-3.18±0.29
4A	Discrete BP	5.08	1.09766	0.006542	-4.06±0.24	-1.55±0.23	-6.51±0.24
4AR	Discrete BP + CR	4.59	0.84158	0.006553	-5.60±0.49	-1.78±0.46	-10.44±0.47
5A	Dispersed BP	2.33	0.79837	0.006556	-5.09±0.40	-4.87±0.40	-22.99±0.38
6A	Increased HM	3.52	1.26294	0.006556	-4.46±0.19	0.16±0.20	-0.39±0.20
7A	19.75% Enriched	2.21	1.50526	0.006530	-2.49±0.13	-0.12±0.12	-0.62±0.12

BP: burnable poison, CR: control rod

* All k_{eff} values have an uncertainty of 0.00010.

** All β_{eff} values have an uncertainty of 0.000001.

3.3.1 Results: Phase I-A

In a recently submitted American Nuclear Society (ANS) M&C 2021 conference paper (which I am a co-author on), Petrovic et al. [64] compared FHR benchmark participants' Phase I-A k_{eff} results. We reported that the standard deviation between participants for each case was in the 231 to 514 pcm range, acceptable and notably close given a blind benchmark, assuring that UIUC's Phase I-A results are acceptable and in agreement with other benchmark participants.

Results: Effective Multiplication Factor (a)

Table 3.3 reports Phase I-A k_{eff} and reactivity coefficients results. I ran the simulations on UIUC's BlueWaters supercomputer with 64 XE nodes, which each have 32 cores [55]. To reduce the statistical uncertainty of k_{eff} to ~10pcm, I ran each simulation with 500 active cycles, 100 inactive cycles, and 200000 neutrons. Each simulation took wall-clock-time (WCT) ranging from 2 to 5 hours.

Cases 2AH and 2AC are at zero power, meaning that the fuel assembly is exactly

critical but not producing any energy. For both cases, k_{eff} is higher than the reference Case 1A, which I attribute to lower fuel temperatures. At lower fuel temperatures, less doppler broadening occurs, resulting in less neutron capture, thus, increasing k_{eff}. As expected, k_{eff} is lower for Cases 3A, 4AR, and 5A than reference case 1A since these cases introduce burnable poisons and control rods to the fuel assembly. Also, as expected, k_{eff} is higher for Case 7A than reference Case 1A, since it has a higher enrichment. However, Case 6A deviated from expectations with a lower k_{eff} despite an increase in heavy metal loading. This behavior is due to reduced moderation and worsened fuel utilization brought about by self-shielding, demonstrating that an increase in fuel packing fraction does not always correspond with an increased k_{eff}.

Results: Reactivity Coefficients (b)

β_{eff} increased by 10–20[pcm] for Cases 4A, 4AR, 5A, and 6A compared to reference Case 1A due to the introduction of control rods and poisons that shift the average neutron velocity to higher values, resulting in decreased thermal fission and increased fast fission [86]. Table 3.3 reports that most of the temperature coefficients are negative, exemplifying the AHTR's passive safety behavior. Negative reactivity feedback results in a self-regulating reactor; if the reactor power rises, resulting in temperature increase, the negative reactivity reduces power.

Results: Fission Source Distribution (c)

Figure 3.7 shows Cases 1A and 3A's fission source distribution discretized by one-fifth fuel stripe. Case 4AR has a similar fission source distribution to Case 3A since both cases have control rod insertion. Case 7A has a similar fission source distribution to case 6A since both have higher heavy metal loading. All other cases have similar fission source distributions to Case 1A.

For Case 1A, intuitively, one might assume that the highest fission source would occur

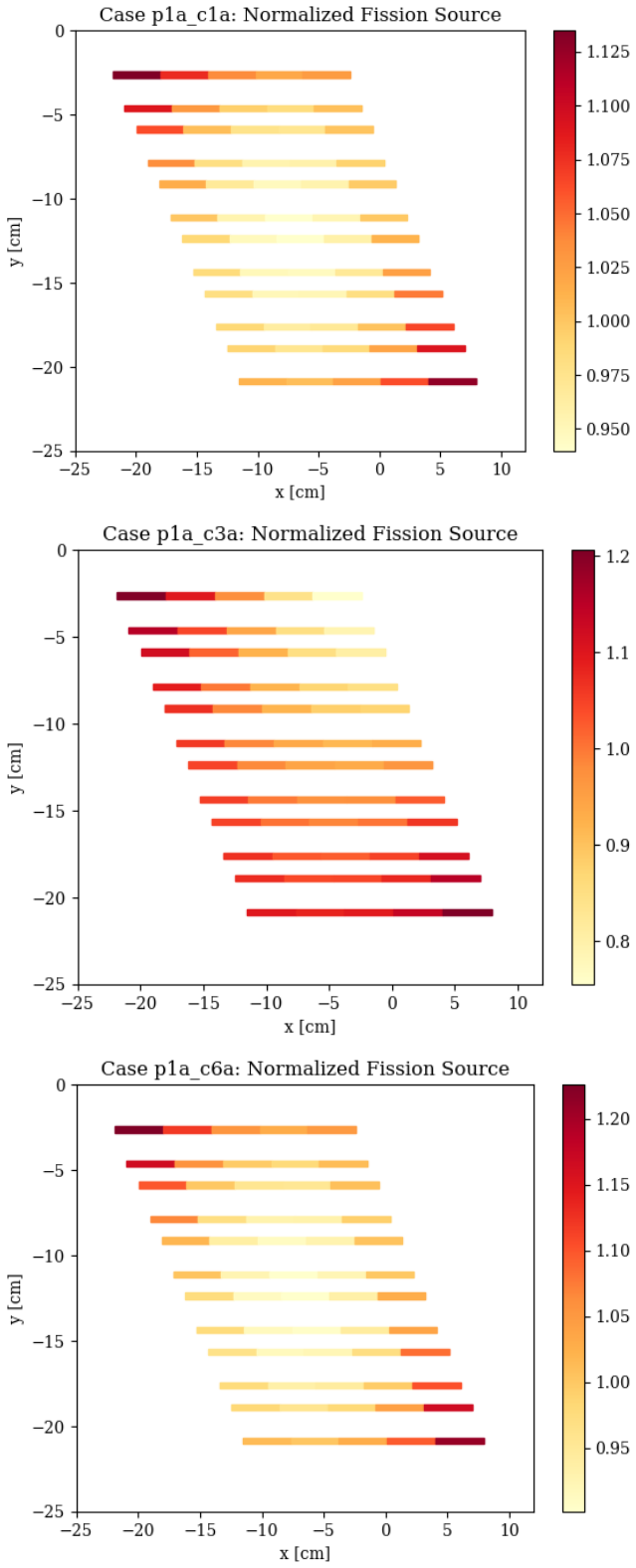


Figure 3.7: University of Illinois at Urbana-Champaign results: Normalized Fission Source Distribution [-] per one-fifth fuel stripe for Fluoride-Salt-Cooled High-Temperature Reactor Benchmark's Phase I-A Case 1A (top), Case 3A (middle), and Case 6A (bottom).

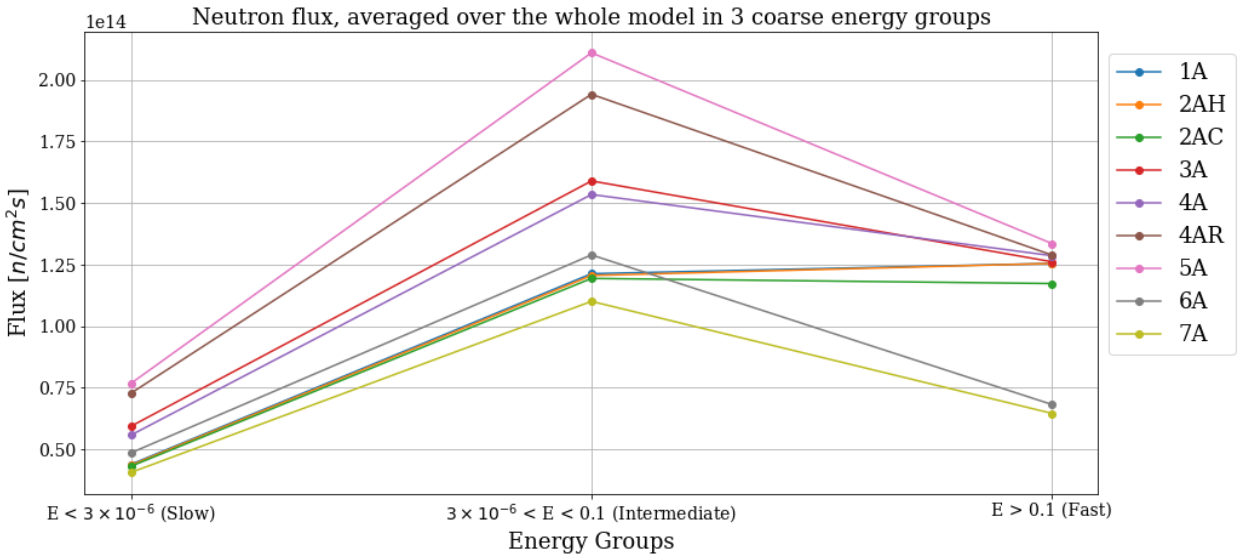


Figure 3.8: University of Illinois at Urbana-Champaign results: Fluoride-Salt-Cooled High-Temperature Reactor Benchmark's Neutron flux, averaged over the whole model, tabulated in three coarse energy groups for each Phase I-A case.

in the center of the diamond fuel segment; however, the opposite is true. Power peaking occurs on exterior stripes and is minimum on the interior stripes. Gentry et al. [28] reported similar power peaking phenomena towards the lattice cell's exterior closest to the Y-shaped carbon support structure. This fission source distribution is caused by diminished resonance escape probability in the interior due to the higher relative fuel-to-carbon volume ratio.

For Case 3A with an inserted control rod, the fission source is lower in the one-fifth stripes closer to the control rod. Case 6A demonstrates a further diminished fission source in the interior stripes due to the higher fuel-to-carbon ratio. This is seen in Figure 3.7 in which case 1A and 6A have similar fission distribution shapes, but case 6A has a bigger fission source value range.

Results: Average Neutron Flux (d)

Figure 3.8 shows the average neutron flux in the fuel assembly in three coarse energy groups. Most of the cases have the most flux in the intermediate group, followed by the thermal group, and the least flux in the fast group.

Results: Neutron Flux Distribution (e)

Figure 3.9 shows the neutron flux distribution in a 100 × 100 mesh for Cases 1A, 3A, and 6A for three coarse energy groups. For all three cases, fast-flux peaks in the diamond-shaped sectors containing the fuel stripes, whereas thermal flux peaks outside of the diamond-shaped sectors. This can be attributed to fission occurring at thermal energies in the fuel stripe area. For Case 3A, the thermal and intermediate neutron flux is depressed in the fuel assembly's control rod region. An increased heavy metal loading in Case 6A results in a more pronounced fast-flux peaking and thermal flux dip in the fuel stripe area.

Results: Neutron Spectrum (f)

Figure 3.10 shows the neutron spectrum for Cases 1A and 6A. Case 7A has a similar neutron spectrum to Case 6A since both cases have higher fuel content. All other cases have a similar neutron spectrum to Case 1A. The neutron spectra in Cases 6A and 7A are faster due to more heavy metal loading and higher enrichment, respectively.

3.3.2 Results: Phase I-B

Figure 3.11 shows the k_{eff} evolution during depletion for Cases 1B, 4B, and 7B. The k_{eff} at zero burnup corresponds to each case's corresponding Phase I-A k_{eff} value reported in Table 3.3. Case 1B is the reference case with 9% fuel enrichment and no burnable poisons (BPs). Case 1B's k_{eff} steadily decreases until it reaches 0.967845 at the final 70 GWd/tU burnup. Case 4B includes burnable poisons resulting in a lower initial k_{eff}. Its k_{eff} decreases at a slower rate in the beginning due to the presence of burnable poisons, which decreases the flux in the core. At approximately 20 GWd/tU, k_{eff} begins decreasing at a faster rate, presumably due to burn-up of the poison material. Case 7B has a 19.75% fuel enrichment, resulting in a higher initial k_{eff}. With a higher enrichment, the fuel can achieve a final burnup of 160 GWd/tU.

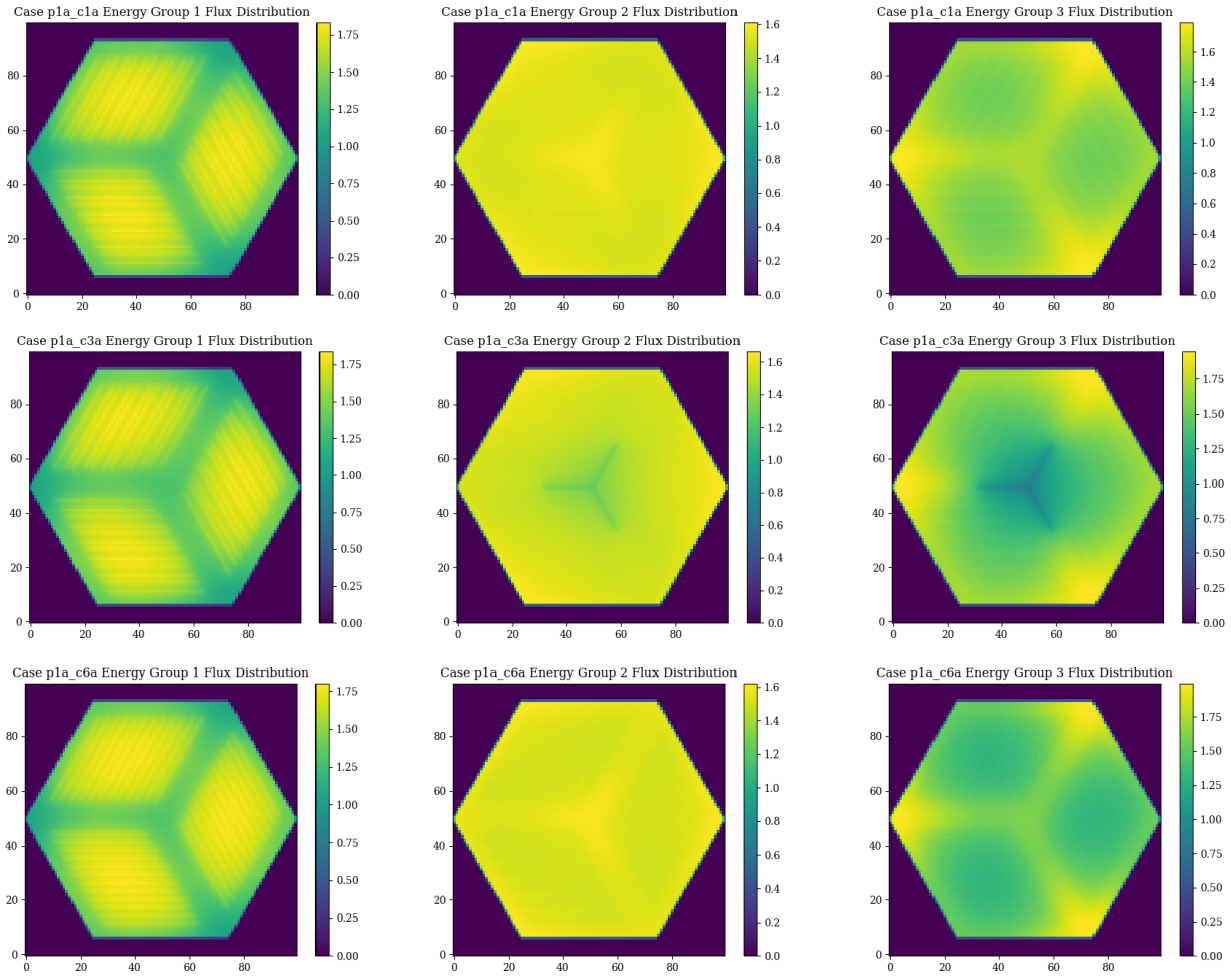


Figure 3.9: University of Illinois at Urbana-Champaign results: Fluoride-Salt-Cooled High-Temperature Reactor Benchmark neutron flux distribution in 100 × 100 mesh for three coarse energy groups: Case 1A (above), Case 3A (middle), Case 6A (below). Energy group 1: $E > 0.1$ MeV, Energy group 2: $3 \times 10^{-6} < E < 0.1$ MeV, Energy group 3: $E < 3 \times 10^{-6}$ MeV.

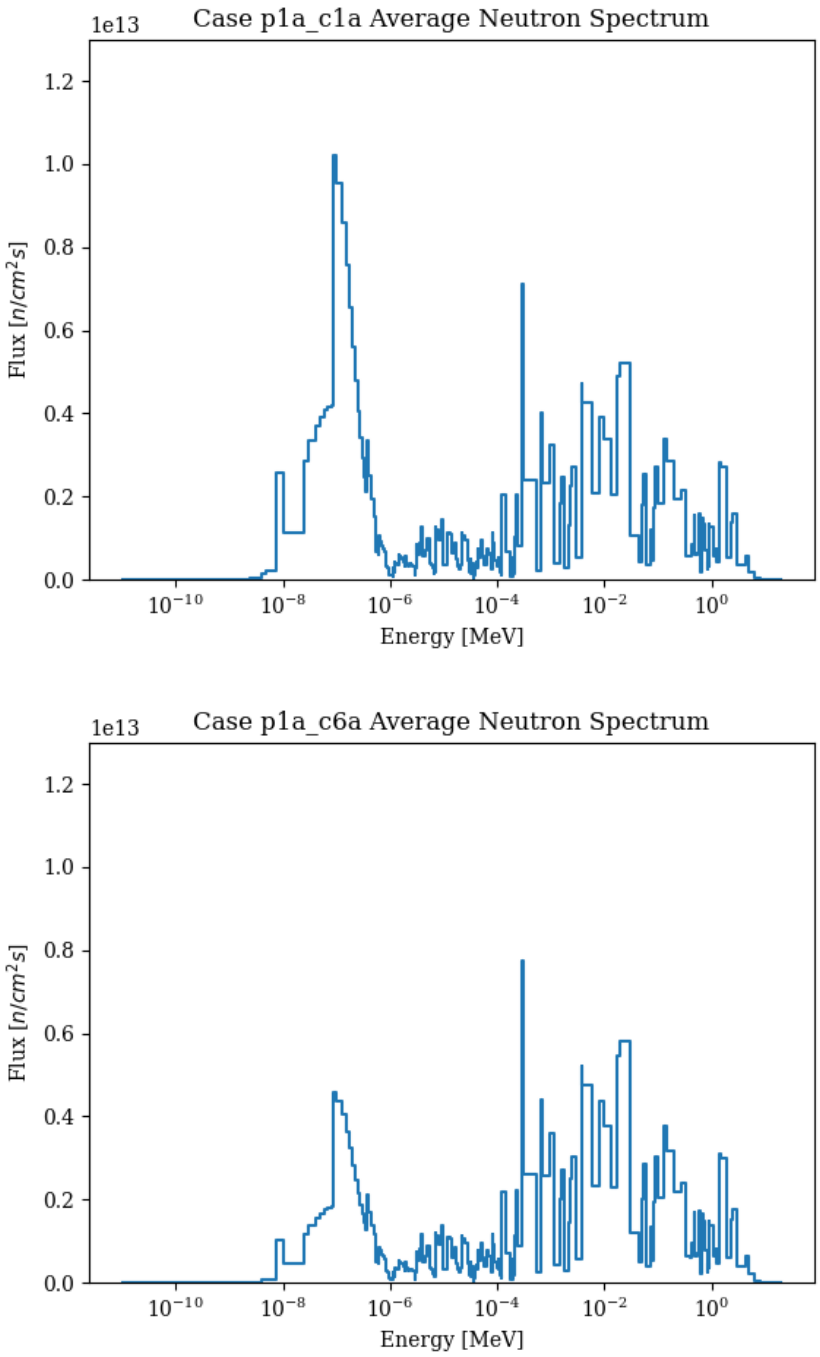


Figure 3.10: University of Illinois at Urbana-Champaign results: Neutron spectrum for Fluoride-Salt-Cooled High-Temperature Reactor Benchmark Phase I-A Case 1A (left) and Case 6A (right).

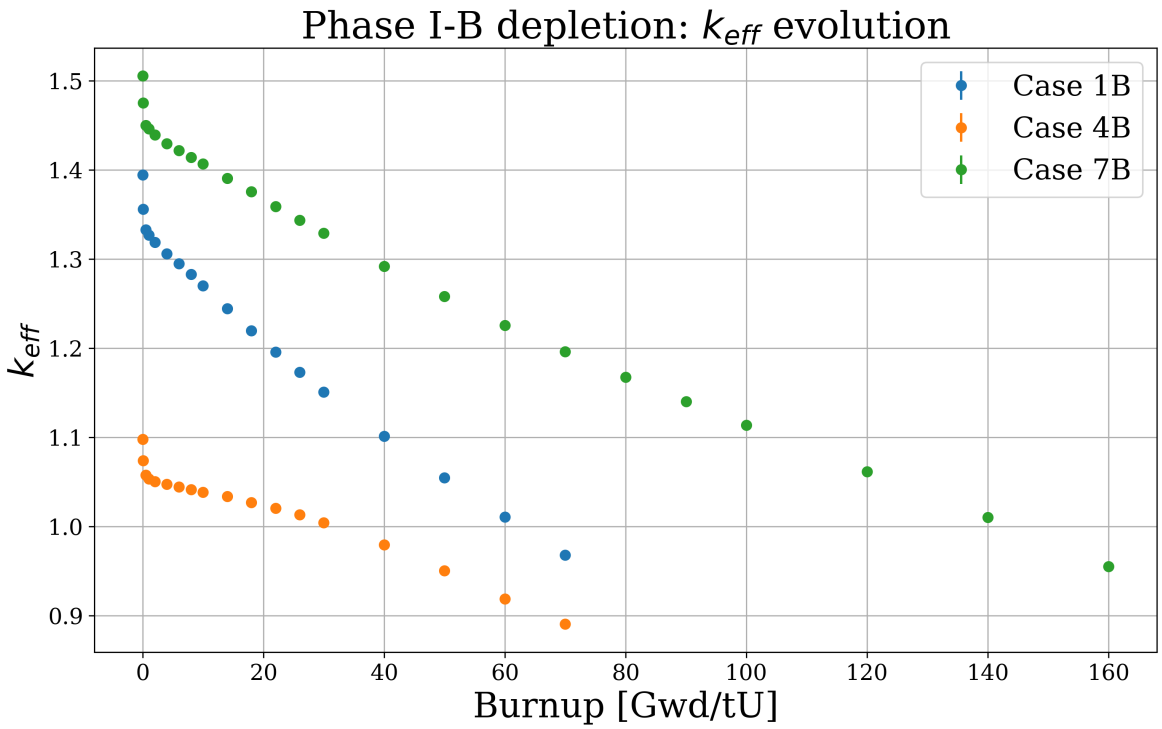


Figure 3.11: University of Illinois at Urbana-Champaign results: Fluoride-Salt-Cooled High-Temperature Reactor Benchmark Phase I-B depletion k_{eff} evolution for Cases 1B, 4B, and 7B. Case 1B is the reference case, Case 4B is the discrete burnable poison case, and Case 7B is the 19.75% enrichment case. Error bars are included but are barely visible due to the low ~40pcm uncertainty.

3.4 Summary

This chapter described the FHR benchmark specifications, AHTR design, and Phase I-A and I-B results obtained by the UIUC team, consisting of myself and my advisor. The benchmark results highlight the AHTR's passive safety behavior with negative temperature coefficients. Results such as a lower k_{eff} for the AHTR configuration with higher heavy metal loading demonstrated that increased fuel packing does not always correspond with increased k_{eff} due to self-shielding effects. These results hint at the possibility of minimizing fuel required by optimizing for heterogenous fuel distributions within the core. This will be further explored in the later chapters.

Chapter 4

ROLLO: Reactor evOLutionary aLgorithm Optimizer

In this chapter, I introduce the ROLLO framework developed as preliminary work for the proposed PhD scope. ROLLO is a Python package that applies evolutionary algorithm techniques to optimize nuclear reactor design. Applying evolutionary algorithms to nuclear design problems is not new, as I previously discussed in Section 2.3, and available evolutionary algorithm packages can be customized for reactor design optimization problems. However, evolutionary algorithm setup is highly customizable with an assortment of genetic algorithm designs and operators. A reactor designer unfamiliar with evolutionary algorithms will have to go through the cumbersome process of customizing a genetic algorithm for their needs and determine which operators and hyperparameters work best for their problem. Furthermore, computing fitness values with nuclear software is computationally expensive, necessitating using supercomputers and setting up parallelization for the genetic algorithm.

Therefore, the motivation behind creating ROLLO is to limit these inconveniences and facilitate using evolutionary algorithms for reactor design optimization. ROLLO provides a general genetic algorithm framework, sets up parallelization for the user, and promotes usability with an input file that only exposes mandatory parameters. ROLLO also strives to be effective, flexible, open-source, parallel, reproducible, and usable. I briefly summarize how ROLLO achieves these goals:

- Effective: ROLLO is well documented, tested, and version-controlled on Github [**?**].
- Flexible: The proposed work aims to utilize ROLLO to explore arbitrary reactor geometries and heterogeneous fuel distributions. However, future users might want to

utilize ROLLO to explore other arbitrary design parameters. Thus, I designed the ROLLO framework accordingly. The user can vary any imaginable parameter because ROLLO uses a templating method to edit the input file of the coupled software.

- Open-source: ROLLO utilizes a well-documented, open-source evolutionary algorithm Python package to drive the optimization process, and established open-source nuclear software (OpenMC [70] and Moltres [50]) to compute the objective function and constraints. I also provide a simple tutorial for future developers to follow for coupling other nuclear software to ROLLO.

- Parallel: Users have the option to run ROLLO in parallel using either the `multiprocessing_on_dill` or `mpi4py` Python packages [16].

- Reproducible: Data from every ROLLO run saves into a unique, pickled file (pickle is a Python module that serializes Python objects), and all results from this work are available on Github.

ROLLO essentially couples an evolutionary algorithm driver with nuclear software, such as neutron transport and thermal-hydraulics codes. Figure 2.3 from Chapter 2 outlines the evolutionary algorithm iterative problem solving process. I modified Figure 2.3 to produce Figure 4.1, which depicts how the nuclear transport and thermal-hydraulics software fit within the process. Therefore, ROLLO initially reads and validates the JSON input file, initializes the Distributed Evolutionary Algorithms in Python (DEAP) [24] genetic algorithm hyperparameters and operators, and finally runs the genetic algorithm following the flow chart in Figure 4.1, in which the nuclear software evaluates each individual's fitness.

In the subsequent sections, I describe the evolutionary algorithm software (DEAP) that drives ROLLO, the nuclear software coupled to ROLLO, and details about the ROLLO framework, such as the input file format and software architecture.

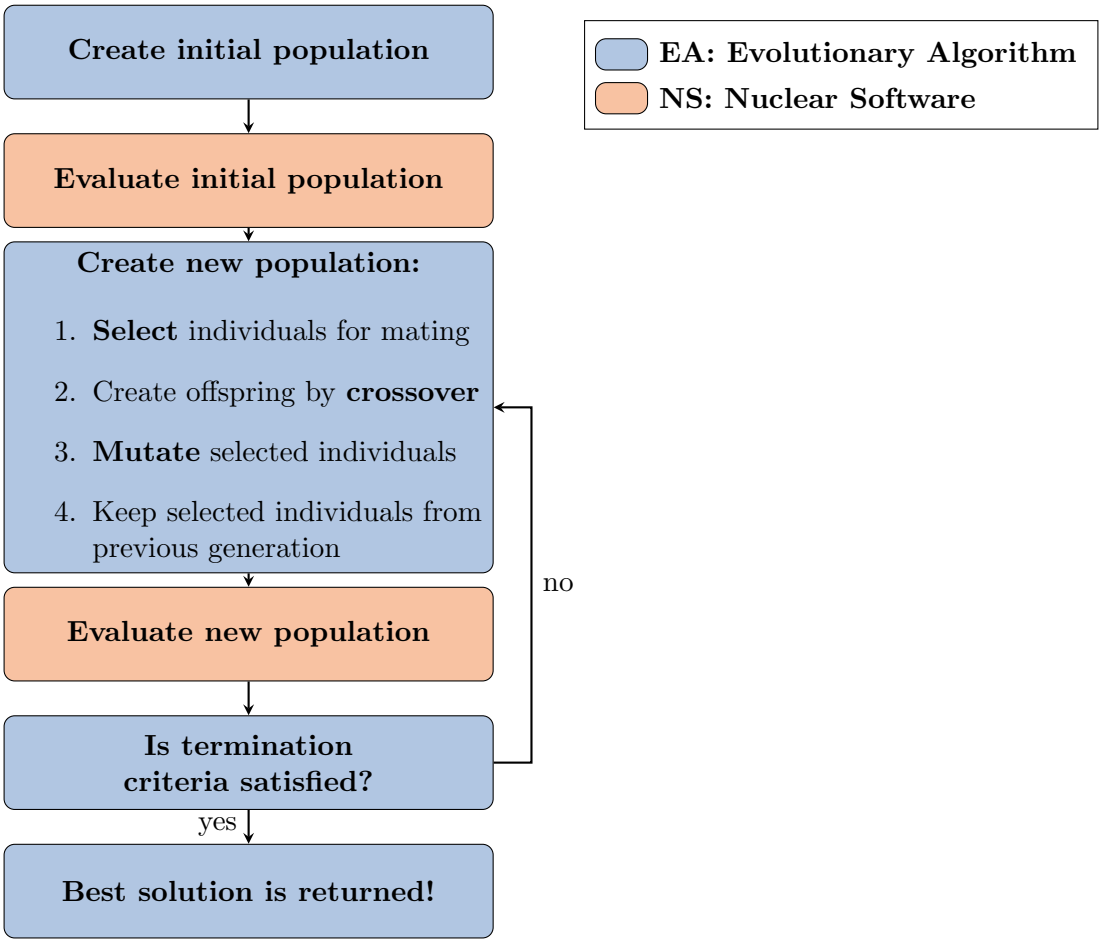


Figure 4.1: Process of finding optimal solutions for a problem with a genetic algorithm. Nuclear software evaluates each new population.

4.1 Evolutionary Algorithm Driver

Evolutionary algorithm computation uses sophisticated, diverse techniques and mechanisms, resulting in even the most well-designed, software frameworks being complicated under the hood. Utilizing an existing evolutionary algorithm framework presents implementation challenges as the user must edit the framework's source code to customize for their application and hyperparameters [24]. Therefore, a computation framework that gives the user the capability to build custom evolutionary algorithms is ideal for this project.

Many evolutionary algorithm computation packages exist: DEAP [24], inspyred [26], Pyevolve [62], and OpenBEAGLE [25]. DEAP is the newest package and places a high value on code compactness and clarity [24]. DEAP is the only framework that allows the user to prototype evolutionary algorithms rapidly and define custom algorithms without digging deep into the source code to modify hyperparameters and their application methods. Accordingly, I chose DEAP to drive the ROLLO framework's evolutionary algorithm component. DEAP provides building blocks for each optimizer function and allows the user to customize a specialized algorithm to fit their project [24].

4.1.1 Distributed Evolutionary Algorithms in Python

DEAP is composed of two simples structures: a *creator* and a *toolbox*. The *creator* module allows the run-time creation of classes via inheritance and composition, enabling individual and population creation from from any data structure: lists, sets, dictionaries, trees, etc [24]. The *toolbox* is a container that the user manually populates. In the *toolbox*, the user defines the selection, crossover, and mutation operator types and hyperparameters. For example, the user registers a crossover operator under the 'mate' alias, and a selection operator under the 'select' alias. Then, the evolutionary algorithm uses these aliased operators from the *toolbox*. If the user wants to change the crossover operator, they would update the 'mate' alias in the *toolbox*, while keeping the evolutionary algorithm unchanged [24].

```
1         from deap import creator, base, tools, algorithms
2         creator.create("Objective", base.Fitness, weights=(-1.0,)) # minimum
3         creator.create("Individual", list, fitness=creator.Objective)
4
5         toolbox = base.Toolbox()
6         toolbox.register("variable_1", random.uniform, 0.0, 10.0)
7         toolbox.register("variable_2", random.uniform, -1.0, 0.0)
8         def individual_creator():
9             return creator.Individual([toolbox.variable_1(), toolbox.variable_2()])
10        toolbox.register("individual", individual_creator())
11        toolbox.register("population", tools.initRepeat, list, toolbox.individual)
12        def evaluator_fn(individual):
13            return tuple([sum(individual)])
14        toolbox.register("evaluate", evaluator_fn)
15        toolbox.register("select", tools.selBest, k=5)
16        toolbox.register("mutate", tools.mutPolynomialBounded, eta=0.5, low=[0, -1], up=[-1, 0])
17        toolbox.register("mate", tools.cxOnePoint)
```

Figure 4.2: DEAP sample code demonstrating the usage of the *creator* and *toolbox* modules to initialize the genetic algorithm. In ROLLO, DEAP's *creator* and *toolbox* modules are initialized in the source code based on the genetic algorithm parameters defined by the user in the ROLLO input file.

Figure 4.2 illustrates DEAP's usage of the *creator* and *toolbox* modules. Line 2 creates a single-objective fitness class, `Objective`. The first argument defines the name of the derived class, the second argument specifies the inherited base class, `base.fitness`, and the third argument indicates the objective fitness (−1.0 indicates a minimum objective, +1.0 indicates a maximum objective). Line 3 derives an `Individual` class from the standard Python list type, and defines its fitness attribute to be the newly created `Objective` object. Lines 5-9 initialize the DEAP toolbox, register `variable_1` and `variable_2` with their upper and lower bounds, and defines the `individual_creator` function to return an `Individual` initialized with `variable_1`, and `variable_2`. Lines 10-11 and 14-17 are aliases for initializing individuals and population, specifying variation operators (`select`, `mutate`, `mate`), and evaluating individual fitness (`evaluate`) [24]. Lines 12-13 define the evaluation function that returns the fitness values.

ROLLO initializes DEAP's *creator* and *toolbox* modules based on the genetic algorithm parameters defined by the user in the ROLLO input file. The evaluation function runs the nuclear software and returns user-defined fitness values.

4.1.2 General Genetic Algorithm Framework

DEAP creators provided variations of a classical genetic algorithm exposing different explicitness levels [24]. The high-level examples use the built-in DEAP genetic algorithms, whereas the low-level example completely unpacks the genetic algorithm to expose a generational loop. The general genetic algorithm included in the ***Algorithm*** class is based on the low-level example. The algorithm begins by initializing the starting population and evaluating each individual's fitness value. Then, it enters a generational loop. During each iteration, selection, mating, and mutation operators are applied to the population, then, the new individuals are evaluated, the constraints are applied, and the results are saved.

4.2 Nuclear Software

Many nuclear software tools, applications, libraries, packages have restricted public access. In the preliminary work, I enabled ROLLO to work with open-source nuclear transport and thermal-hydraulics software, OpenMC [70] and Moltres [50]. OpenMC is an open-source Monte Carlo neutron transport code capable of performing k-eigenvalue calculations on models built using either constructive solid geometry or CAD representation. OpenMC can run in parallel using a hybrid Message Passing Interface (MPI) and OpenMP programming model. Moltres is an open-source tool designed to simulate MSRs using deterministic neutronics and thermal-hydraulics implemented as an application atop the Multiphysics Object-Oriented Simulation Environment (MOOSE) finite-element framework. Moltres solves arbitrary-group neutron diffusion, temperature, and precursor governing equations on a single mesh and can be deployed on an arbitrary number of processing units [50].

OpenMC and Moltres are both open-source, well-documented, well-supported, and Github version-controlled codes that can run in parallel on HPC machines. Thus they achieve the ROLLO goals listed at the start of this chapter, making them suitable to be used as ROLLO's nuclear dependency. However, users can easily use restricted nuclear software if they so desire with ROLLO by coupling ROLLO with the restricted software on their local machine. In the ROLLO documentation [**?**], I outline how to couple other nuclear software to ROLLO.

4.3 ROLLO Input File

ROLLO's input file is in JSON format. There are four sections that the user must define: `control_variables`, `evaluators`, `constraints`, and `algorithm`. Figure 4.3 shows an example ROLLO input file. In this simulation, ROLLO uses a genetic algorithm with the defined hyperparameters to minimize the `output1` parameter which is calculated using the OpenMC evaluator that accepts input parameters: `variable1` and `variable2`.

Next, I will describe how to define each section of a ROLLO input file. The ROLLO documentation [**?**] provides further descriptions for setting up a ROLLO input file.

4.3.1 Control Variables

Control variables are parameters the genetic algorithm will vary. For each control variable, the user must specify its minimum and maximum values. For example, Lines 2 to 5 in Figure 4.3 demonstrate that the control variables, `variable1` and `variable2`, will be varied from 0 to 10 and -1 to 0, respectively.

4.3.2 Evaluators

Evaluators are the nuclear software ROLLO utilizes to calculate objective functions. Presently, only `openmc` and `moltres` evaluators are available in ROLLO. In a single ROLLO input file,

```
{
    "control_variables": {
        "variable1": {"min": 0.0, "max": 10.0},
        "variable2": {"min": -1.0, "max": 0.0}
    },
    "evaluators": {
        "openmc": {
            "input_script": "openmc_inp.py",
            "output_script": "openmc_output.py",
            "inputs": ["variable1", "variable2"],
            "outputs": ["output1", "output2"]
        }
    },
    "constraints": {
        "output1": {"operator": [">=", "<"], "constrained_val": [1.0, 1.5]}
    },
    "algorithm": {
        "objective": ["min"],
        "optimized_variable": ["output1"],
        "pop_size": 100,
        "generations": 10,
        "mutation_probability": 0.23,
        "mating_probability": 0.46,
        "selection_operator": {"operator": "selTournament", "inds": 15, "tournsize":5},
        "mutation_operator": {
            "operator": "mutPolynomialBounded",
            "indpb": 0.23,
            "eta": 0.23
        },
        "mating_operator": {"operator": "cxBlend", "alpha": 0.46}
    }
}
```

Figure 4.3: Reactor evOLutionary aLgorithm Optimizer (ROLLO) sample JSON input file.

```
1  import openmc
2  # templating
3  variable1 = {{variable1}}
4  variable2 = {{variable2}}
5  # run openmc
6  ...
```

```
1  import openmc
2  # templating
3  variable1 = 3.212
4  variable2 = -0.765
5  # run openmc
6  ...
```

Figure 4.4: `openmc_inp.py` input script template (left). Templated `openmc_inp.py` with variable1 and variable2 values defined (right).

a user may define any number of evaluators. For each evaluator, mandatory input parameters are `input_script`, `inputs`, and `outputs`, and the optional input parameters are `output_script` and `keep_files`. The `input_script` is input file template's name for the evaluator software. The user must include a input file template in the same directory as the ROLLO input file. The `inputs` parameter lists the control variables that are placed into the input file template. ROLLO utilizes jinja2 templating to insert the control variable values into the `input_script`. Lines 6 to 12 in the ROLLO input file (Figure 4.3) demonstrate that `variable1` and `variable2` are `inputs` into the `openmc_inp.py input_script`. Figure 4.4 shows the template and templated openmc script; once the `openmc_inp.py input_script` is templated, {{`variable1`}} and {{`variable2`}} on Lines 3 and 4 will be replaced with values selected by the ROLLO genetic algorithm.

The `outputs` parameter lists the output variables that the evaluator will return to the genetic algorithm. These output parameters are also known as the objective functions used to evaluate the individual. ROLLO uses three methods to return an output parameter. First, if the output parameter is also an input parameter, ROLLO will automatically return the input parameter's value. Second, the user can use predefined evaluations. For example, in `OpenMCEvaluation`, there is a predefined k_{eff} evaluation. The user may also add predefined evaluations to `OpenMCEvaluation` or `MoltresEvaluation`, or any other coupled software's evaluation file. Third, the user may include an `output_script` that returns the desired output parameter. The `output_script` must include a line that prints a dictionary

containing the output parameters' names and their corresponding value as key-value pairs. The `keep_files` parameter accepts true or false, which directs ROLLO to save or not save each evaluations templated input file and output files.

4.3.3 Constraints

In the constraints section, the user can define constraints on any output parameter. Any individual that does not meet the defined constraints is removed from the population, encouraging the proliferation of individuals that meet the constraints. For each constrained output parameter, the user lists the `operator`s and `constrained_val`s as in Line 15 of the ROLLO input file (Figure 4.3). Thus, for this ROLLO simulation, `output_1` is constrained to be >= 1.0 and < 1.5.

4.3.4 Algorithm

In the algorithm section, the user defines all the hyperparameters for the genetic algorithm. The mandatory input parameters include `optimized_variable`, `objective`, `pop_size`, and `generations`. The user may define a single or multi objective optimization problem with the `optimized_variable` and `objective` parameters. The user specifies a list of `optimized_variable`s, which must be output parameters from the evaluators' `outputs`. The user has the option to maximize or minimize each of the `optimized_variable`s by defining the `objective` parameter as a list of `max` or `min` values which correspond to the variable order in the `optimized_variable` parameter. The user must also specify the population size (`pop_size`) and number of generations (`generations`) in the genetic algorithm.

The optional input parameters include `parallel`, `mutation_probability`, `mating_probability`, `selection_operator`, `mutation_operator`, and `mating_operator`. There are three options for the `parallel` parameter: *none*, *multiprocessing*, and *mpi_evals*. The *none* option results in ROLLO running without parallelization, and the *multiprocess-*

Table 4.1: Selection, mutation, and mating operators available in Reactor evOLutionary aLgorithm Optimizer (ROLLO) and their corresponding hyperparameters.

Operator	**Available Options**	**Hyperparameters**
Selection	`selTournament`	`tournsize`: no. of individuals in each tournament `inds`: no. of individuals to select
	`selNSGA2`	`inds`: no. of individuals to select
	`selBest`	`inds`: no. of individuals to select
Mutation	`mutPolynomialBounded`	`eta`: crowding degree of the mutation `indpb`: independent probability for each attribute to be mutated
Mating	`cxOnePoint`	-
	`cxUniform`	`indpb`: independent probability for each attribute to be exchanged
	`cxBlend`	`alpha`: Extent of the interval in which the new values can be drawn for each attribute on both side of the parents attributes

ing option results in ROLLO using the `multiprocessing_on_dill` Python module to run ROLLO's nuclear software evaluations in parallel. The *mpi_evals* option is specially created for supercomputer parallelization, and will be further discussed in Section 4.4.2.

As mentioned previously in Section 2.4.1, it is important to select genetic algorithm hyperparameters that balance the extent of exploration and exploitation. The user can define the mutation and mating probability or use default values of 0.23 and 0.46, respectively. For each operator, the user can choose from a list of operators and define each of their required hyperparameters. Table 4.1 shows the available operators and their respective hyperparameters. The default selection operator is `selTournament` with a default `inds` value of 15 and `tournsize` value of 5. The default mutation operator is `mutPolynomialBounded` with default `eta` and `indpb` values of 0.23. The default mating operator is `cxBlend` with a default `alpha` of 0.46. Lines 17 to 31 in the example ROLLO input file (Figure 4.3) demonstrate `algorithm` specifications.

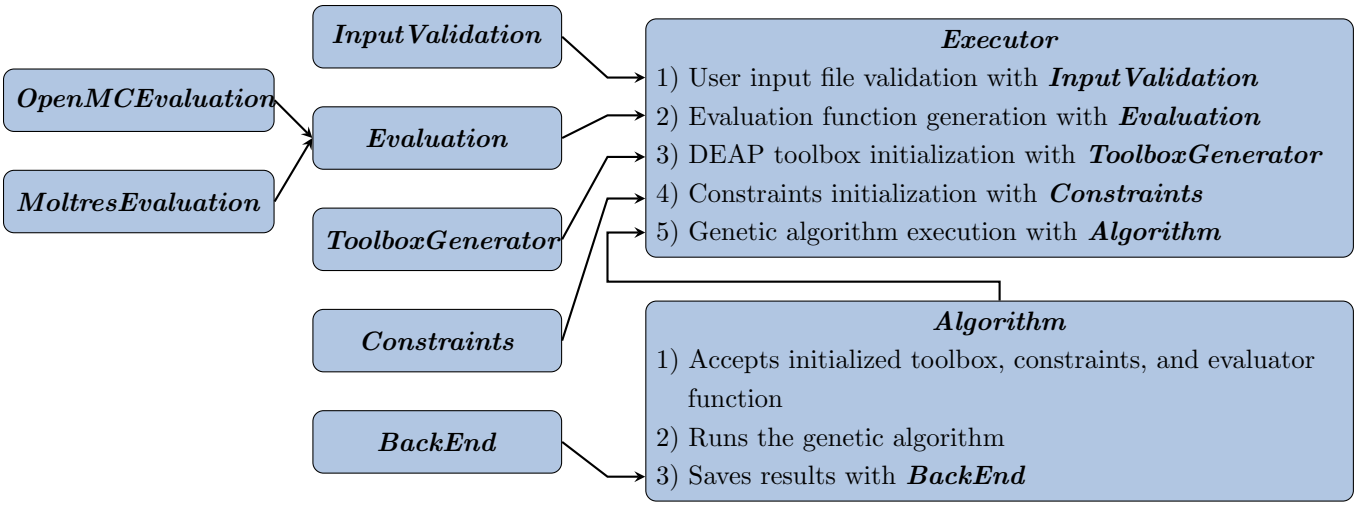


Figure 4.5: Visualization of ROLLO architecture.

4.4 ROLLO Software Architecture

In this section, I will describe the ROLLO v1.0 software architecture and how all the parts come together to optimize reactor design. Table 4.2 outlines the classes in the ROLLO software and describes each class's purpose. Figure 4.5 depicts the ROLLO software architecture. When the user runs a ROLLO input file, the ***Executor*** class drives ROLLO's execution from beginning to end. The ***Executor*** calls ***InputValidation*** to parse the input file to ensure that the user defined all mandatory parameters and used the correct formatting. Next, it initializes an ***Evaluator*** object based on the `evaluators` specifications in the input file. It uses the ***Evaluator*** object to create a function that will run each evaluator software with the desired input parameters and return the output parameters calculated by the evaluator software. Next, it uses the ***ToolboxGenerator*** to create an initialized DEAP toolbox object based on the input file's `algorithm` specifications. The ***ToolboxGenerator*** object accepts the ***Evaluator*** object and registers it as the toolbox's 'evaluate' tool. Then, it initializes a ***Constraints*** object to contain `constraints` specified in the input file. Next, the ***Executor*** initializes an ***Algorithm*** object that accepts the initialized DEAP toolbox and ***Constraints*** object. Finally, the ***Executor*** class uses a method in the ***Algorithm*** object to run a general genetic algorithm with hyperparameters from the DEAP toolbox,

Table 4.2: Classes that make up the ROLLO architecture.

Class	Description
InputValidation	The ***InputValidation*** class contains methods to read and validate the JSON ROLLO input file to ensure the user defined all key parameters. If they did not, ROLLO raises an exception to tell the user which parameters are missing.
Evaluation	DEAP's fitness evaluator (as mentioned in Section 4.1.1) requires an evaluation function to evaluate each individual's fitness values. The ***Evaluation*** class contains a method that creates an evaluation function that runs the nuclear software and returns the required fitness values, defined in the input file.
OpenMCEvaluation	The ***OpenMCEvaluation*** class contains built-in methods for evaluating OpenMC output files. Developers can update this file with methods to evaluate frequently used OpenMC outputs.
ToolboxGenerator	The ***ToolboxGenerator*** class initializes DEAP's *toolbox* and *creator* modules with genetic algorithm hyperparameters defined in the input file.
Constraints	The ***Constraints*** class contains methods to initialize constraints defined in the input file and applies the constraints by removing individuals that do not meet the constraint.
BackEnd	The ***BackEnd*** class contains methods to save genetic algorithm population results into a pickled checkpoint file and to restart a partially completed genetic algorithm from the checkpoint file.
Algorithm	The ***Algorithm*** class contains methods to initialize and execute the genetic algorithm. It executes a general genetic algorithm framework that uses the hyperparameters defined in the ***ToolboxGenerator***, applies constraints defined in ***Constraints***, evaluates fitness values using the evaluation function produced by ***Evaluation***, and saves all the results with ***BackEnd***.
Executor	The ***Executor*** class drives the ROLLO code execution with the following steps: 1) User input file validation with ***InputValidation*** 2) Evaluation function generation with ***Evaluation*** 3) DEAP toolbox initialization with ***ToolboxGenerator*** 4) Constraint initialization with ***Constraints*** 5) Genetic algorithm execution with ***Algorithm***

apply constraints defined in the ***Constraints*** object, and calculate objective functions using the evaluation function created by the ***Evaluator*** object, all the while saving the results using the ***BackEnd*** class.

In the ROLLO Github repository [**?**], I include a tests directory that contains unit tests for all methods in the classes described above.

4.4.1 Installing and Running ROLLO

There are two ways to install ROLLO. First, a user can utilize The Python Package Index (PyPI) to install ROLLO: `python -m pip install rollo` [14]. Second, a user can download the ROLLO Github repository [**?**] and install it from source.

ROLLO is run from the command line interface. A user should first set up the ROLLO JSON input file and evaluator scripts in a directory. When running ROLLO from the command line, there is one mandatory argument and one optional argument. The mandatory argument is the input file (`-i`). The optional argument is the checkpoint file (`-c`). Thus, the structure of a command line input for running ROLLO is:

```
python -m rollo -i <input file name> -c <checkpoint file name>
```

The checkpoint file holds the results from the ROLLO simulation and also acts as a restart file. Thus, if a ROLLO simulation ends prematurely, the checkpoint file can be used to restart the code from the most recent population and continue the simulation. The user must include each evaluator software's input script template and optional output script in the same directory as the ROLLO input file.

4.4.2 ROLLO parallelization for High Performance Computers

The *mpi_evals* option is specially created for supercomputer parallelization, particularly the BlueWaters supercomputer [55]. The `multiprocessing_on_dill` Python module only parallelizes the ROLLO simulation across one BlueWaters node, thus, to enable parallelization

across multiple BlueWaters nodes, the `mpi4py` Python module is used. However, the user must ensure that the nuclear evaluation software does not utilize MPI for parallelization. I had to recompile OpenMC without MPI to successfully use the *mpi_evals* option. Therefore, *mpi_evals* uses MPI to run each nuclear software evaluation in parallel on different BlueWaters nodes.

4.4.3 ROLLO Results Analysis

The ***BackEnd*** class manages results from each ROLLO simulation. ***BackEnd*** puts all the results in a pickled dictionary, which is saved as `checkpoint.pkl` in the same directory as the input file. The checkpoint file can be loaded into a Jupyter notebook and organized to produce desired plots. Examples of ROLLO results analysis can be found in the ROLLO documentation [**?**].

The evaluation function creates a new directory for each software, generation, and individual and stores the templated input file and output files associated with that particular run. The generation and individual values are indexed by zero. For example, the directory containing files associated with an OpenMC run for the tenth individual in the genetic algorithm's third generation will be named: `openmc_2_9`. If the `keep_files` parameter in the `evaluators` section of the input file is set false, ROLLO will delete each directory once it extracts the fitness values.

4.5 Summary

This chapter described the Reactor evOLutionary aLgorithm Optimizer (ROLLO) framework developed as preliminary work for the proposed PhD scope. ROLLO is a Python package that applies evolutionary algorithm optimization techniques to nuclear reactor design using the Distributed Evolutionary Algorithms in Python (DEAP) module, OpenMC, and Moltres. The motivation for ROLLO is to enable reactor designers to utilize robust

evolutionary algorithm optimization methods without going through the cumbersome process of setting up a genetic algorithm framework, selecting appropriate hyperparameters, and setting up its parallelization. ROLLO is designed to be effective, flexible, open-source, parallel, reproducible, and usable. ROLLO is hosted on Github [**?**].

Chapter 5

AHTR Optimization Preliminary Work

This chapter demonstrates the preliminary work completed for AHTR optimization. I used ROLLO to apply genetic algorithms to maximize k_{eff} in a single AHTR fuel slab. Then, I presented spatial and energy homogenizations for applications to AHTR multiphysics simulations. The `dissertation-results` Github repository contains all the scripts, results, and plots shown in this chapter [**?**].

5.1 ROLLO Optimization: AHTR Fuel Slab

This demonstration problem explores how heterogenous fuel distributions impact k_{eff} compared with homogenous fuel distributions customary in most reactor designs. I use OpenMC v0.12.0 for these neutronics calculations with the ENDF/B-VII.1 data library [12].

5.1.1 Problem Definition

The reactor core explored is a single fuel slab from the FHR benchmark AHTR design. I modified the fuel slab to be straightened with perpendicular sides, instead of slanted as in Figure 3.4. Figure 5.1 illustrates the straightened fuel slab with periodic boundary conditions in the x-y axis. The periodic surfaces are: 1-3, 2-4. The slab has $27.1 \times 3.25 \times 1.85$ cm^3 dimensions with reflective top and bottom (along z-axis) boundary conditions. I use the same materials as in the FHR benchmark, except that I homogenized each TRISO particle's four outer layers: porous carbon buffer, inner pyrolytic carbon, silicon carbide layer, and the outer pyrolytic carbon. The TRISO particle dimensions remain the same. Table 5.1

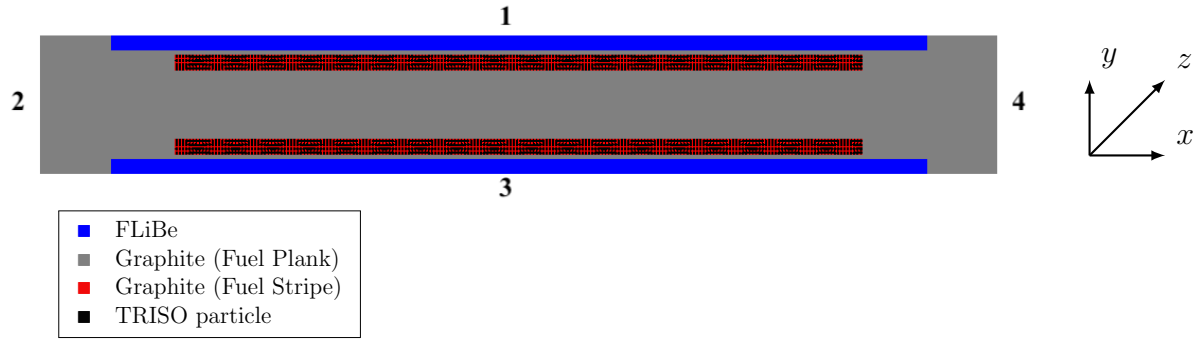


Figure 5.1: Straightened Advanced High Temperature Reactor (AHTR) fuel slab.

Table 5.1: Straightened Advanced High Temperature Reactor (AHTR) fuel slab k_{eff} for case with no TRISO homogenization and case with homogenization of the four outer layers. Both simulations were run on one BlueWaters XE Node.

TRISO Homogenization	k_{eff}	**Simulation time** [s]
None	1.38548 ± 0.00124	233
Four outer layers	1.38625 ± 0.00109	168

reports the k_{eff} for this original straightened AHTR configuration with and without the outer layer TRISO homogenization. The TRISO particle outer four-layer homogenization resulted in a 30% speed-up without compromising accuracy with k_{eff} values within each other's uncertainty.

The ROLLO optimization objective aims to maximize the slab k_{eff}. It does so by varying the TRISO particle packing fraction across the slab while keeping the total packing fraction constant at 0.0979. This total packing fraction is consistent with the original straightened slab with TRISO particles in fuel stripes (Figure 5.1). I divided the slab into ten cells along the x-axis between the FLiBe and graphite buffers, resulting in ten $2.31 \times 2.55 \times 1.85$ cm^3 cells. A sine distribution governs the TRISO particle packing fraction's distribution across cells:

$$PF(x) = (a \cdot sin(b \cdot x + c) + 2) \cdot NF \tag{5.1}$$

where

PF = packing fraction [−]

a = amplitude, peak deviation of the function from zero [−]

b = angular frequency, rate of change of the function argument $[\frac{radians}{cm}]$

c = phase, the position in its cycle the oscillation is at t = 0 $[radians]$

x = midpoint value for each cell $[cm]$

NF = Normalization factor [−]

The normalization factor ensures a consistent total packing fraction in the slab regardless the TRISO particle distribution. For example, a packing fraction distribution of $PF(x) = \big(0.5 \cdot sin(\frac{\pi}{3} \cdot x + \pi) + 2\big) \cdot NF$, results in the following packing fractions for the ten cells: 0.103, 0.120, 0.049, 0.138, 0.076, 0.081, 0.136, 0.048, 0.125, and 0.098. Figure 5.2 shows this sine distribution, highlights the packing fraction at the respective midpoints, and displays the slab's x-y axis view with packing fraction varying based on this sine distribution.

In ROLLO, a genetic algorithm varies the a, b, and c variables to find a combination that produces a packing fraction distribution that maximizes the slab's k_{eff}. I defined a, b, and c's upper and lower bounds as:

$$0 < a < 2$$

$$0 < b < \frac{\pi}{2}$$

$$0 < c < 2\pi$$

The bounds of a keep the sine distribution from falling below zero. The b and c variable bounds spread wide enough to allow the genetic algorithm to explore various sine distributions. The OpenMC evaluator calculates k_{eff}. OpenMC runs each simulation

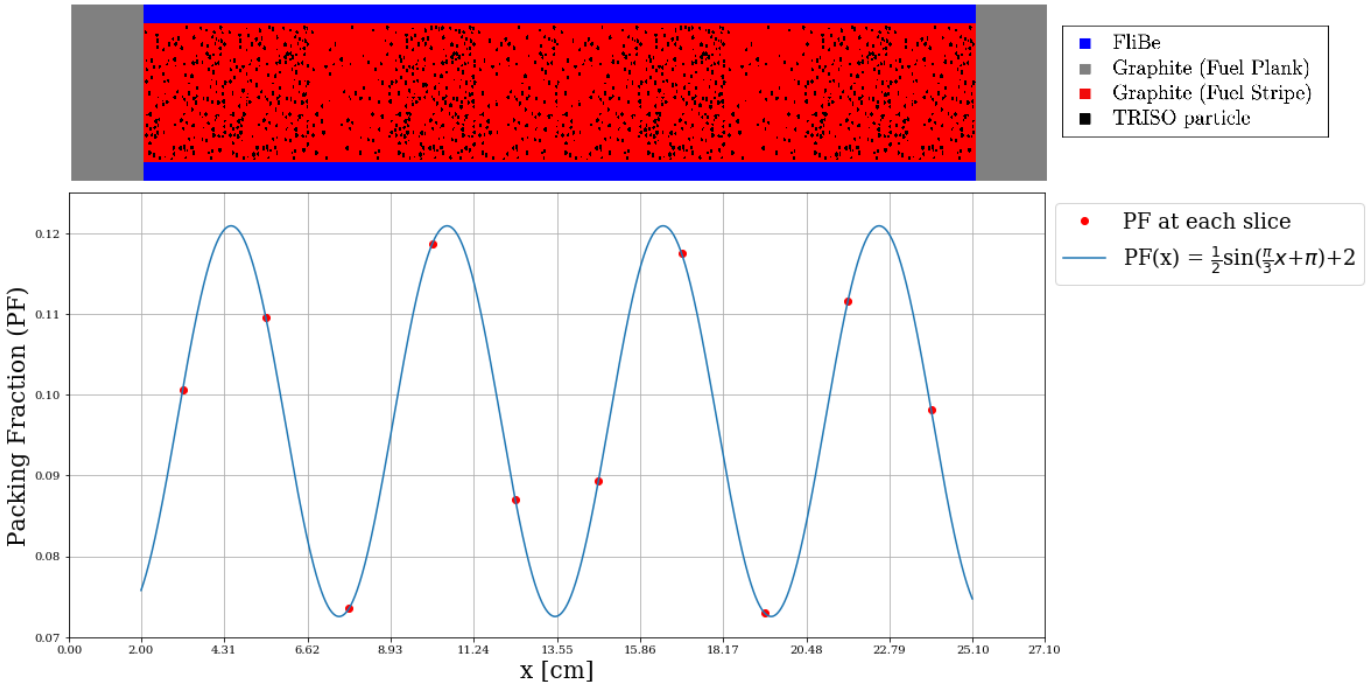


Figure 5.2: Above: Straightened Advanced High Temperature Reactor (AHTR) fuel slab with varying TRISO particle distribution across ten cells based on the sine distribution. Below: $PF(x) = (0.5\ sin(\frac{\pi}{3}x + \pi) + 2) \times NF$ sine distribution with red points indicating the packing fraction at each cell.

with 80 active cycles, 20 inactive cycles, and 8000 particles to reach ~130pcm uncertainty. Figure 5.3 shows the ROLLO input file for this genetic algorithm optimization problem. `ahtr_slab_openmc.py` is the template OpenMC straightened AHTR slab script that accepts a, b and c from ROLLO, calculates packing fraction distribution, and assigns packing fraction values to each fuel cell. Subsequently, ROLLO runs the templated OpenMC script to generate k_{eff}.

5.1.2 Hyperparameter Search

In a ROLLO input file, the user defines hyperparameters for the genetic algorithm. A good hyperparameter set guides the optimization process by balancing exploitation and exploration to find an optimal solution quickly and accurately. Finding a good hyperparameter set requires a trial-and-error process.

I performed the hyperparameter search with a coarse-to-fine random sampling scheme, whose advantages I previously discussed in Section 2.4.2. The hyperparameters varied included population size, number of generations, mutation probability, mating probability, selection operator, selection operator's number of individuals, selection operator's tournament size, mutation operator, and mating operator. I started with 25 coarse experiments and fine-tuned the hyperparameters with 15 more experiments. For each genetic algorithm experiment, the number of OpenMC evaluations remained constant at 600. The number of evaluations correlated the population size and number of generations. I randomly sampled population size and used the following equation to calculate the number of generations:

$$\text{no. of generations} = \frac{\text{no. of evaluations}}{\text{population size}} \tag{5.2}$$

Table 5.2 shows the lower and upper bounds used for randomly sampling each hyperparameter.

The initial 25 coarse experiments' sought to narrow down the hyperparameters to find

```
{
    "control_variables": {
        "a": {"min": 0.0, "max": 2.0},
        "b": {"min": 0.0, "max": 1.57},
        "c": {"min": 0.0, "max": 6.28},
    },
    "evaluators": {
        "openmc": {
            "input_script": "ahtr_slab_openmc.py",
            "inputs": ["a", "b", "c"],
            "outputs": ["keff"],
            "keep_files": false,
        }
    },
    "constraints": {"keff": {"operator": [">="], "constrained_val": [1.0]}},
    "algorithm": {
        "objective": "max",
        "optimized_variable": "keff",
        "pop_size": 60,
        "generations": 10,
        "mutation_probability": 0.23,
        "mating_probability": 0.46,
        "selection_operator": {"operator": "selTournament", "inds": 15, "tournsize": 5},
        "mutation_operator": {
            "operator": "mutPolynomialBounded",
            "eta": 0.23,
            "indpb": 0.23,
        },
        "mating_operator": {"operator": "cxBlend", "alpha": 0.46},
    },
}
```

Figure 5.3: Reactor evOLutionary aLgorithm Optimizer (ROLLO) JSON input file to maximize k_{eff} in the straightened Advanced High Temperature Reactor (AHTR) fuel slab by varying packing fraction distribution with control variables a, b, and c.

Table 5.2: Hyperparameter search is conducted in three phases: *Coarse Search*, *Fine Search 1*, *Fine Search 2*. Each hyperparameter's lower and upper bounds for each search phase are listed.

Hyperparameter	**Type**	**Coarse Search Bounds**	**Fine Search 1 Bounds**	**Fine Search 2 Bounds**
Experiments	-	0 to 24	24 to 34	35 to 39
Population size (pop)	Continuous	$10 < x < 100$	$20 < x < 60$	60
Mutation probability	Continuous	$0.1 < x < 0.4$	$0.2 < x < 0.4$	$0.2 < x < 0.3$
Mating probability	Continuous	$0.1 < x < 0.6$	$0.1 < x < 0.3$	$0.45 < x < 0.6$
Selection operator	Discrete	`SelTournament`, `SelBest`, `SelNSGA2`	`SelTournament`, `SelBest`, `SelNSGA2`	`SelTournament`
Selection individuals	Continuous	$\frac{1}{3}pop < x < \frac{2}{3}pop$	$\frac{1}{3}pop < x < \frac{2}{3}pop$	15
Selection tournament size (only for SelTournament)	Continuous	$2 < x < 8$	$2 < x < 8$	5
Mutation operator	Discrete	`mutPolynomialBounded`	`mutPolynomialBounded`	`mutPolynomialBounded`
Mating operator	Discrete	`cxOnePoint`, `cxUniform`, `cxBlend`	`cxOnePoint`, `cxUniform`, `cxBlend`	`cxOnePoint`, `cxBlend`

a smaller set of hyperparameter bounds that produce higher k_{eff} values. Figure 5.4 shows the hyperparameters' plotted against each other with a third color dimension representing the average k_{eff} value ($\overline{k_{eff}}$) in each experiment's final generation. Lighter scatter points indicate higher final population $\overline{k_{eff}}$ values, which suggests better hyperparameter sets. I plotted the hyperparameters against each other to visualize the interdependence between hyperparameters. From the coarse hyperparameter search, I noticed the following trends:

- Mutation probability has a higher $\overline{k_{eff}}$, between 0.2 and 0.4.
- Mating probability has a higher $\overline{k_{eff}}$, between 0.1 and 0.3.
- Population size has a higher $\overline{k_{eff}}$, between 20 and 60.
- No obvious interdependence between hyperparameters.

Next, I proceeded to the fine searches. From Figure 5.4, I narrowed down population size, mutation probability, and mating probability bounds, as shown in Table 5.2's *Fine Search 1 Bounds* column. I found no significant trends in the other hyperparameters, so

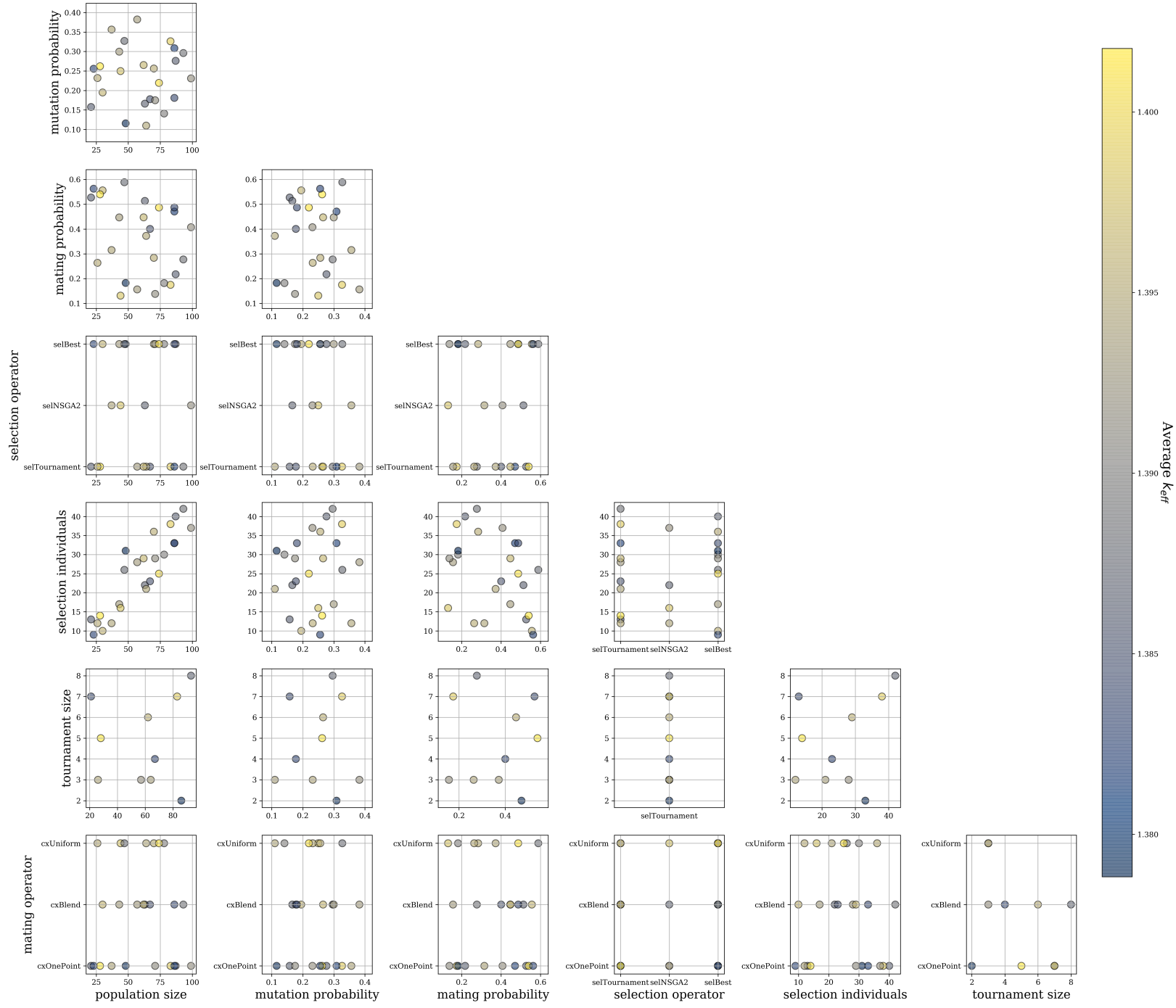


Figure 5.4: Coarse hyperparameters search's results. Hyperparameter values are plotted against each other with a third color dimension representing each experiment's final population's $\overline{k_{eff}}$.

I left them as is. I ran ten more experiments (25 to 34), sampling hyperparameters from the *Fine Search 1 Bounds*. From these results, I conducted a second fine search with five experiments (35 to 39) with further tuned hyperparameter bounds, as shown in Table 5.2's *Fine Search 2 Bounds* column. I determined these new hyperparameter bounds based on these reasons:

- Mutation probability has a higher $\overline{k_{eff}}$, between 0.2 and 0.3.
- I overlooked $\overline{k_{eff}}$ peaking at mating probability between 0.45 and 0.6 in the previous *Fine Search 1*, thus shifted the bounds.
- The highest $\overline{k_{eff}}$ occurred for `selTournament`.
- I narrowed down mating operator options to `cxBlend` and `cxOnePoint` since they had higher $\overline{k_{eff}}$.
- I selected arbitrary numbers for population size, selection individuals, and tournament size since they did not correlate with $\overline{k_{eff}}$ values.

Figure 5.5 shows the relationship between hyperparameter values and a, b, c control parameters, final generation k_{effmax}, and final generation $\overline{k_{eff}}$. The coarse experiments' scatter points are 50% transparent, while the fine experiments' scatter points are opaque. In Figure 5.5, on average, the fine experiments (opaque scatter points) have higher $\overline{k_{eff}}$, which indicates that the hyperparameter search process met its objective of finding hyperparameter bounds that enable quicker and more accurate optimization.

Table 5.3 shows the hyperparameters for the five experiments with the highest final generation $\overline{k_{eff}}$. Figure 5.6 shows the packing fraction distributions that produced the k_{effmax} from the top five experiments. Four experiments had similar packing fraction distributions peaking at approximately 0.23 in the slab's center. In contrast, one experiment had an exponential-like distribution with a peak packing fraction of 0.31 at the slab's side. The

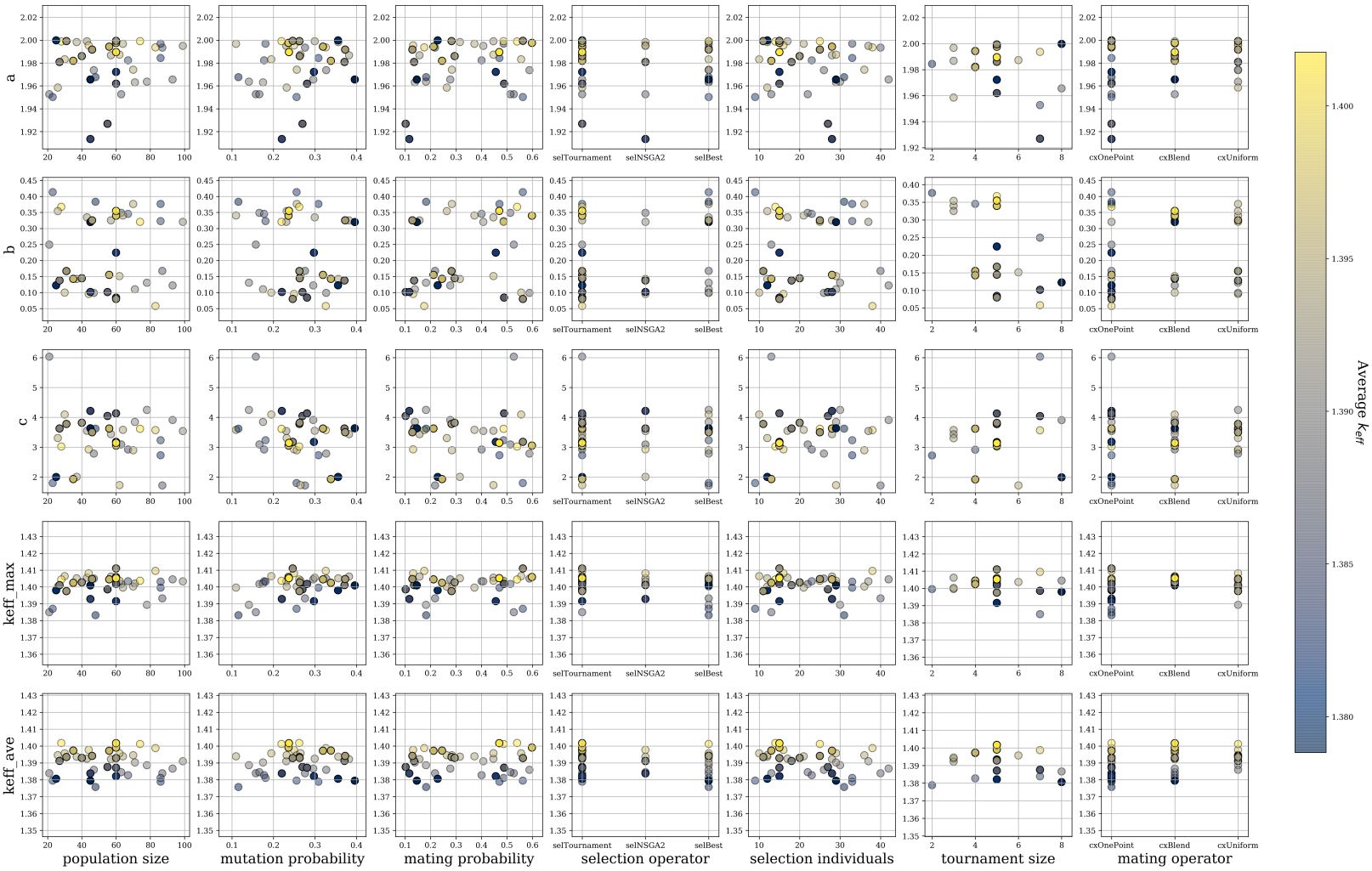


Figure 5.5: Hyperparameters search's results for all 40 experiments (coarse and fine). I plotted the hyperparameters against: a,b,c control parameters, each experiment's final generation k_{effmax}, and final generation $\overline{k_{eff}}$ with a third dimension representing each experiment's final population's $\overline{k_{eff}}$. Coarse experiments' (0 to 24) scatter points are 50% transparent, while the fine experiments' (24 to 39) scatter points are opaque.

Table 5.3: Control Parameters, k_{eff} results, and hyperparameter values for the five hyperparameter search experiments with the highest final generation $\overline{k_{eff}}$.

	Experiment No.				
Control/Output Parameters	**6**	**15**	**24**	**36**	**39**
$\overline{k_{eff}}$ [-]	1.39876	1.40155	1.40118	1.39906	1.40165
k_{effmax} [-]	1.40954	1.40440	1.40365	1.40590	1.40519
a [-]	1.993	1.998	1.999	1.997	1.989
b [$\frac{radians}{cm}$]	0.057	0.367	0.320	0.339	0.354
c [radians]	3.571	3.022	3.615	3.053	3.143
Hyperparameter					
Population size	83	28	74	60	60
Generations	8	22	9	10	10
Mutation probability	0.32	0.26	0.21	0.23	0.23
Mating probability	0.17	0.53	0.48	0.59	0.46
Selection operator	`selTournament`	`selTournament`	`selBest`	`selTournament`	`selTournament`
Selection individuals	38	14	25	15	15
Selection tournament size	7	5	-	5	5
Mutation operator	`mutPolynomial Bounded`	`mutPolynomial Bounded`	`mutPolynomial Bounded`	`mutPolynomial Bounded`	`mutPolynomial Bounded`
Mating operator	`cxOnePoint`	`cxOnePoint`	`cxUniform`	`cxBlend`	`cxBlend`

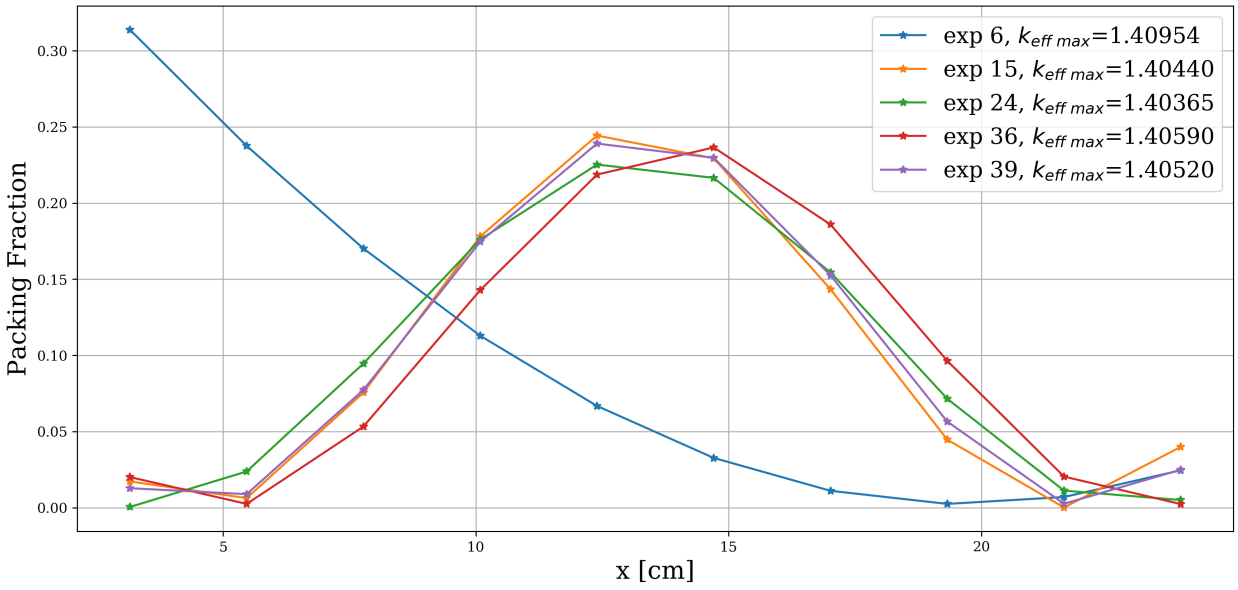


Figure 5.6: Packing fraction distribution across the x-axis of the Advanced High Temperature Reactor (AHTR) slab for the five hyperparameter search experiments with the highest final generation $\overline{k_{eff}}$.

similar final packing fraction distributions demonstrate genetic algorithms' robustness to find the optimal global solutions with different hyperparameters.

I ran these simulations on the BlueWaters supercomputer [55]. In each ROLLO simulation, each generation runs a population size number of individual OpenMC simulations. Each OpenMC simulation takes approximately 13 minutes to run on a single BlueWaters XE node. With approximately 600 OpenMC evaluations per ROLLO simulation, the ROLLO simulation takes about 130 BlueWaters node-hours. The hyperparameter search ran 40 ROLLO simulations, thus using approximately 5200 node-hours.

5.1.3 Results for Best Hyperparameter Set

I define the best-performing hyperparameter set as the experiment that produces the highest $\overline{k_{eff}}$ in its final generation. *Fine Search 2*'s experiment 39 produces the best performing hyperparameter set, shown in Table 5.3, with center-peaking packing fraction distribution of $max(k_{eff})$ = 1.40519. Experiment 39's k_{effmax} exceeds the original straightened AHTR

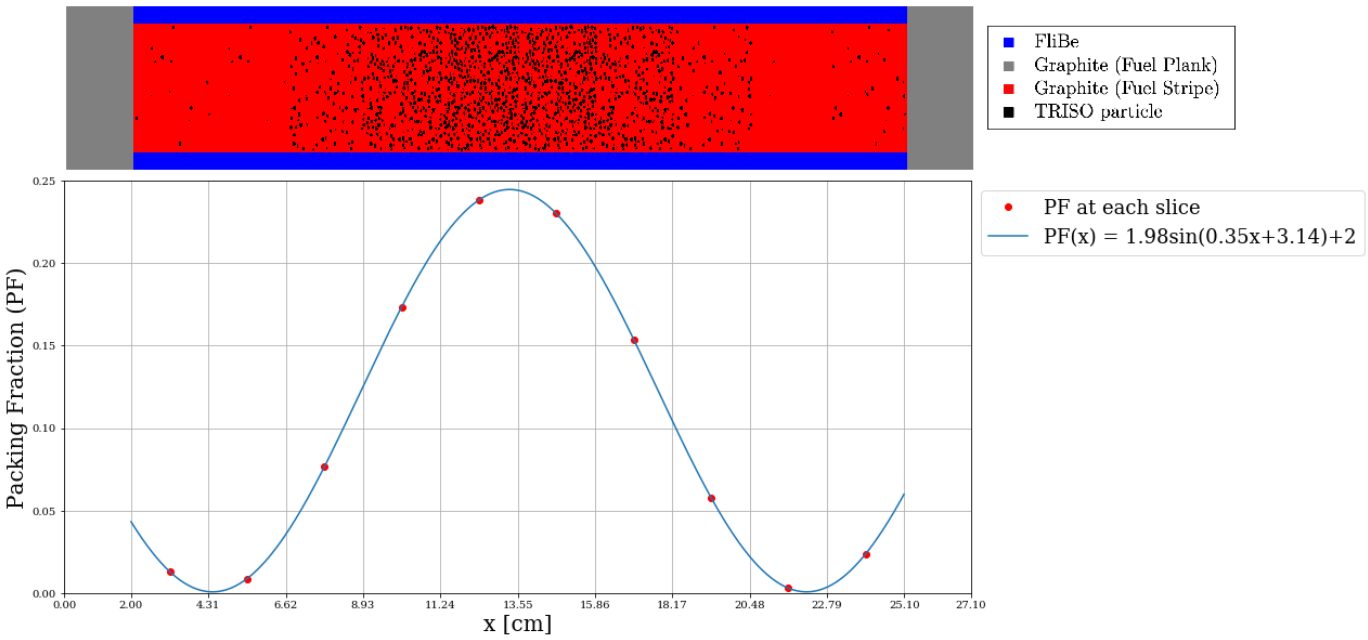
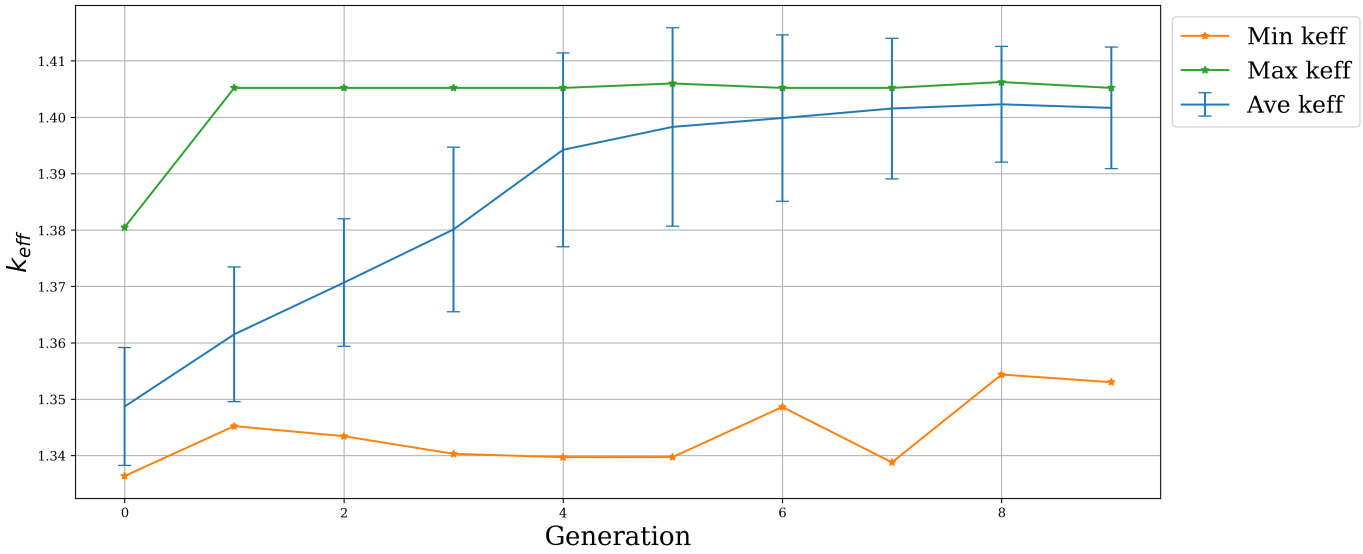


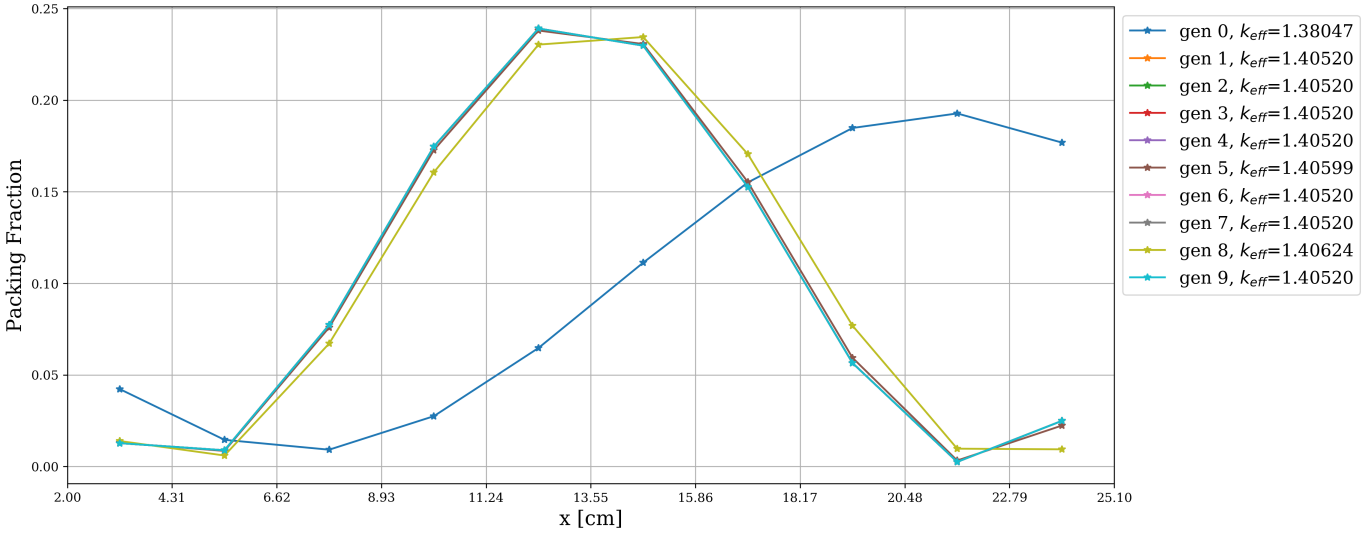
Figure 5.7: Experiment 39 packing distribution that produced k_{effmax} = 1.40519. Below: $PF(x)$ = $(1.98\ sin(0.35x + 3.14) + 2) \times NF$ sine distribution with red points indicating the packing fraction at each cell. Above: Straightened Advanced High Temperature Reactor (AHTR) fuel slab with varying TRISO particle distribution across ten cells based on the sine distribution.

configuration's k_{eff} by ~ 2000pcm, proving that optimizing inhomogenous fuel distributions enables better neutronics. Figure 5.7 shows the packing fraction distribution that produced k_{effmax} = 1.40519.

Figures 5.8a and 5.8b show the k_{eff} evolution and packing fraction distribution through the best performing 39^{th} experiment's generations. The k_{effmax} converged quickly by generation 1; however, this usually does not occur. The genetic algorithm optimizes stochastically, resulting in the possibility that the algorithm randomly samples a control parameter set that maximizes the objective function early in the optimization process. The $\overline{k_{eff}}$ demonstrates how each generation's average k_{eff} converges towards a higher value with each generation's improvements. To demonstrate how the genetic algorithm optimization process usually goes, Figures 5.9a and 5.9b show the k_{eff} evolution and packing fraction distribution through the second-best performing 15^{th} experiment's generations. Experiment 15 demonstrates how



(a) Minimum, average, and maximum k_{eff} value evolution.



(b) k_{effmax} packing fraction distribution evolution.

Figure 5.8: Each generation's results for ROLLO's genetic algorithm optimization of the Straightened Advanced High Temperature Reactor (AHTR) Fuel Slab. The ROLLO simulation used the 39^{th} experiment's hyperparameter set.

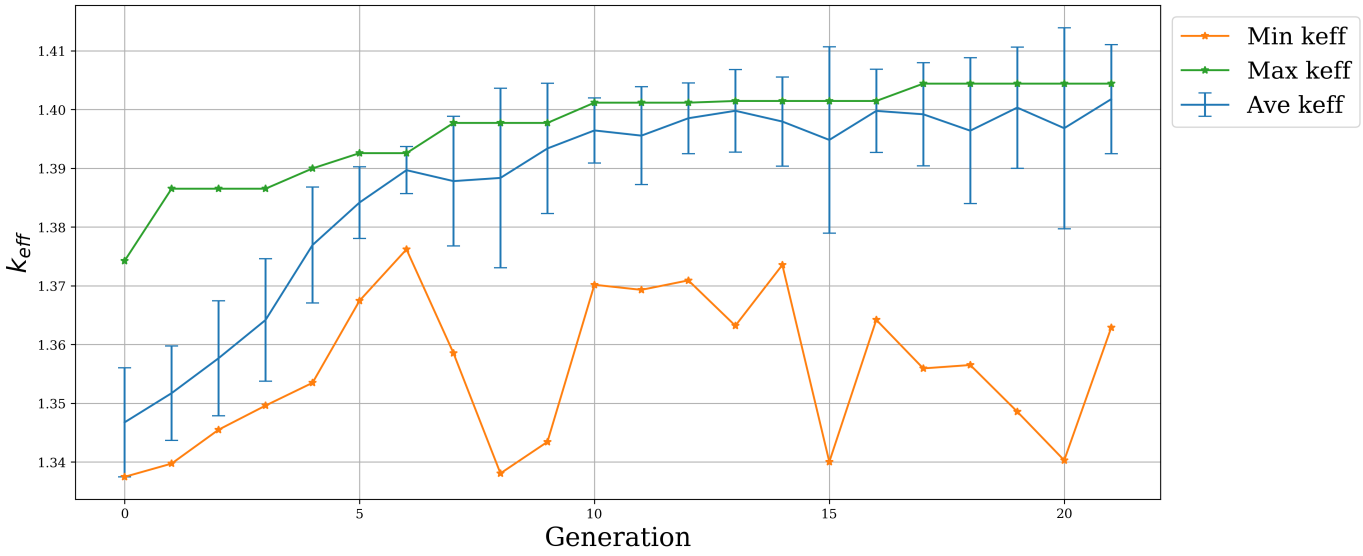
both maximum and average k_{eff} converge towards a higher k_{eff} with improvements from each generation.

In both Experiments 39 and 15, packing fractions peaked at approximately 0.23 in the slab center and decreased to zero at the sides. The amplitude, a, for the packing fraction distribution that produced k_{effmax} for Experiment 39 and the other top-five experiments (Table 5.3) have settled at the upper bound of approximately 2. A large sine distribution amplitude, a, demonstrates that a slab geometry with larger packing fraction variations results in a higher k_{eff}. These observations about packing fraction distribution for k_{effmax} are consistent with conclusions from the FHR benchmark (Chapter 3): a high k_{eff} occurs with a good balance between fuel loading and moderation space. Fission occurs at high TRISO particle concentration areas at thermal flux; however, the neutrons are born at fast flux and require moderation to slow down to thermal ranges. Therefore, larger moderation areas ensure higher resonance escape probability for the fast neutrons resulting in higher thermal flux, leading to more fission occurring and a higher k_{eff}.

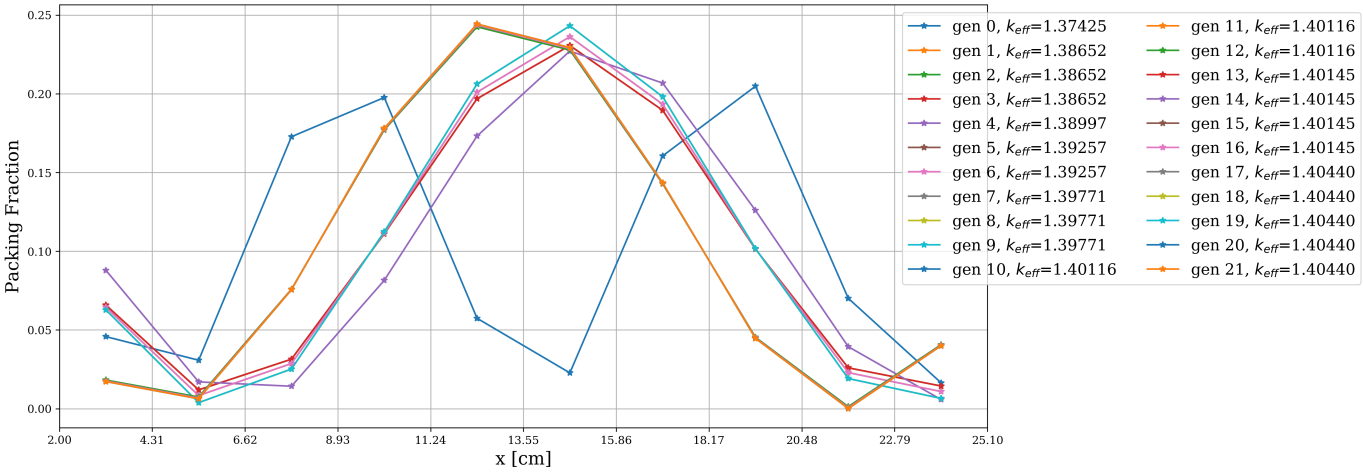
I also observed that TRISO particle packing fraction peaks in the center of the slab, proving that if the optimization problem focuses purely on the slab's neutronics by maximizing k_{eff}, the fuel tends to culminate in the middle. Center-peaking fuel density is nonideal for other key reactor core qualities, such as maximal heat transfer and minimal power peaking factor (PPF).

5.2 AHTR Multiphysics Model Preliminary Work

In the proposed PhD scope, I will use the open-source simulation tool, Moltres, to conduct AHTR multiphysics simulations. Moltres, an application built atop the MOOSE parallel finite element framework [27], contains physics kernels and boundary conditions to solve arbitrary-group deterministic neutron diffusion and thermal-hydraulics Partial Differential Equations (PDEs) simultaneously on a single mesh [50, 59]. AHTR Moltres simulations will



(a) Minimum, average, and maximum k_{eff} values evolution.



(b) k_{effmax}'s packing fraction distribution evolution.

Figure 5.9: Results for each generation for ROLLO's genetic algorithm optimization of the Straightened Advanced High Temperature Reactor (AHTR) Fuel Slab. The ROLLO simulation used the 15^{th} experiment's hyperparameter set.

capture thermal feedback effects, absent from the purely neutronics OpenMC simulations. The objective of setting up the Moltres AHTR simulation is to eventually couple Moltres with ROLLO for AHTR multiphysics optimization.

The benefits of Moltres over other multiphysics software, RELAP5 and NESTLE (used previously for AHTR modeling and described in Section 2.1.2), for coupled neutronics and thermal-hydraulics simulation:

- Moltres supports up to 3-D meshes, solving neutron diffusion and thermal-hydraulics PDEs simultaneously on the same mesh [59]. This is much more flexible than NESTLE and RELAP5, which only support rectangular and hexagonal assembly lattices. Therefore, Moltres can explore arbitrary reactor geometries easily.

- Moltres tightly couples neutronics and thermal-hydraulics, thus providing higher accuracy in some tightly coupled problems.

- Moltres, a MOOSE-based application, uses MPI for parallel computing, and compiles and runs on HPCs.

To run Moltres simulations, the user provides group constant data from a neutron transport solver, such as OpenMC, for the Moltres multigroup neutron diffusion calculations and a mesh file representing the reactor geometry. A TRISO-level fidelity mesh file is impractical and will result in an extremely long Moltres runtime. For successful AHTR Moltres simulation, I must establish suitable spatial and energy homogenization that preserves accuracy while maintaining an acceptable runtime.

5.2.1 Straightened AHTR Fuel Slab Multigroup Simulation

I use the continuous energy OpenMC simulation to generate multigroup cross section data defined over discretized energy groups and spatial segments. I then use OpenMC's multigroup calculation mode with the previously generated multigroup cross section data to calculate

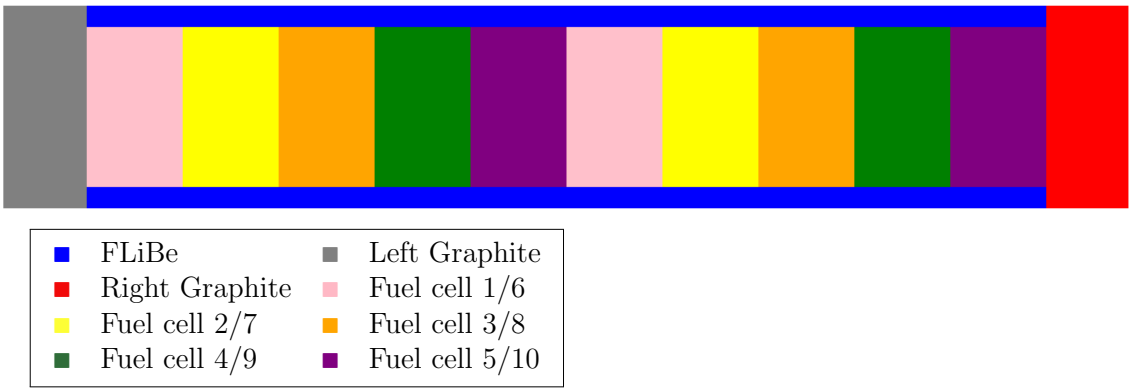


Figure 5.10: Straightened Advanced High Temperature Reactor (AHTR) fuel slab spatially discretized into 13 *cells* for OpenMC multigroup calculation.

Table 5.4: 4-group energy structures for Advanced High Temperature Reactor (AHTR) geometry derived by [28].

Group Boundaries [MeV]		
Group #	**Upper Bound**	**Lower Bound**
1	2.0000×10^{1}	9.1188×10^{-3}
2	9.1188×10^{-3}	2.9023×10^{-5}
3	2.9023×10^{-5}	1.8554×10^{-6}
4	1.8554×10^{-6}	1.0000×10^{-12}

k_{eff}. Comparison of k_{eff} for the continuous and multigroup simulations determines if the energy and spatial homogenization used are acceptable.

In this section, the straightened AHTR fuel slab simulations use the TRISO particle distribution that generated k_{effmax} from the best hyperparameter set (Section 5.1.3). For spatial homogenization of the straightened AHTR fuel slab, I used OpenMC's *cell* domain type to compute multigroup cross sections for different *cells*. I discretized the slab into 13 *cells*: FLiBe, left graphite, right graphite, and ten fuel cells (each cell has a different packing fraction). Figure 5.10 illustrates the AHTR spatial homogenization for the OpenMC multigroup calculation. I used the four group energy structure derived by Gentry et al. [28] for AHTR geometries. Table 5.4 defines the group boundaries.

Table 5.5 shows the k_{eff} values from the continuous energy simulation and the spatial and energy homogenized simulation. The 26pcm difference between k_{eff} values is within both

Table 5.5: Straightened Advanced High Temperature Reactor (AHTR) fuel slab's k_{eff} for case with continuous energy and space and case with spatial and energy homogenization. Both simulations were run on one BlueWaters XE Node, with 80 active cycles, 20 inactive cycles, and 8000 particles.

Homogenization	k_{eff}	**Simulation time [s]**
None	1.40473 ± 0.00115	808
Spatial and Energy	1.40499 ± 0.00109	50

uncertainty values, assuring that the spatial and energy homogenization used is suitable for generating group constants for Moltres.

5.3 Summary

This chapter demonstrated the preliminary work completed for AHTR optimization. I conducted a multigroup AHTR slab simulation with four-group energy and spatial homogenization, which resulted in k_{eff} within the uncertainty of the continuous energy simulation. The minimal k_{eff} difference assures that I can use these homogenizations when generating group constants for Moltres. I also successfully applied ROLLO to maximize k_{eff} in a straightened Advanced High Temperature Reactor (AHTR) fuel slab by varying the TRISO particle packing fraction distribution. The optimization process began with a coarse-to-fine random sampling hyperparameter search to find the genetic algorithm hyperparameters that worked best. Experiment 39 performed the best with a hyperparameter set that produced the highest final generation $\overline{k_{eff}}$ of 1.40165. The TRISO particle packing fraction distribution that produced the final generation's maximum k_{eff} of 1.40519 peaks at the slab's center with packing fraction distribution: $PF(x) = 1.989\ sin(0.54x + 3.143)$. This demonstration problem had a single objective function of maximizing k_{eff}. However, many other objectives should be considered, such as maximizing heat transfer and minimizing power peaking factor. Thus, in the next chapter, I propose future simulations for optimizing these objective functions simultaneously.

Chapter 6

Future Work and Proposed Simulations

Chapter 2 demonstrated the need for this work with a summary of how additive manufacturing of nuclear reactor core components frees complex reactor geometries from traditional manufacturing constraints and enables reactor designers to reexamine reactor core design optimization. The literature review (Chapter 2) also concluded that stochastic evolutionary algorithm optimization methods could find global optimums for reactor design problems in the vast exploration design space enabled by additive manufacturing. Chapter 3 introduced the Fluoride-Salt-Cooled High-Temperature Reactor (FHR) benchmark with the AHTR design and highlighted its benefits, such as passive safety behavior with negative temperature coefficients. Chapter 4 introduced the Reactor evOLutionary aLgorithm Optimizer (ROLLO) software package, which applies evolutionary algorithm optimization techniques to nuclear reactor design. In Chapter 5, I successfully applied ROLLO to optimize the TRISO packing fraction distribution in an AHTR slab and demonstrated the neutron transport energy and spatial homogenizations for generating group constants for Moltres.

Based on the preliminary work conducted, this chapter proposes future simulations categorized into two groups: AHTR development and ROLLO optimization. The proposed work aims to address AHTR modeling challenges further and demonstrate using ROLLO for multi-objective AHTR optimization of arbitrary geometries and fuel distribution. For AHTR development, I propose the following simulations:

- AHTR 3D full core neutronics OpenMC simulation
- AHTR fuel slab and one-third fuel assembly multiphysics Moltres simulation

For ROLLO optimization, I propose the following ROLLO simulations:

- AHTR slab geometry optimization to maximize k_{eff}, minimize power peaking factor (PPF), and maximize heat transfer ($\dot{Q}$) by varying TRISO x-axis distribution and FLiBe channel shape using OpenMC
- AHTR one-third fuel assembly optimization to maximize k_{eff}, minimize power peaking factor, and maximize heat transfer by varying TRISO x-y axis distribution and FLiBe channel shape using OpenMC

The proposed AHTR simulations contributes to the OECD-NEA FHR benchmark, which aims to understand the technical challenges and validate the available neutron transport and thermal-hydraulics software for the AHTR design. Previous efforts toward nuclear reactor optimization, focused on optimizing classical reactor parameters such as radius of fuel pellet and clad, enrichment of fuel, pin pitch, etc. The promise of cheaper and faster manufacturing of reactor components with additive manufacturing frees complex reactor geometries from previous manufacturing constraints and allows reactor designers to reexamine reactor design optimization. Thus, the proposed multi-objective ROLLO simulations will be the first to explore optimizing arbitrary reactor geometries and fuel distributions. The results from these optimization problems will inform on how heterogenous fuel distributions and arbitrary coolant channel shapes perform compared to classical distributions and shapes for the AHTR design. These optimization problems will also demonstrate ROLLO's capabilities for multi-objective reactor design optimization, and ROLLO's open-source availability will enable other reactor designers to utilize the tool as well.

6.1 AHTR Model Development

The FHR benchmark introduced in Chapter 3 is an ongoing NEA project to assess the modeling and simulation capabilities for the AHTR. Benchmark participants, including the

UIUC team, contributed Phases I-A and I-B (2D assembly steady-state and depletion) so far. The upcoming phases consist of 3D neutronics models and multiphysics models. Thus, to support the FHR benchmark, the proposed work will complete the benchmark's Phase I-C. In preparation for the later multiphysics benchmark phases, the proposed work will utilize Moltres to model AHTR multiphysics.

6.1.1 FHR Benchmark Phase I-C

The FHR benchmark's Phase I-C extends the 2D assembly model from Phases I-A and I-B into a 3D assembly model. The benchmark organizers will release the Phase I-C detailed specifications and required results in June 2021.

6.1.2 AHTR Multiphysics Model

Setting up a Moltres multiphysics simulation requires the user to provide group constant data from a neutron transport solver, such as OpenMC. Moltres neutronics calculations use the following group constants: [50, 59]:

Σ_g^f: macroscopic fission cross section in group g

Σ_g^r: macroscopic removal cross section in group g

$\Sigma_{g'\to g}^s$: macroscopic scattering cross section from group g' to g

D_g: diffusion coefficient of neutrons in group g

ϵ_g: average fission energy per fission by a neutron from group g

ν: average neutron yield per fission by a neutron from group g

$\frac{1}{v}$: inverse neutron speed in group g

λ_i: decay constant of delayed neutron precursor (DNP) group i

β_{eff}: effective delayed neutron fraction.

A Python script from the Moltres Github repository [49] extracts group constants from the neutron transport solver's output files. The Python script currently enables group constant extraction from Serpent [46] and SCALE [10] output files. I used OpenMC to model the AHTR neutronics for the FHR benchmark; thus, I will add the capability to extract group constants from OpenMC output files to the Moltres Python group constants extraction script.

Section 5.2 demonstrated that the multigroup neutronics simulation with four-group energy and spatial homogenization of the AHTR fuel slab generated a k_{eff} within uncertainty of the continuous energy and space neutronics simulation. I will utilize these homogenizations to create group constants for the Moltres AHTR fuel slab simulation. I will then set up a mesh for the AHTR fuel slab, run a Moltres simulation, and verify Moltres' ability to reproduce the following key neutronics parameters:

- k_{eff} (effective multiplication factor)
- reactivity coefficients: β_{eff}, α_D (doppler coefficient), $\alpha_{T,FliBe}$ (FLiBe temperature coefficient), and α_M (moderator temperature coefficient)
- Neutron energy spectrum
- $\phi_1(\vec{x},\vec{y})$, $\phi_2(\vec{x},\vec{y})$, and $\phi_3(\vec{x},\vec{y})$ (neutron flux distribution in four coarse energy groups).

Once verified, I will run a steady-state Moltres multiphysics simulation to determine the maximum temperature in the fuel slab at steady-state.

With information gleaned from the Moltres AHTR fuel slab simulation, I will test out energy and spatial homogenization for generating group constants for a one-third AHTR fuel assembly model. Then, I will proceed to set up the one-third AHTR fuel assembly model simulation, verify its key neutronics parameters, and finally, run a steady-state Moltres simulation.

6.2 ROLLO Optimization

Section 5.1 concluded that the AHTR slab optimization problem should be further developed by considering other objectives such as maximizing heat transfer and minimizing power peaking factor. In the proposed work, I will explore each objective separately and then together. Table 6.1 describes each objective and how I will quantify each objective. I will

Table 6.1: Reactor evOLutionary aLgorithm Optimizer (ROLLO) optimization problem objectives with their quantification descriptions.

Objective	**Quantification**
Best neutronics	Maximize k_{eff}
Maximize heat transfer	Maximize ϕ_{total} in areas along FLiBe coolant
Minimize power peaking factor	Minimize $P_{high} - P_{low}$

vary the following slab parameters to meet the described problem objectives:

- TRISO particle packing fraction distribution $\rho_{TRISO}(\vec{r})$
- FLiBe coolant channel shape

I will conduct these optimizations for the straightened AHTR fuel slab geometry (as seen in Figure 5.1) and for one diamond-shaped sector (as seen in Figure 3.2) with x-y axis periodic and z axis reflective boundary conditions. Table 6.2 outlines the proposed simulations' details. I will use the optimal hyperparameters derived in Section 5.1.2 for the proposed simulations. Ideally, a new hyperparameter search should be conducted for each simulation to find the best hyperparameter set for each unique problem; however, the computational expense for conducting 11 hyperparameter searches is impractical. Due to the problem similarity, the hyperparameters can remain unchanged. Table 6.3 summarizes the optimal hyperparameters.

I will extend the ROLLO simulations proposed in Table 6.2 to include Moltres evaluations if the proposed AHTR multiphysics Moltres simulations (Section 6.1.2) find approximations and assumptions that maintain accuracy while keeping acceptable Moltres runtimes.

Table 6.2: Proposed Reactor evOLutionary aLgorithm Optimizer (ROLLO) simulations for optimizing Advanced High Temperature Reactor (AHTR) fuel assembly. Simulations explore two geometries: straightened AHTR fuel slab and AHTR's diamond-shaped section, containing six fuel slabs. $\dot{Q}$: Heat transfer, PPF: Power Peaking Factor, $\rho_{TRISO}(\vec{r})$: TRISO particle distribution

Simulation	**AHTR Geometry**	**Objectives**	**Varying Parameters**
1	Single fuel slab	• $\max(k_{eff})$	• $\rho_{TRISO}(\vec{r})$
2	Single fuel slab	• $\max(\dot{Q})$	• $\rho_{TRISO}(\vec{r})$
3	Single fuel slab	• $\min(PPF)$	• $\rho_{TRISO}(\vec{r})$
4	Single fuel slab	• $\max(k_{eff})$	• FLiBe channel shape
5	Single fuel slab	• $\max(\dot{Q})$	• FLiBe channel shape
6	Single fuel slab	• $\min(PPF)$	• FLiBe channel shape
7	Single fuel slab	• $\max(k_{eff})$ • $\max(\dot{Q})$ • $\min(PPF)$	• $\rho_{TRISO}(\vec{r})$
8	Single fuel slab	• $\max(k_{eff})$ • $\max(\dot{Q})$ • $\min(PPF)$	• FLiBe channel shape
9	Single fuel slab	• $\max(k_{eff})$ • $\max(\dot{Q})$ • $\min(PPF)$	• $\rho_{TRISO}(\vec{r})$ • FLiBe channel shape
10	Diamond section with six fuel slabs	• $\max(k_{eff})$ • $\max(\dot{Q})$ • $\min(PPF)$	• $\rho_{TRISO}(\vec{r})$
11	Diamond section with six fuel slabs	• $\max(k_{eff})$ • $\max(\dot{Q})$ • $\min(PPF)$	• FLiBe channel shape
12	Diamond section with six fuel slabs	• $\max(k_{eff})$ • $\max(\dot{Q})$ • $\min(PPF)$	• $\rho_{TRISO}(\vec{r})$ • FLiBe channel shape

Table 6.3: Hyperparameter values for the best hyperparameter set calculated in Section 5.1.2.

Hyperparameters	**Values**
Population size	60
Generations	10
Mutation probability	0.23
Mating probability	0.46
Selection operator	`selTournament`
Selection individuals	15
Selection tournament size	5
Mutation operator	`mutPolynomialBounded`
Mating operator	cxBlend

6.3 Conclusion

Breakthroughs in metal component additive manufacturing fabrication have expedited the development of methods for nuclear reactor component additive manufacturing. The promise of cheaper and faster manufacturing of reactor components with additive manufacturing frees complex reactor geometries from previous manufacturing constraints and allows reactor designers to reexamine reactor design optimization. Therefore, I propose to explore the vast design space enabled by additive manufacturing with the evolutionary algorithm optimization technique to find global optima in multi-objective design problems, such as nuclear reactor optimization.

In the preliminary work, I designed the ROLLO Python package that applies evolutionary algorithm optimization techniques to nuclear reactor design using the DEAP Python module, OpenMC, and Moltres. ROLLO seeks to enable reactor designers to utilize robust evolutionary algorithm optimization methods without going through the cumbersome process of setting up a genetic algorithm framework. The many advantageous features of AHTRs led me to choose to apply the evolutionary algorithm optimization methods to this reactor type. I participated in Phase I-A and I-B of the Organisation for Economic Co-operation and Development (OECD) NEA's FHR benchmarking exercise. I also applied ROLLO to a single objective function problem: maximize k_{eff} in the AHTR fuel slab by varying the

TRISO particle packing fraction distribution. This problem demonstrated the effectiveness and robustness of genetic algorithms at optimizing reactor parameters for an objective function. However, many other objectives should also be considered, such as maximizing heat transfer and minimizing power peaking factor.

Therefore, I propose to further explore using ROLLO for multi-objective AHTR optimization of arbitrary geometries and fuel distribution. Optimization objectives include maximizing k_{eff}, maximizing heat transfer, and minimizing power peaking factor. I also propose to further address AHTR modeling challenges by completing the FHR benchmark Phase I-C and to set up Moltres simulations to model AHTR multiphysics.

References

[1] Boeing: 3D printing done right. URL: https://www.boeing.com/features/innovation-quarterly/nov2017/feature-thought-leadership-3d-printing.page.

[2] Fluoride Salt-Cooled High-Temperature Reactor (FHR) Benchmark. URL: https://www.oecd-nea.org/jcms/pl_20249/fluoride-salt-cooled-high-temperature-reactor-fhr-benchmark.

[3] Roadmap for Regulatory Acceptance of Advanced Manufacturing Methods in the Nuclear Energy Industry. NEI Report, Nuclear Energy Institute. URL: https://www.nrc.gov/docs/ML1913/ML19134A087.pdf.

[4] Printed titanium parts expected to save millions in Boeing Dreamliner costs. *Reuters*, April 2017. URL: https://www.reuters.com/article/us-norsk-boeing-idUSKBN17C264.

[5] *Climate Change and Nuclear Power 2018*. Non-serial Publications. INTERNATIONAL ATOMIC ENERGY AGENCY, Vienna, 2018. URL: https://www.iaea.org/publications/13395/climate-change-and-nuclear-power-2018.

[6] Transformation In 3D: How A Walnut-Sized Part Changed The Way GE Aviation Builds Jet Engines | GE News, 2018. URL: http://www.ge.com/news/reports/transformation-3d-walnut-sized-part-changed-way-ge-aviation-builds-jet-engines.

[7] C. Behar. Technology roadmap update for generation IV nuclear energy systems. In *OECD Nuclear Energy Agency for the Generation IV International Forum, accessed Jan*, volume 17, pages 2014–03, 2014. Issue: 2018.

[8] Andrew Bergeron and J. B. Crigger. Early progress on additive manufacturing of nuclear fuel materials. *Journal of Nuclear Materials*, 508:344–347, 2018. ISBN: 0022-3115 Publisher: Elsevier.

[9] Benjamin R. Betzler, David Chandler, David H. Cook, Eva E. Davidson, and Germina Ilas. Design optimization methods for high-performance research reactor core design. *Nuclear Engineering and Design*, 352:110167, 2019. ISBN: 0029-5493 Publisher: Elsevier.

[10] J. A. Bucholz. SCALE: A modular code system for performing standardized computer analyses for licensing evaluation. NUREG NUREG/CR-0200-Vol.2-Bk.2; ORNL/NUREG/CSD-2-Vol.2-Bk.2 ON: DE82013370; TRN: 82-013156, Oak Ridge National Lab., TN (USA), 1982. URL: http://www.osti.gov/scitech/biblio/5360496.

[11] Jonathan Byrne, Philip Cardiff, Anthony Brabazon, and Michael ONeill. Evolving parametric aircraft models for design exploration and optimisation. *Neurocomputing*, 142:39–47, October 2014. URL: https://linkinghub.elsevier.com/retrieve/pii/S092523121400530X, doi:10.1016/j.neucom.2014.04.004.

[12] M. B. Chadwick, M. Herman, P. Obloinsk, M. E. Dunn, Y. Danon, A. C. Kahler, D. L. Smith, B. Pritychenko, G. Arbanas, R. Arcilla, R. Brewer, D. A. Brown, R. Capote, A. D. Carlson, Y. S. Cho, H. Derrien, K. Guber, G. M. Hale, S. Hoblit, S. Holloway, T. D. Johnson, T. Kawano, B. C. Kiedrowski, H. Kim, S. Kunieda, N. M. Larson, L. Leal, J. P. Lestone, R. C. Little, E. A. McCutchan, R. E. MacFarlane, M. MacInnes, C. M. Mattoon, R. D. McKnight, S. F. Mughabghab, G. P. A. Nobre, G. Palmiotti, A. Palumbo, M. T. Pigni, V. G. Pronyaev, R. O. Sayer, A. A. Sonzogni, N. C. Summers, P. Talou, I. J. Thompson, A. Trkov, R. L. Vogt, S. C. van der Marck, A. Wallner, M. C. White, D. Wiarda, and P. G. Young. 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, December 2011. doi:10.1016/j.nds.2011.11.002.

[13] Gwendolyn Chee. arfc/fhr-benchmark, 2021.

[14] Gwendolyn J. Y. Chee. rollo: Reactor Evolutionary Algorithm Optimizer. URL: https://github.com/arfc/rollo.

[15] Anselmo T. Cisneros and Dan Ilas. Neutronics and Depletion Methods for Parametric Studies of Fluoride Salt Cooled High Temperature Reactors with Slab Fuel Geometry and Multi-Batch Fuel Management Schemes. 2012.

[16] Lisandro Dalcn, Rodrigo Paz, Mario Storti, and Jorge DEla. MPI for Python: Performance improvements and MPI-2 extensions. *Journal of Parallel and Distributed Computing*, 68(5):655–662, 2008. ISBN: 0743-7315 Publisher: Elsevier.

[17] David L. Poole. *Computational intelligence*. Oxford University Press, 1998. URL: http://archive.org/details/computationalint00pool.

[18] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[19] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002. ISBN: 1089-778X Publisher: IEEE.

[20] M. I. T. Facilitators, UW Madison Facilitators, and UC Berkeley Facilitators. Fluoride-Salt-Cooled, High-Temperature Reactor (FHR) Development Roadmap and Test Reactor Performance Requirements White Paper. 2013. URL: http://fhr.nuc.berkeley.edu/wp-content/uploads/2013/08/12-004-FHR-Workshop-4-Report-Final.pdf.

[21] C. D. Fletcher and R. R. Schultz. RELAP5/MOD3 code manual. Technical report, Nuclear Regulatory Commission, 1992.

[22] C. W. Forsberg, K. A. Terrani, L. L. Snead, and Y. Katoh. Fluoride-Salt-Cooled High-Temperature Reactor (FHR) with Silicon-Carbide-Matrix Coated-Particle Fuel. In *Transactions of the American Nuclear Society*, volume 107, page 907, 2012. URL: http://info.ornl.gov/sites/publications/files/Pub37875.pdf.

[23] Charles W. Forsberg, Per F. Peterson, and Paul S. Pickard. Molten-salt-cooled advanced high-temperature reactor for production of hydrogen and electricity. *Nuclear Technology*, 144(3):289–302, 2003. ISBN: 0029-5450 Publisher: Taylor & Francis.

[24] Flix-Antoine Fortin, Franois-Michel De Rainville, Marc-Andr Gardner, Marc Parizeau, and Christian Gagn. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13(Jul):2171–2175, 2012.

[25] Christian Gagn and Marc Parizeau. Open BEAGLE: A New Versatile C++ Framework for Evolutionary Computation. In *GECCO Late Breaking Papers*, pages 161–168. Citeseer, 2002.

[26] Aaron Garrett. inspyred: Bio-inspired Algorithms in Python. *URL: https://pypi.python. org/pypi/inspyred (visited on 11/28/2016)*, 2014.

[27] Derek Gaston, Chris Newman, Glen Hansen, and Damien Lebrun-Grandie. MOOSE: A parallel computational framework for coupled systems of nonlinear equations. *Nuclear Engineering and Design*, 239(10):1768–1778, October 2009. URL: http://www.sciencedirect.com/science/article/pii/S0029549309002635, doi:10.1016/j.nucengdes.2009.05.021.

[28] Cole Andrew Gentry. *Development of a Reactor Physics Analysis Procedure for the Plank-Based and Liquid Salt-Cooled Advanced High Temperature Reactor*. PhD thesis, 2016.

[29] Ian Gibson, David W. Rosen, and Brent Stucker. *Additive manufacturing technologies*, volume 17. Springer, 2014.

[30] GIF. A technology roadmap for generation IV nuclear energy systems. Technical Report GIF-002-00, US DOE Nuclear Energy Research Advisory Committee and the Generation IV International Forum, 2002.

[31] David E. Goldberg. Genetic algorithms in search. *Optimization, and MachineLearning*, 1989. Publisher: Addison Wesley Publishing Co. Inc.

[32] David E. Goldberg, Kalyanmoy Deb, and Dirk Thierens. Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers*, 32(1):10–16, 1993. ISBN: 0453-4662 Publisher: The Society of Instrument and Control Engineers.

[33] Sherrell R. Greene, Jess C. Gehin, David Eugene Holcomb, Juan J. Carbajo, Dan Ilas, Anselmo T. Cisneros, Venugopal Koikal Varma, William R. Corwin, Dane F. Wilson, and Graydon L. Yoder Jr. Pre-conceptual design of a fluoride-salt-cooled small modular advanced high-temperature reactor (SmAHTR). *Oak Ridge National Laboratory, ORNL/TM-2010/199*, 2010.

[34] M. K. M. Ho, G. H. Yeoh, and G. Braoudakis. Molten salt reactors. In A. Mndez-Vilas, editor, *Materials and processes for energy: communicating current research and technological developments*, number 1 in Energy Book Series, pages 761–768. Formatex Research Center, Badajoz, Spain, 2013 edition, 2013. http://www.formatex.info/energymaterialsbook/ http://www.energymaterialsbook.org/chapters.html.

[35] David E. Holcomb, George F. Flanagan, Gary T. Mays, W. David Pointer, Kevin R. Robb, and Graydon L. Yoder Jr. Fluoride salt-cooled high-temperature reactor technology development and demonstration roadmap. *ORNL/TM-2013/401, ORNL, Oak Ridge, TN*, 2013.

[36] David Eugene Holcomb, Dan Ilas, Venugopal Koikal Varma, Anselmo T. Cisneros, Ryan P. Kelly, and Jess C. Gehin. Core and refueling design studies for the advanced high temperature reactor. *ORNL/TM-2011/365, Oak Ridge National Laboratory, Oak Ridge, TN*, 2011.

[37] Jeremy Jordan. Hyperparameter tuning for machine learning models., November 2017. URL: `https://www.jeremyjordan.me/hyperparameter-tuning/`.

[38] Sarah Kamalpour and Hossein Khalafi. SMART reactor core design optimization based on FCM fuel. *Nuclear Engineering and Design*, page 110970, 2020. ISBN: 0029-5493 Publisher: Elsevier.

[39] K. Koebke. A new approach to homogenization and group condensation. Technical report, 1980.

[40] Takaaki Koyanagi, Kurt Terrani, Shay Harrison, Jian Liu, and Yutai Katoh. Additive manufacturing of silicon carbide for nuclear applications. *Journal of Nuclear Materials*, 543:152577, 2020. ISBN: 0022-3115 Publisher: Elsevier.

[41] Sivam Krish. A practical generative design method. *Computer-Aided Design*, 43(1):88–100, January 2011. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0010448510001764`, `doi:10.1016/j.cad.2010.09.009`.

[42] David J. Kropaczek and Ryan Walden. Constraint annealing method for solution of multiconstrained nuclear fuel cycle optimization problems. *Nuclear Science and Engineering*, 193(5):506–522, 2019. ISBN: 0029-5639 Publisher: Taylor & Francis.

[43] David J. Kropaczek and Ryan Walden. Large-Scale Application of the Constraint Annealing Method for Pressurized Water Reactor Core Design Optimization. *Nuclear Science and Engineering*, 193(5):523–536, 2019. ISBN: 0029-5639 Publisher: Taylor & Francis.

[44] David L. Krumwiede, Raluca O. Scarlat, Jae Keun Choi, Tung M. Phan, and Per F. Peterson. Three-dimensional modeling of the pebble-bed fluoride-salt-cooled, high-temperature reactor (PB-FHR) commercial plant design. In *Proceedings of the American Nuclear Society 2013 Winter Meeting*, 2013.

[45] 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, November 2015. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454915002285, doi:10.1016/j.anucene.2015.04.028.

[46] Jaakko Leppanen, Maria Pusa, Tuomas Viitanen, Ville Valtavirta, and Toni Kaltiaisenaho. The Serpent Monte Carlo code: Status, development and applications in 2013. *Annals of Nuclear Energy*, 82:142–150, August 2014. doi:10.1016/j.anucene.2014.08.024.

[47] Hsun-Chia Lin. *Thermal Hydraulics System-Level Code Validation and Transient Analyses for Fluoride Salt-Cooled High-Temperature Reactors*. PhD thesis, 2020.

[48] B. A. Lindley, J. G. Hosking, P. J. Smith, D. J. Powney, B. S. Tollit, T. D. Newton, R. Perry, T. C. Ware, and P. N. Smith. Current status of the reactor physics code WIMS and recent developments. *Annals of Nuclear Energy*, 102:148–157, 2017. ISBN: 0306-4549 Publisher: Elsevier.

[49] Alexander Lindsay. Moltres, software for simulating Molten Salt Reactors, 2017. https://github.com/arfc/moltres. URL: https://github.com/arfc/moltres.

[50] Alexander Lindsay, Gavin Ridley, Andrei Rykhlevskii, and Kathryn Huff. Introduction to Moltres: An application for simulation of Molten Salt Reactors. *Annals of Nuclear Energy*, 114:530–540, April 2018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454917304760, doi:10.1016/j.anucene.2017.12.025.

[51] Markus List, Peter Ebert, and Felipe Albrecht. Ten Simple Rules for Developing Usable Software in Computational Biology. *PLOS Computational Biology*, 13(1):e1005265, January 2017. URL: https://dx.plos.org/10.1371/journal.pcbi.1005265, doi:10.1371/journal.pcbi.1005265.

[52] Limin Liu, Dalin Zhang, Qing Lu, Kunpeng Wang, and Suizheng Qiu. Preliminary neutronic and thermal-hydraulic analysis of a 2 MW Thorium-based Molten

Salt Reactor with Solid Fuel. *Progress in Nuclear Energy*, 86:1–10, January 2016. URL: https://linkinghub.elsevier.com/retrieve/pii/S0149197015300779, doi:10.1016/j.pnucene.2015.09.011.

[53] Longwei Mei, Xiangzhou Cai, Dazhen Jiang, Jingen Chen, Yuhui Zhu, Yafen Liu, and Xiaohe Wang. The investigation of thermal neutron scattering data for molten salt Flibe. *Journal of Nuclear Science and Technology*, 50(7):682–688, 2013. ISBN: 0022-3131 Publisher: Taylor & Francis.

[54] Lawrence Murr. Frontiers of 3D Printing/Additive Manufacturing: from Human Organs to Aircraft Fabrication. *Journal of Materials Science & Technology*, 32(10):987–995, October 2016. URL: https://www.sciencedirect.com/science/article/pii/S1005030216301335, doi:10.1016/j.jmst.2016.08.011.

[55] NCSA. About Blue Waters: The National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign, 2017. http://www.ncsa.illinois.edu/. URL: http://www.ncsa.illinois.edu/.

[56] Andrew Ng, Kian Katanforoosh, and Younes Bensouda Mourri. Improving Deep Neural Networks: Hyperparameter Tuning, Regularization and Optimization, 2021. URL: https://www.coursera.org/learn/deep-neural-network.

[57] Faisal Y. Odeh and Won Sik Yang. Core design optimization and analysis of the Purdue Novel Modular Reactor (NMR-50). *Annals of Nuclear Energy*, 94:288–299, 2016. ISBN: 0306-4549 Publisher: Elsevier.

[58] James M. Osborne, Miguel O. Bernabeu, Maria Bruna, Ben Calderhead, Jonathan Cooper, Neil Dalchau, Sara-Jane Dunn, Alexander G. Fletcher, Robin Freeman, Derek Groen, Bernhard Knapp, Greg J. McInerny, Gary R. Mirams, Joe Pitt-Francis, Biswa Sengupta, David W. Wright, Christian A. Yates, David J. Gavaghan, Stephen Emmott, and Charlotte Deane. Ten Simple Rules for Effective Computational Research. *PLoS Computational Biology*, 10(3):e1003506, March 2014. URL: https://dx.plos.org/10.1371/journal.pcbi.1003506, doi:10.1371/journal.pcbi.1003506.

[59] Sun Myung Park. Advancement and Verification of Moltres for Molten Salt Reactor Safety Analysis. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL, August 2020. URL: https://www.ideals.illinois.edu/handle/2142/108542.

[60] Cludio 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. ISBN: 0306-4549 Publisher: Elsevier.

[61] Cludio M.N.A. Pereira and Wagner F. Sacco. A parallel genetic algorithm with niching technique applied to a nuclear reactor core design optimization problem. *Progress in Nuclear Energy*, 50(7):740–746, September 2008. URL: https://linkinghub.elsevier.com/retrieve/pii/S014919700800019X, doi:10.1016/j.pnucene.2007.12.007.

[62] Christian S. Perone. Pyevolve: a Python open-source framework for genetic algorithms. *Acm Sigevolution*, 4(1):12–20, 2009.

[63] Bojan Petrovic and Kyle M. Ramey. FHR/AHTR NEA Benchmark, October 2019. URL: http://montecarlo.vtt.fi/mtg/2019_Atlanta/Petrovic1.pdf.

[64] Bojan Petrovic, Kyle M. Ramey, Ian Hill, E. Losa, M. Elsawi, Z. Wu, C. Lu, J. Gonzalez, D. Novog, G. Chee, K. D. Huff, M. Margulis, N. Read, and Eugene Shwageraus. Preliminary results for the NEA FHR benchmark phase I-A and I-B (submitted). In *ANS M&C 2021 - The International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering*, Raleigh, North Carolina, 2021. ANS.

[65] David Petti. The future of nuclear energy in a carbon-constrained world. *Massachusetts Institute of Technology Energy Initiative (MITEI)*, page 272, 2018.

[66] Farzad Rahnema, David Diamond, Dumitru Serghiuta, and Paul Burke. Phenomena, gaps, and issues for neutronics modeling and simulation of FHRs. *Annals of Nuclear Energy*, 123:172–179, January 2019. URL: http://www.sciencedirect.com/science/article/pii/S0306454918304572, doi:10.1016/j.anucene.2018.08.035.

[67] Farzad Rahnema, Bojan Petrovic, Christopher Edgar, Dingkang Zhang, Pietro Avigni, Michael Huang, and Stefano Terlizzi. The Current Status of the Tools for Modeling and Simulation of Advanced High Temperature Reactor Neutronics Analysis. Technical report, Georgia Institute of Technology, 2015.

[68] Kyle M. Ramey and Bojan Petrovic. Monte Carlo modeling and simulations of AHTR fuel assembly to support V&V of FHR core physics methods. *Annals of Nuclear Energy*, 118:272–282, August 2018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454918301816, doi:10.1016/j.anucene.2018.04.003.

[69] Gbor Renner and Anik Ekrt. Genetic algorithms in computer aided design. *Computer-aided design*, 35(8):709–726, 2003.

[70] Paul K. Romano and Benoit Forget. The OpenMC Monte Carlo particle transport code. *Annals of Nuclear Energy*, 51:274–281, January 2013. URL: http://www.sciencedirect.com/science/article/pii/S0306454912003283, doi:10.1016/j.anucene.2012.06.040.

[71] Jhonathan Rosales, Isabella J. van Rooyen, and Clemente J. Parga. Characterizing surrogates to develop an additive manufacturing process for U3Si2 nuclear fuel. *Journal of Nuclear Materials*, 518:117–128, 2019. ISBN: 0022-3115 Publisher: Elsevier.

[72] Wagner F. Sacco, Hermes Alves Filho, Nlio Henderson, and Cassiano RE de Oliveira. A Metropolis algorithm combined with NelderMead Simplex applied to nuclear reactor core design. *Annals of Nuclear Energy*, 35(5):861–867, 2008. ISBN: 0306-4549 Publisher: Elsevier.

[73] Wagner F. Sacco and Cludio MNA Pereira. Two stochastic optimization algorithms applied to nuclear reactor core design. *Progress in Nuclear Energy*, 48(6):525–539, 2006. ISBN: 0149-1970 Publisher: Elsevier.

[74] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology*, 9(10):e1003285, October 2013. 00018. URL: `http://dx.plos.org/10.1371/journal.pcbi.1003285`, `doi:10.1371/journal.pcbi.1003285`.

[75] Cdric Sauder. Ceramic matrix composites: nuclear applications. *Ceramic matrix composites: materials, modeling and technology*, pages 609–646, 2014. Publisher: Wiley Online Library.

[76] Raluca O. Scarlat, Michael R. Laufer, Edward D. Blandford, Nicolas Zweibaum, David L. Krumwiede, Anselmo T. Cisneros, Charalampos Andreades, Charles W. Forsberg, Ehud Greenspan, Lin-Wen Hu, and Per F. Peterson. Design and licensing strategies for the fluoride-salt-cooled, high-temperature reactor (FHR) technology. *Progress in Nuclear Energy*, 77:406–420, November 2014. URL: `http://www.sciencedirect.com/science/article/pii/S0149197014001838`, `doi:10.1016/j.pnucene.2014.07.002`.

[77] Raluca O. Scarlat and Per F. Peterson. The current status of fluoride salt cooled high temperature reactor (FHR) technology and its overlap with HIF target chamber concepts. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 733:57–64, 2014. ISBN: 0168-9002 Publisher: Elsevier. URL: `http://www.sciencedirect.com/science/article/pii/S0168900213007055`, `doi:10.1016/j.nima.2013.05.094`.

[78] Herbert A. Simon. *The sciences of the artificial*. MIT press, 2019.

[79] Joseph Simpson, James Haley, Corson Cramer, Olivia Shafer, Amy Elliott, Bill Peter, Lonnie Love, and Ryan Dehoff. CONSIDERATIONS FOR APPLICATION OF ADDITIVE MANUFACTURING TO NUCLEAR REACTOR CORE COMPONENTS. page 43, May 2019.

[80] Lance L. Snead, Takashi Nozawa, Yutai Katoh, Thak-Sang Byun, Sosuke Kondo, and David A. Petti. Handbook of SiC properties for fuel performance modeling. *Journal of nuclear materials*, 371(1-3):329–377, 2007. ISBN: 0022-3115 Publisher: Elsevier.

[81] Vladimir Sobes, Briana Hiscox, Emilian Popov, Marco Delchini, Richard Archibald, Paul Laiu, Ben Betzler, and Kurt Terrani. ARTIFICIAL INTELLIGENCE DESIGN OF NUCLEAR SYSTEMS EMPOWERED BY ADVANCED MANUFACTURING. page 8, 2020.

[82] Niyanth Sridharan, Maxim N. Gussev, and Kevin G. Field. Performance of a ferritic/martensitic steel for nuclear reactor applications fabricated using additive manufacturing. *Journal of Nuclear Materials*, 521:45–55, 2019. ISBN: 0022-3115 Publisher: Elsevier.

[83] ASTM Standard. Standard terminology for additive manufacturing technologies. *ASTM International F2792-12a*, 2012.

[84] K. A. Terrani. Transformational Challenge Reactor demonstration program. Technical report, 2019. URL: `Retrievedfromhttp://tcr.ornl.gov.`

[85] Kurt A. Terrani, J. O. Kiggans, Chinthaka M. Silva, C. Shih, Yutai Katoh, and Lance Lewis Snead. Progress on matrix SiC processing and properties for fully ceramic microencapsulated fuel form. *Journal of Nuclear Materials*, 457:9–17, 2015. ISBN: 0022-3115 Publisher: Elsevier.

[86] Mina Torabi, A. Lashkari, Seyed Farhad Masoudi, and Somayeh Bagheri. Neutronic analysis of control rod effect on safety parameters in Tehran Research Reactor. *Nuclear Engineering and Technology*, 50(7):1017–1023, 2018. ISBN: 1738-5733 Publisher: Elsevier.

[87] Michael P. Trammell, Brian C. Jolly, M. Dylan Richardson, Austin T. Schumacher, and Kurt A. Terrani. Advanced Nuclear Fuel Fabrication: Particle Fuel Concept for TCR. Technical report, 2019.

[88] P.J. Turinsky, R.M.K. Al-Chalabi, P. Engrand, H.N. Sarsour, F.X. Faure, and W. Guo. NESTLE: Few-group neutron diffusion equation solver utilizing the nodal expansion method for eigenvalue, adjoint, fixed-source steady-state and transient problems. Technical Report EGG-NRE–11406, 10191160, June 1994. URL: `http://www.osti.gov/servlets/purl/10191160-pnc6Jn/webviewable/`, `doi:10.2172/10191160`.

[89] Adrin Uriondo, Manuel Esperon-Miguez, and Suresh Perinpanayagam. The present and future of additive manufacturing in the aerospace sector: A review of important aspects. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 229(11):2132–2147, 2015. ISBN: 0954-4100 Publisher: SAGE Publications Sage UK: London, England.

[90] Venugopal Koikal Varma, David Eugene Holcomb, Fred J. Peretz, Eric Craig Bradley, Dan Ilas, A. L. Qualls, and Nathaniel M. Zaharia. AHTR mechanical, structural, and neutronic preconceptual design. Technical report, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2012.

[91] Dingkang Zhang and Farzad Rahnema. Integrated Approach to Fluoride High Temperature Reactor Technology and Licensing Challenges (FHR-IRP). Technical report, Georgia Inst. of Technology, Atlanta, GA (United States), 2019.

[92] Y. Zhu and A. I. Hawari. Thermal neutron scattering cross section of liquid FLiBe. *Progress in Nuclear Energy*, 101:468–475, 2017. ISBN: 0149-1970 Publisher: Elsevier.