Utilizing a Reduced-Order Model and Physical Programming for Preliminary Reactor Design Optimization

Ryan Stewart[1] **and Todd S. Palmer**[1]
[1]Oregon State University 1500 SW Jefferson St. Corvallis, OR 97331

stewryan@oregonstate.edu, palmerts@engr.orst.edu

ABSTRACT

Reactor core design is inherently a multi-objective problem which spans a large design space, and potentially larger objective space. This process relies on high-fidelity models to probe the design space, and sophisticated computer codes to calculate the important physics occurring in the reactor. In the past, the design space has been reduced by individuals with extensive knowledge of reactor core design; however, this approach is not always available. In this paper, we utilize a set of high-fidelity models to generate a reduced-order model, and couple this with a genetic algorithm to quickly and effectively optimize a preliminary design for a prototypical sodium fast reactor. We also examine augmenting the genetic algorithm with physical programming to generate the fitness function(s) that evaluates the degree to which a core has been optimized. Physical programming is used in two variations of multi-objective optimization and is compared with a traditional weighting scheme to examine the solutions present on the Pareto front.

Optimization on the reduced-order model produces a set of solutions on the Pareto front for a designer to examine. The uncertainty for the objective functions examined in the reduced-order model is less than 7% for the given designs, and improves as additional data points are employed. Utilizing a reduced-order model can significantly reduce the computation time and storage to perform preliminary optimization. Physical programming was shown to reduce the objective space when compared with a traditional weighting scheme. It also provides an intuitive and computationally efficient way to produce a Pareto front that meets the designer's objectives.

KEYWORDS: multi-objective optimization, physical programming, reduced-order modeling, genetic algorithm, core design

1. INTRODUCTION

Nuclear reactor core design can involve vast numbers of independent variables for designers, which grants them the freedom to creatively solve notoriously difficult problems with elegant methods. This freedom of design can have unintended consequences in attempting to understand the impact these choices have on the core, which may increase the time to converge on an appropriate design. This paper examines two major topics to aid the optimization process: utilizing a reduced-order model to perform preliminary reactor design, and examining the effectiveness of physical programming as a fitness function for a genetic algorithm. Our primary goal is to reduce the computational

time required to perform preliminary reactor core optimization. In addition, we explore the potential for designers to translate design goals into quantitative objectives via physical programming to generate a feasible reactor design with minimal iterations.

To reduce the computational time required to find an optimal core, multiple approaches have been examined. One approach is dimensionality reduction, which seeks to reduce the number of energy groups or angles when solving the transport equation. This can also take the form of homogenization to reduce spatial complexities. The software package CASMO-5 does this by solving a high-fidelity lattice-physics problem for macroscopic cross-sections using a representative geometry [1]. Data is calculated at multiple state points within the design space to generate an accurate representation. This data is used for physics-informed dimensionality reduction to create a reduced-order model, which can reduce the number of energy groups, angles, and introduce spatial homogenization. Much of the physics associated with the higher-fidelity model is maintained; however, a lower-order diffusion equation is solved in place of the neutron transport equation. The reduced-order model can then be employed by an optimization algorithm to solve for multiple reactor configurations, where different states of the reactor can be determined by interpolation between the various state points without reverting back to the high-fidelity code.

Another approach has been to utilize surrogate modeling to reduce the number of high-fidelity computations required. Surrogate modeling uses a mathematical representation to determine objectives based on a set of design variables, and has previously been explored by multiple authors. Fabbris et al [2] utilized artificial neural networks to create surrogate models for sodium fast reactor optimization. Utilizing artificial neural networks provided a high-degree of accuracy for the surrogates, but required over 8,000 separate high-fidelity models for the 30 design variables used. Although highly-accurate, this approach may not be achievable for small design teams. We augment this approach by reducing the required number of high-fidelity model to create a a reduced order model. In our implementation, the reduced order model is a small database of high-fidelity solutions in conjunction with a multi-dimensional linear interpolator. A small database allows a designer to perform rapid trade-off analyses on broad integral design variables. Given the reduced size of the database, there will be an inherent reduction in accuracy. This can be overcome in future design iterations after ranges for the initial design variables have been established.

Multi-objective optimization (MOO) helps to alleviate the task of finding optimal designs by providing algorithms which sort solutions based on their applicability to the designer's goals. However, determining what constitutes an optimized design can be arduous. Designers are often required to assign weights to objectives deemed important, but this task is not intuitive and must be iterated to produce a viable design. In this research, physical programming will be implemented as a substitute for standard weighting functions. We apply methods similar to those of Reynoso-Meza et al [3], where physical programming is integrated directly into an evolutionary algorithm to perform the optimization. Their approach is applied to a set of standard benchmark design problems, where it is found to be competitive with genetic algorithms and helps control the size of the acceptable solutions for the benchmark problems. In this paper, we describe a decoupled method where physical programming provides the fitness function to the genetic algorithm, which is used to determine solution acceptability. This method is applied to the design of a small prototypical sodium fast reactor, with a secondary goal of creating a more intuitive framework for engineers to perform MOO.

1.1. Multi-objective optimization

Preliminary reactor design involves examining a large design space to find a set of designs which meet external objectives. The design space consists of independent variables (materials, geometries, etc), which are used to determine objective functions (power density, reactivity feedback, etc.). To find solutions within the design space that meet the objectives, a MOO algorithm can be used. Eq.1 describes an optimization problem, where **F** is the objective vector being optimized [4]. The n design variables are described by $\mathbf{x} = (x_1, x_2, ..., x_n)$, where $f_k(\mathbf{x})$ describes the k different objective functions. Other constraints can be placed on the problem, and are denoted as g_j and h_l, which are a set of m inequality and e equality constraints. The goal is to find a set of design variables ($\hat{\mathbf{x}}$), which simultaneously optimizes the k objective functions, thus producing an objective vector (solution) **F**.

$$\begin{aligned} min\ \mathbf{F}(\mathbf{x}) &= (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x}))^T \\ \text{subject to } g_j(\mathbf{x}) &\leq 0, \quad j = 1, 2, ..., m \\ h_l(\mathbf{x}) &= 0, \quad l = 1, 2, ..., e \end{aligned} \tag{1}$$

The objective vector **F** can be considered a mapping function between the design and objective spaces, as seen in Figure 1. For a set of design variables, denoted **a**, **b**, and **c**, and objective functions f_1 and f_2, there exists a corresponding set of objective vectors $\mathbf{F}_a$, $\mathbf{F}_b$, and $\mathbf{F}_c$ in objective space. The goal of a MOO algorithm is to find solutions which are optimal; where optima for objective functions f_1 and f_2 are denoted f_1^* and f_2^*, respectively. A solution which is able to optimize f_1 and f_2 simultaneously is denoted by $\mathbf{F}^*$. $\mathbf{F}^*$ is often not obtainable and it falls to the designer to find a solution which simultaneously optimizes f_1 and f_2. Solutions which can simultaneously optimize f_1 and f_2 are said to be Pareto optimal, and fall on the Pareto front (**P**), as seen in Figure 1. The remainder of solutions either do not meet the objectives laid out for them (i.e. $\mathbf{F}_a$, which falls outside of the objective space), or fall within the objective space but are not optimal.

To help define the quality of a solution, the term *dominance* is used. If a given objective vector **F** is equal to, or better, in all objective functions ($f_i(\mathbf{b}) \leq f_i(\mathbf{c})$), and strictly better in at least one objective function ($f_i(\mathbf{b}) < f_i(\mathbf{c})$), then $\mathbf{F}_b$ dominates $\mathbf{F}_c$. The set of solutions which are non-dominated are called the Pareto set and fall on the Pareto front, where they describe optimal solutions to the design. Once the Pareto front has been found it is up to the designer to determine which objectives are more important, i.e. where on the Pareto front they should select their solution.

Two different methods are employed in this research; a Single Objective Genetic Algorithm (SOGA) and a Multi-Objective Genetic Algorithm (MOGA). Both methods are used to search the design space of sodium fast reactors for optimal designs. A genetic algorithm (GA) is an optimization tool which promotes superior solutions via evolutionary concepts to explore areas of the design space which are optimal [5]. A chromosome denotes the set of design variables (**x**), and within the chromosome, a gene denotes a specific design variable (x_i). Each gene can take on multiple values, denoted as an allele, which encompasses a particular dependent variable value. A set of chromosomes in a genetic algorithm is denoted as a population, and this population is examined for the most optimal chromosomes (solutions).

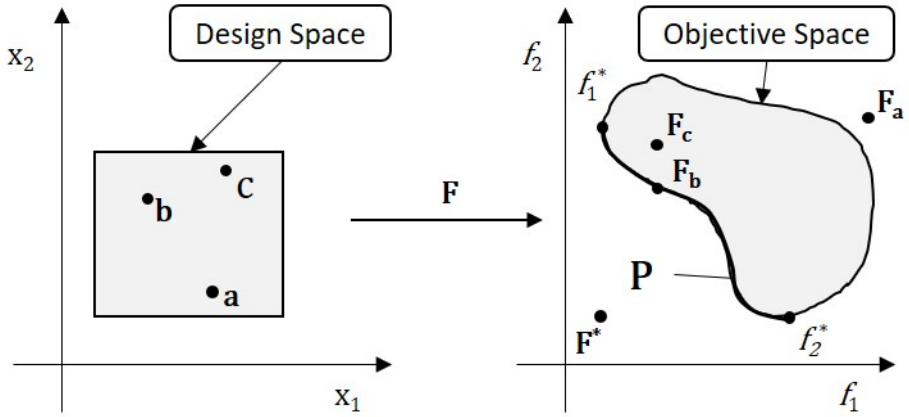


Figure 1: MOO Mapping

GAs utilize multiple generations of populations to determine where optimal solutions lie in the design space. For each generation, the best chromosomes are determined, and will be used to create new chromosomes to explore additional areas of the design space. Two methods are typically employed to generate new chromosomes: crossover and mutation. Crossover exchanges gene segments between superior chromosomes, and mutation introduces a change, at random, to a single gene. The algorithm stops after a set number of fitness function evaluations or generations, where the most optimal solutions are typically on the Pareto front, which provides the designer with a range of solutions to select from.

The viability of a chromosome in a GA is typically determined via the ranking of fitness functions. There are a few ways in which fitness functions can be utilized to determine a solution's viability. One common method is to condense the multi-objective problem into a single objective, where each objective function is combined with an associated weight and summed. This method is referred to as a weighted linear approach, and is utilized by the SOGA method. The approach utilizes a weight (w_i) to denote the importance of each objective function (f_i), where the product of these values are summed, and the resulting fitness function (FF) is to be minimized. Eq. 2 provides a mathematical representation for the fitness function, where the weights can be continually updated to explore different sections of the Pareto front until solutions which meet the designer's needs are found.

$$FF = \sum_{i=0}^{i=k} w_i \cdot f_i \tag{2}$$

MOGAs uses another approach to determine a solution's fitness. Instead of condensing the problem, the fitness is determined by the number of other chromosomes which dominate that solution [7]. If the limit of dominated solutions is breached, the chromosome is discarded. The Pareto front is explored simultaneously, where solutions which reside on or near the front will be maintained. Decreasing the limit of solutions which dominate the problem can reduce the size of the Pareto front. This approach helps circumvent the need for utilizing weights associated with the

SOGA approach; however, assigning weights in an intelligent fashion can also help reduce this burden.

To reduce the difficulties in assigning weights for the SOGA method, an optimization technique called physical programming (PP) can be used to create the fitness function. Physical programming employs a designer's knowledge of a problem to describe a set of physically meaningful constraints for the objective functions [8]. For each objective function, the designer assigns regions of acceptability ranging from unacceptable to highly desirable. Each objective function is also assigned a classification such as smaller-is-better, larger-is-better, or center-is-better, to describe how the objective function should be optimized [8]. Using the acceptability regions in conjunction with a given classification provides an optimization approach which is intuitively logical and eliminates the need for iterative weighing schemes.

An illustration of the two schemes can be seen in Figure 2. The left hand side of Figure 2 shows the exploration of the Pareto front based on the the weighting scheme. The red portion applies a large weight for objective 1, the orange region applies a large weight for objective 2, and the purple region applies nearly equal weights to both objectives. For physical programming, only a small region of the Pareto front is discovered, as this region provides desirable results for both objectives. Attempting to bring either objective function further towards highly desirable would decrease the other into the tolerable or undesirable range. This is achieved by defining acceptable values for objective functions *a priori* rather than continually manipulating weights to find an acceptable solution.

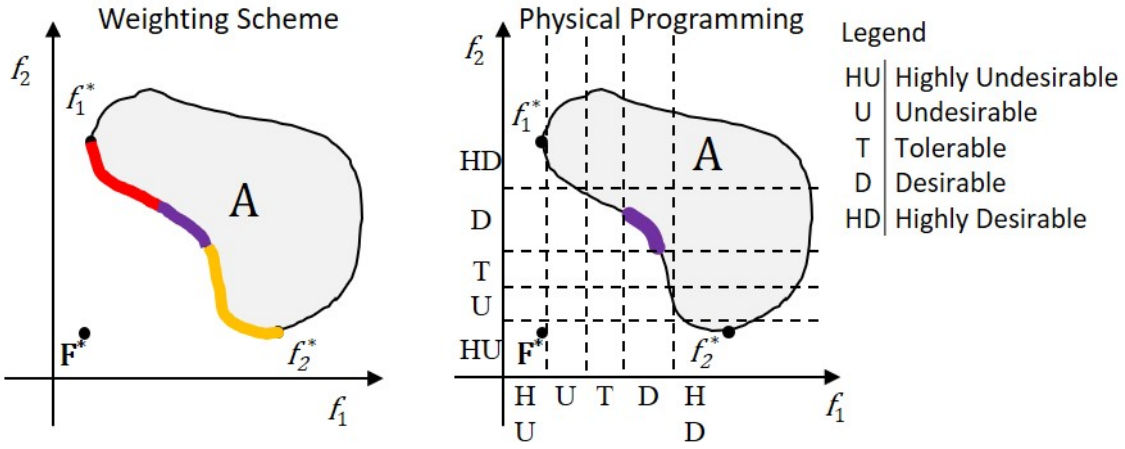


Figure 2: Pareto front for WS and PP

Physical programming uses a dynamic weighting scheme to determine the fitness of the objective function via the acceptability ranges. Each range from unacceptable to desirable is significantly better than the previous range, and the difference from desirable to highly desirable is only somewhat better. Within each range, if an objective function is closer to the next desirable region, it will have a higher fitness than an objective function which is closer to the next undesirable region. It is important to note that increasing an objective's fitness will result in a smaller fitness function value, as the fitness function is minimized. In Figure 2, moving objective 1 (f_1) from a range of

desirable to highly desirable would increase its fitness (thereby decreasing its fitness function) by a moderate amount. This same action would require objective 2 (f_2) to traverse from desirable to highly undesirable, which would significantly decrease its fitness. The resulting objective vector would be significantly larger (i.e. less fit) when compared to a solution with both objectives being desirable, and thus the solution would be rejected. These key distinctions have the potential to give physical programming a powerful advantage over simple weighting schemes. A more in-depth discussion of fitness functions generated by physical programming can be found in Messace [8].

1.2. Fast reactor design

Sodium fast reactors (SFRs) are a type of liquid-metal cooled fast reactor, that rely on fast neutrons to drive the fission reaction, rather than thermal neutron induced fission in typical light water reactors (LWRs). The reliance on fast neutrons increases the mean free path of a neutron, which tends to spatially couple the physics occurring in the core. This can have resounding effects on various safety parameters; for example SFR cores can have a positive reactivity feedback due to coolant voiding in an individual assembly, or on a core wide scale. When coolant is voided from the core, the average energy of neutrons in the system increases due to a lack of scattering with the coolant. The increase in neutron energy causes an increase in the probability of fast fission if it interacts with the fuel. This process is directly impacted by the leakage rate out of the core due to the long neutron path length. Thus, the smaller the core, the higher the leakage, which will produce a negative feedback mechanism if the core is voided. For a larger core, there is a higher probability that a neutron will interact with the fuel and cause a fission instead of leaking out, which could create a positive reactivity feedback.

An SFR core has design variables similar to those of an LWR, and can include fuel material, enrichment, fuel height, etc. SFRs have historically been fueled with either metallic or oxide based fuel. Metallic fuel holds multiple advantages over oxide fuel for SFRs due to its high thermal conductivity, high fuel loading, and compatibility with bond materials. The reliance on fast neutrons means that the cross-section for fission is much lower than in a LWR. Consequently, there is a need for increased fuel loading and total fissile mass in the core to sustain the reaction. This has often been achieved by increasing the enrichment, increasing the core size, or adding plutonium to the fuel. Increasing the core size can have negative impacts on safety parameters. To determine an appropriate core configuration, the fuel height and fuel smear are often adjusted to find optimal configurations to encourage negative feedback and minimize the core size. Fuel smear is a term in the fast reactor design literature that describes the fractional area of fuel inside the cladding, rather than describing the fuel diameter explicitly [6]. The fuel smear (A_f) relates the inner cladding (R_{IC}) to fuel radius (R_f) via $R_f = \sqrt{A_f} R_{IC}$.

2. MODEL

The design space utilized for optimization was a small SFR, which could be used as a test facility or small power plant. The design constraints can be seen in Table 1, where the total fissile material is 27 wt% of the fuel matrix, and the plutonium fraction is the percent of plutonium in the fissile material.

MCNP6 was used to produce the high-fidelity data points for beginning of life neutronic char-

Table 1: Design Variables

Design Variable	Lower Bound	Upper Bound
Fuel Height (cm)	50	80
Fuel Smear (%)	50	70
Pu Fraction (%)	0	100

acteristics, with input files produced by the Fast Reactor Input Deck Generator (FRIDGe) [9,10]. These simulations provided reactor physics parameters such as k_{eff}, β_{eff}, and the mean generation time, and derived quantities such as the Doppler and void coefficient. An initial model was created using a uniform temperature distribution of 900K, along with cross-sections generated from ENDF-VII.1 to determine normal operating conditions [11]. The next model reduced the sodium coolant density to 0.1 percent of the nominal value to simulate a voided core. To examine the Doppler coefficient, the temperature was reduced to 600K. Using these three models, normal operating conditions could be explored along with the void coefficient and the Doppler coefficient. The sodium void coefficient and the Doppler coefficient were then found using Eq. 3 and Eq. 4, respectively.

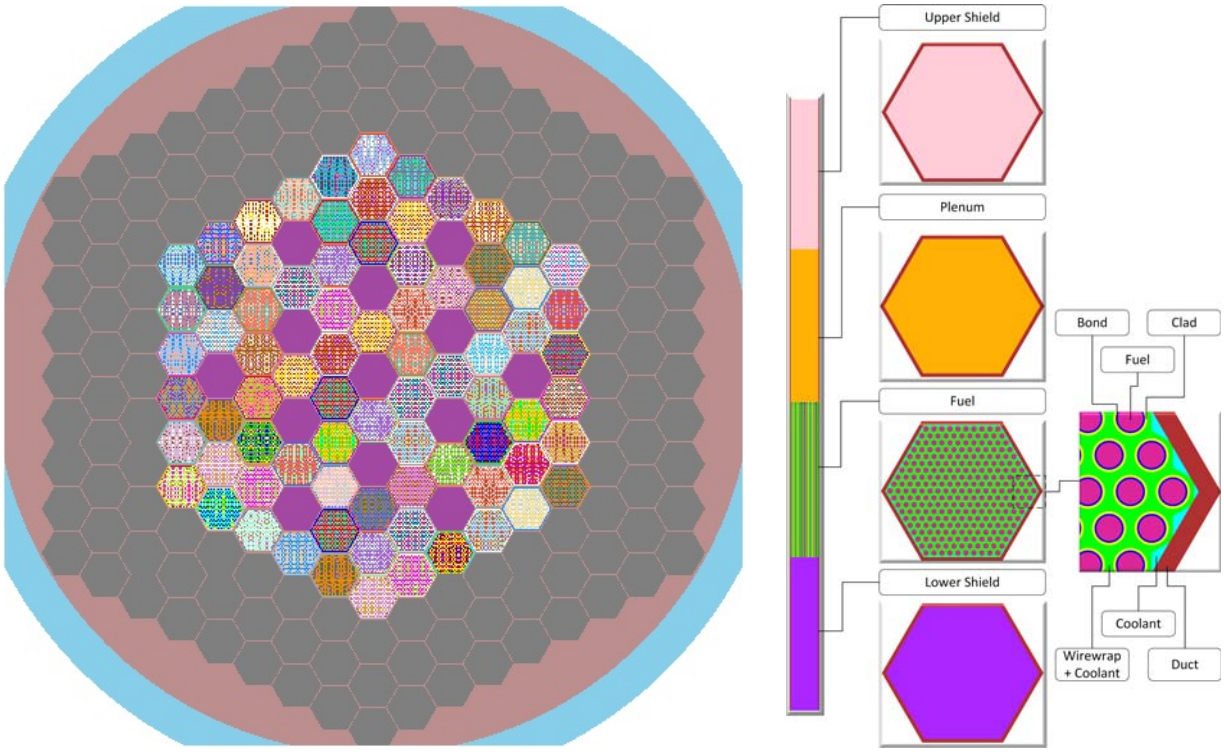
$$\alpha_v = \frac{\Delta\rho}{\Delta\%_{void}} \quad [\frac{pcm}{\%void}] \tag{3}$$

$$\alpha_f = \frac{\Delta\rho}{\Delta T} \quad [\frac{pcm}{K}] \tag{4}$$

The reactor model utilizes a hexagonal lattice of fuel, where the central core region consists of 78 fueled assemblies and 10 blank assemblies. Surrounding the core are three rows of stainless steel reflectors, a sodium region, and a 10 cm thick HT9 reactor vessel. Figure 3a shows a plan view of the core, where blank assemblies are seen in purple, reflector assemblies in grey, and the remaining multi-colored assemblies are fuel.

Three types of assemblies were created to simulate fuel assemblies, blank regions, and reflectors. Fuel assemblies contain six regions; upper/lower sodium, upper/lower reflector, plenum, and fuel. These regions can be seen in Figure 3b, where the sodium regions are 10 cm in height and contain 100 wt% sodium. The reflector regions are 55 cm in height and are 70 wt% HT9 and 30 wt% sodium, and the plenum has the same height as the fuel and consists of 25 wt% HT9, 25 wt% sodium, and 25 wt% void. The blank regions in the core consist of a homogenization of 90 wt% sodium and 10 wt% HT9, but could contain experiments or control rods. The reflectors are a homogeneous block of 30 wt% sodium and 70 wt% HT9.

The fuel region of the assembly consists of a hexagonal lattice with 271 fuel pins. The fuel pin height and smear are allowed to vary in accordance with Table 1, but the thickness, outer cladding,



(a) Plan View of the SFR Core **(b) Elevated View of a Fueled Assembly**

Figure 3: Assembly and Core Diagram

and pitch are fixed at 0.037 cm, 0.395 cm, and 0.661 cm respectively. Surrounding the fuel pin is a sodium bond which extends approximately 1.0 cm above the fuel. The wire wrap is homogenized with the coolant surrounding the fuel pin rather than being explicitly represented. All structural components for the fuel are HT9.

3. METHODOLOGY

3.1. Reduced-order model

Performing any type of optimization using high-fidelity models would be an inefficient use of time and resources. To avoid this, a set number of high-fidelity models were simulated to create a database which correlates the known design variables (height, smear, and plutonium content) to objective functions (k_{eff}, void coefficient, Doppler coefficient, and plutonium content). Data from each MCNP model was parsed and stored in a HDF5 database for accessibility [12,13]. The database contains core-wide integral properties (k_{eff}, β_{eff}, etc.) produced from an MCNP output file [9], along with a number of derived quantities, such as the void coefficient and Doppler coefficient. The reduced-order model only utilizes a subset of the data contained in the database, namely k_{eff} and the void/Doppler coefficient. This database provides a set of solutions, where some type of algorithm can be applied to determine an objective function given a set of design variables, without running a high-fidelity simulation. The database contains 36 entries spanning

the design space described in Section 2. Because the high-fidelity solution is generated using Monte Carlo neutron transport, there is an inherent statistical uncertainty in the data entries [9]. The uncertainty is maintained in the database, where 3-σ uncertainty is kept below 15 pcm for k_{eff}, $5.0\ \frac{pcm}{\%void}$ for α_v, and $0.010\ \frac{pcm}{K}$ for α_f. Additional work may be needed to further reduce these values.

A script translates the HDF5 database into a Pandas dataframe [14], which is easier to manipulate and access during the optimization process. Once the data is stored in a dataframe, the reduced-order model (ROM) is created using a custom module that wraps around scipy [15]. The *LinearNDInterpolator* module is used to perform linear interpolation to determine objective functions which are not explicitly given. For example, a given height, smear, and plutonium fraction is used to generate k_{eff}, the void coefficient, and the Doppler coefficient. This method currently assumes a linear relationship between the design variable points and the objective functions. Additional regression techniques could be employed in the future to create a more robust ROM.

3.2. Optimization Scheme

The optimization algorithms used in this research were the SOGA and MOGA algorithms contained within DAKOTA [7]. DAKOTA provides a framework for design optimization and easily interfaces with the ROM to generate an objective vector given a set of design variables and objective functions. For this method, the objective functions serve as the variables for the fitness function and are assessed via two different methods. The first is via a weighting scheme, and the second utilizes physical programming to determine Pareto optimal solutions. Both of these methods return the fitness function back to DAKOTA, where the process is repeated until some convergence criteria is met, or a set number of generations has passed. After this process, DAKOTA returns a set number of optimized solutions for the designer to examine. The process can be seen in Figure 4.

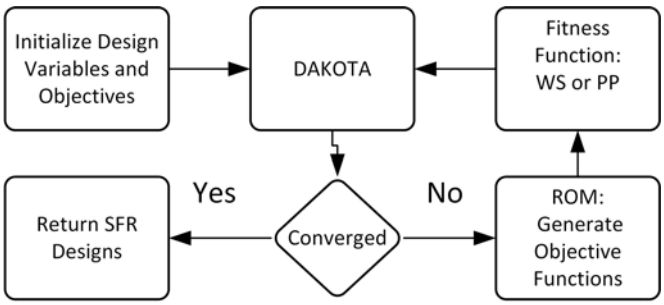


Figure 4: DAKOTA Interface for Optimization

The determination of the fitness function is the major difference between the two approaches. It was hypothesized that utilizing physical programming to provide a fitness function would yield a more intuitive optimization framework, and produce solutions which better represent the designer's goals. The python package *physprog* was used to determine the fitness functions based on physical programming [16]. The ranking scheme for physical programming can be seen in Table 2, and provides a range for the four objective functions to allow for a comparison with the weighting

scheme. The value given for each range is the lowest value for that range, i.e. highly undesirable for k_{eff} has a range from 1.00 to 1.025, and any value below highly undesirable is unacceptable.

Table 2: Ranking Scheme for Physical Programming

Objective Function	Class	Highly Undesirable	Undesirable	Tolerable	Desirable	Highly Desirable
k_{eff}	Larger is Better	1.00	1.025	1.05	1.075	1.10
α_v	Smaller is Better	-50	-75	-100	-150	-200
α_f	Smaller is Better	0.0	-0.25	-0.5	-0.75	-1.0
Pu %	Smaller is Better	90	80	70	60	50

Eq. 5 shows the representation of the fitness function for each objective function in the problem. Each fitness function is scaled based on the maximum/minimum values in the database, which will allow the weights to be distributed evenly.

$$\begin{aligned} FF_k &= 1 - (k_{eff} - k_{eff,min})/(k_{eff,max} - k_{eff,min}) \\ FF_v &= 1 - (\alpha_v - \alpha_{v,min})/(\alpha_{v,max} - \alpha_{v,min}) \\ FF_d &= 1 - (\alpha_f - \alpha_{f,min})/(\alpha_{f,max} - \alpha_{f,min}) \\ FF_{pu} &= Pu\% \end{aligned} \tag{5}$$

4. RESULTS

4.1. Reduced-Order Model

The first set of results explores the ability of the ROM to provide adequate solutions for a multi-objective optimization problem. The fitness functions in the previous section were used, with the exception that the fitness function for the plutonium fraction was absent, and a k_{eff} of 1.0 was required. Removing the fitness function for the plutonium fraction provides an easier interpretation for the results in this section. Optimization was performed using MOGA to discover the Pareto front [7]. This can be seen in Figure 5, where the color scheme shows the fraction of plutonium for the solutions presented.

Figure 5 illustrates that the Pareto front is large and diverse in the objective space, where the plutonium fraction plays a large role in determining where on the Pareto front the solution falls. Utilizing the Pareto front, the designer can examine a set of optimal solutions and determine which objective vector they deem the ‘best’ solution. A solution from this set can then be selected based on any other feature (such as economics) that was not captured in the optimization process. Table 3 shows a selection of solutions that were presented as optimal for the MOGA. The objective functions are given by the ROM, where a comparison to the high-fidelity solutions will be discussed later.

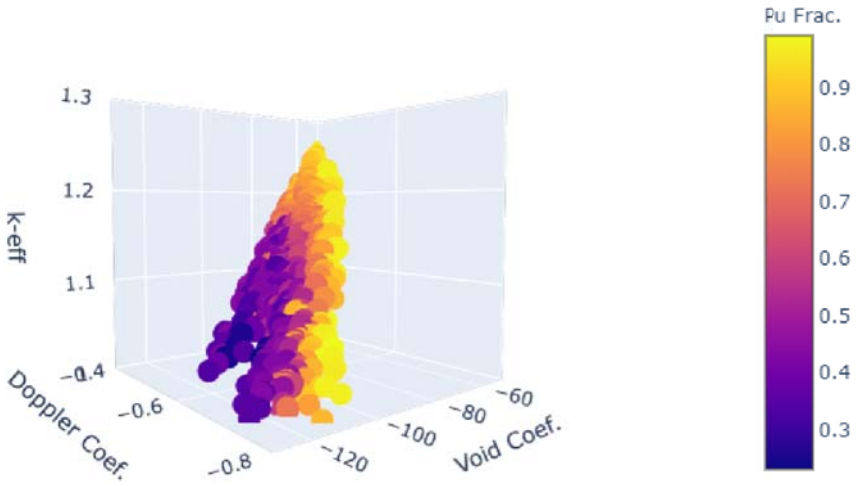


Figure 5: Pareto front for ROM

Table 3: Optimal Solutions for SFR Preliminary Design

	Height	Smear	Pu %	k_{eff}	α_v	α_f
Design 1	61.37	51.58	73.40	1.004	-113.9	-0.7439
Design 2	59.72	50.01	86.94	1.002	-111.4	-0.7665
Design 3	71.06	55.77	35.36	1.003	-124.1	-0.7157

A particular trend appears from the results of the optimization. The Pareto front shows a diversity of solutions; however, solutions presented as the 'most' optimal are relatively similar and tend to favor designs which maximize safety constraints at the expense of reactivity. This likely arises because the safety constraints are often optimized concurrently, and in opposition to k_{eff}. While a designer could easily examine the Pareto front to find additional solutions, it is worthwhile to note this phenomena, as it will be discussed further in Section 4.2.

We now turn to an examination of the accuracy of the ROM. High-fidelity models for the designs in Table 3 were simulated and compared, where the percent difference for each of the objectives can be seen in the columns labeled '36 DB - ROM' in Table 4. The difference in k_{eff} is relatively low; however, this difference equates to 1500 pcm for Design 3, which is significant. The most egregious error is from the model using a low plutonium fraction. This is due to the fact that the relationship between plutonium content and the objective functions is likely to be nonlinear between the current data points.

The genesis of this data was previous research performed by the authors, which examined material and geometric effects separately [17,18]. To transform this data into the current format, a smaller

set of data points were selected from the design space. This created a database with 36 entries spanning the design space and produces inadequate results, even for preliminary analysis. One source of error into the ROM is the uncertainty in the MCNP model. The uncertainty for an individual simulation is relatively small; when attempting to interpolate these values, the error is likely to compound. The estimation of the Doppler coefficient may be more sensitive to this error, despite the fact that uncertainty was kept below $0.01\ \frac{pcm}{K}$. However, this is a 3% uncertainty, which propagates in the ROM. Reducing the uncertainty in the Doppler coefficient would likely help increase the accuracy of the ROM.

Another aspect of the difference between the ROM and high-fidelity model is the lack of resolution in the database, but this could be improved upon if additional high-fidelity simulations were performed. To investigate this, data for two additional plutonium contents (20% and 75%) were added to the database. The total number of data points in the ROM is increased to 48 and 60, respectively. Adding data requires a degree of human interaction to develop a database which is sparse enough to reduce computational resources, but sufficiently resolved enough to provide accurate predictions using the ROM. The difference in objective functions is presented in Table 4.

Table 4: Percent Difference in Objective Function between MCNP & ROM

	36 DB - ROM			48 DB - ROM			60 DB - ROM		
	k_{eff}	α_v	α_f	k_{eff}	α_v	α_f	k_{eff}	α_v	α_f
Design 1	-0.06	3.08	5.97	-0.27	1.91	6.79	-0.16	1.91	6.79
Design 2	-0.07	0.95	7.27	-0.07	0.94	7.26	-0.05	0.94	7.26
Design 3	-1.57	9.89	18.43	-0.08	2.56	9.32	-0.08	2.56	6.61

Adding these additional sets of data reduces the overall inaccuracy of the ROM when compared with the original data set. Although only three designs are being compared for accuracy, we expect that increasing the amount of data points will increase the global accuracy for the ROM. However, a balance must be struck to prevent excessive computations. This balance is determined by the designer, who would decide what degree of accuracy is required for the surrogate model.

Finally, we consider the efficiency of creating the database compared to the optimization process. The computational time required to run a single high-fidelity simulation was $\sim$8,500 minutes, using 24 cores on Oregon State University's high performance cluster. Each core design requires three models to produce k_{eff}, the void coefficient, and the Doppler coefficient. The total computational burden is shown in Table 5, which offers a comparison between the time and size requirements required to produce an optimal solution. The size requirement given for the database refers to the size of the H5 file, and the 'MOGA w\o the ROM' denotes the size of the accumulated MCNP output files (10,000 in total). Both databases require a set of MCNP outputs initially; however once the database is created they are no longer required for the ROM. This allows the optimization process to be added with relative ease, without storing or manipulating gigabits of data associated with the MCNP output files. While this was not an original focus of investigation in this research, it does provide further evidence for using a database structure to reduce the burden of data storage.

The combination of time and size savings indicates the power of utilizing a ROM for performing optimization analysis. Once the high-fidelity models have been simulated, different portions of the Pareto front can be explored within minutes to fine-tune the selection process.

Table 5: Computational Efficiency

	ROM - 36	ROM - 48	ROM - 60	MOGA w/o ROM
Time (min)	$9.18 \cdot 10^5$	$1.224 \cdot 10^6$	$1.53 \cdot 10^6$	$2.55 \cdot 10^8$
Size (Mb)	0.952	1.3	1.6	$94.5 \cdot 10^3$

4.2. Physical Programming as a Fitness Function

In this section of results, we evaluate the use of physical programming as a fitness function for performing optimization using both SOGA and MOGA. Here, we assume that the ROM is adequate, and as such, only compare results relative to the previously described 60 entry ROM. For both the SOGA and MOGA algorithms, the four fitness functions from Section 3.2 were used to find a Pareto front. Minor changes were made to the hyper-parameters of both algorithms to provide an adequate Pareto front, where 2,500 and 10,000 fitness function evaluations were used as a stopping criteria for each SOGA and MOGA, if solution convergence had not been met.

In the SOGA algorithm, over 250 optimization calculations were required to generate a Pareto front for the traditional weighting scheme method. The process involves perturbing the weights (w_i) for each objective function in a uniform manner (i.e. the sum of the weights was equal to 1.0). A selection of results is shown in Table 6, with each objective optimized individually (Designs 4-7), a uniform weighting (Design 8), and an attempt at a critical core (Design 9).

Table 6: Select Designs on Pareto front for SOGA using Weighting Scheme

	Height	Smear	Pu %	k_{eff}	α_v	α_f	w_k	w_v	w_f	w_{pu}
Design 4	79.9	69.95	98.25	1.2850	-44.8	-0.3642	1.0	0.0	0.0	0.0
Design 5	50.0	50.1	0.50	0.7712	-244.5	-1.1438	0.0	1.0	0.0	0.0
Design 6	50.0	50.1	0.25	0.7707	-244.9	-1.1453	0.0	0.0	1.0	0.0
Design 7	76.0	66.8	0.25	1.0360	-101.5	-0.5565	0.0	0.0	0.0	1.0
Design 8	50.85	50.1	0.25	0.8051	-215.4	-1.0598	0.25	0.25	0.25	0.25
Design 9	79.8	69.9	0.25	1.0390	-101.5	-0.5154	0.60	0.15	0.15	0.10

Table 6 demonstrates that there is a tendency for criticality to be out-weighed by the other three objectives. This is largely due to the fact that the latter three objectives are all optimized concur-

Table 7: Design for SOGA using Physical Programming

	Height	Smear	Pu %	k_{eff}	α_v	α_f
Design 10	79.80	54.65	54.75	1.0303	-110.023	-0.6926
Rankings	–	–	Desirable	Tolerable	Tolerable	Tolerable

rently and in opposition to k_{eff}. Figure 6 shows this idea more clearly, where the points in blue represent the Pareto front discovered using the weighting scheme. We notice a large swath of the Pareto front is centered below a k_{eff} of 1.0, indicating the influence of safety parameters on the overall fitness.

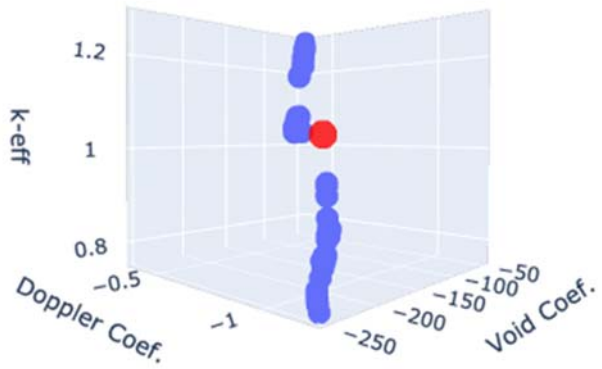


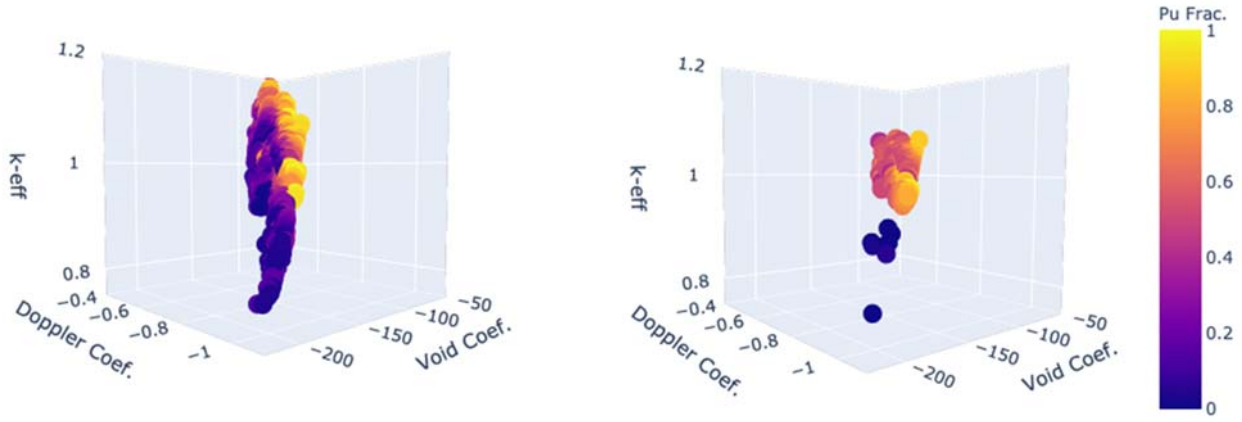
Figure 6: Pareto front for SOGA using a Weighting Scheme

Using physical programming can help reduce the dichotomy of solutions represented by Figure 6. The ranking structure from Table 2 was used to in conjunction with SOGA to perform a single optimization calculation, where the resulting design (Design 10) along with its objective function ranks are shown in Table 7. [Design 10 is the red dot in Figure 6.] Comparing the two designs with equal weighting (Design 8 & 10), Design 10 strikes a more reasonable compromise between the multiple objectives without building non-intuitive weighting sets. To obtain a solution similar to this using the weighting scheme, the skewed weighting scheme in Design 9 was required.

Using physical programming as a fitness function for SOGA produces a singular design rather than a Pareto front, due to the fact that weights are not used. If the solution does not fit the designer's needs, the ranking scheme found in Table 2 would need to be updated. Updating the ranking scheme appears to be similar to varying the weighting scheme; however, a designer is manipulating physical requirements for the core rather than abstract weights.

The improvement of design selection using physical programming as a fitness function for SOGA is promising, and the method is extended to MOGA. The MOGA utilizes a domination scheme

to determine Pareto optimal solutions, which should provide a clear Pareto front in comparison to Figure 6. The resulting Pareto front for the traditional and physical programming schemes can be seen in Figure 7a and 7b. The Pareto front produced by physical programming is much smaller in size and scope compared with the traditional fitness scheme. Physical programming selects a set of solutions which are more centered around $k_{eff} = 1.0$, and does not present many of the extremes seen in the traditional method. This greatly reduces the available designs, and presents more designs within the desired ranges.



(a) Pareto front for WS Fitness **(b) Pareto front for PP Fitness**

Figure 7: Pareto fronts for MOGA

Table 8 shows a selection of designs that were obtained by the MOGA for the traditional (T) and physical programming (PP) methods. There is a strong similarity for the 'optimal' height and smear for in both cases; however, the differentiating aspect is the plutonium content. The plutonium content drives the criticality objective in Design 12, and helps ensure that k_{eff} is greater than 1.0, as specified in Table 2. Design 11 successfully optimizes the safety coefficients and the plutonium content; however, it fails to deliver on a critical core. Table 2 also shows the ranking scheme that associated with each design, where Design 12 accepts multiple tolerable conditions to prevent an unacceptable objective. Due to the lack of physical programming, the traditional method has no way to determine that extremes (such as low Pu % at the cost of low k_{eff}) are detrimental to the design.

For both the SOGA and MOGA, additional constraints could have been placed on the core to create a better representation of the designer's choice when using the weighting scheme. For example, an inequality for k_{eff}, similar to the one used in Section 4.1, could have been used to ensure the algorithm produced a critical core. This further lends credibility to the fact that creating a fitness function using a weighting scheme can require significantly more effort, and often requires multiple iterations to perfect. Despite this drawback, a weighting function can prove useful if the designer has limited knowledge of the design space, and could be used for initial exploration.

In considering the computational requirements, we find that utilizing physical programming requires an extra computational step to calculate the fitness for each design selected by the optimization scheme. Table 9 shows the approximate time requirements and the number of fitness function

Table 8: Designs for MOGA using Weighting Scheme and Physical Programming

Design	Height	Smear	Pu %	k_{eff}	α_v	α_f
Design 11 - T	79.80	50.20	4.00	0.9105	-159.26	-0.8776
Ranking - T	–	–	Highly Desirable	Unacceptable	Desirable	Desirable
Design 12 - PP	79.94	53.70	57.25	1.0260	-111.86	-0.7066
Ranking - PP	–	–	Desirable	Tolerable	Tolerable	Tolerable

evaluations to convergence for both the SOGA and MOGA schemes using the traditional and physical programming methods. The SOGA timing is based on running a set of solutions to generate the Pareto front, where each individual simulation took between 50 - 150 seconds.

Table 9: Computational Efficiency for Physical Programming

	SOGA - T	SOGA - PP	MOGA - T	MOGA - PP
Time (s)	27,000	131	1,381	1,618
Convergence Rate	400 - 1,000	900	$>$10,000	$>$10,000

There is a large time discrepancy between the SOGA methods due to the fact that the traditional method required multiple simulations to create a Pareto front. The convergence rates for the two methods using SOGA are difficult to directly compare, as the traditional method requires multiple optimization calculations to be run. Further statistical analysis may provide insight into an average behavior for the SOGA convergence, however it was not explored here. Both MOGA methods returned the best solutions after 10,000 fitness function evaluations. This indicates that both methods likely converge at similar rates, and obtaining a full set of optimal solutions requires exhaustive searches of the design space.

Physical programming creates more precise results using both algorithms, if the designer's needs are adequately described in the ranking scheme. This process requires extensive knowledge of both the design and objective space, which may not be applicable for every engineering type problem. Special care must be taken in assigning the ranking scheme, as a scheme which over-estimates the objective space can fail to yield an adequate solution. Conversely, a scheme which under-estimates the objective space will evaluate all solutions as equally probable. Physical programming provides a reduction in the initial problem setup, computational savings, and the solutions presented for this problem.

5. CONCLUSIONS

Utilizing a database as a ROM provides core designers with a tool to rapidly iterate on a design, given a set of defined constraints. This can reduce the amount of time required to converge on an appropriate reactor design, while continually increasing the amount of information that is available via the database. Employing a multi-objective optimization algorithm provides core designers with a Pareto set of solutions to examine optimal configurations and understand the consequences associated with using various core geometries or materials. For this study, a small design space for sodium fast reactors was examined, where optimal configurations were found which maximized k_{eff}, while simultaneously minimizing the void and Doppler coefficients.

For the sodium fast reactor design space selected, 36 data-points are not sufficient to represent the underlying physics with a ROM. The addition of 24 data points to the ROM provides an increase in the accuracy of the model, which is promising, as increasing the quality of the database will help provide higher quality solutions. This provides some confidence that performing an initial analysis of the design space can provide long-term computational savings when determining what areas of the ROM require additional data. Adding to the database at large is time-consuming, and thus it is important to determine what level of accuracy is required for the optimization. This could be further supplemented by reducing the design space once an area of interest is found with the initial optimization. A smaller set of high-fidelity simulations could be performed in this reduced space to glean additional information.

Physical programming was utilized as a fitness function for both a single objective and multi-objective genetic algorithm to help determine optimal core configurations. In this approach, the designer's goals are transformed into a ranking system for each objective function to find optimal solutions. For the SOGA, utilizing physical programming resulted in a single optimal point which eliminates the Pareto front associated with the weighting scheme. Utilizing physical programming in a MOGA produces a Pareto front within a more confined space than a traditional MOGA algorithm, which more appropriately represents the designer's goals. Physical programming is also easier to manipulate than a traditional weighting scheme, if the goals of the design change. This scheme is not exclusive to reactor core design and could easily be applied to other areas of optimization in nuclear engineering. Care must be taken when creating the ranking scheme to prevent the optimization from finding to many/few solutions.

ACKNOWLEDGEMENTS

We would like to thank Gilles Youinou at Idaho National Laboratory for his help defining and explaining our design basis, and Nick Touran for his physical programming package.

REFERENCES

[1] J. Rhodes, K. Smith, and D. Lee, "CASMO-5 Development and Applications," *PHYSOR-2006*, Vancouver, Canada, 2006.

[2] O. Fabbris et al, "Surrogates Based Multi-Criteria Predesign Methodology of Sodium-Cooled Fast Reactor Cores - Application to CFV-like cores," *Nuclear Engineering and Design*, vol. 305, pp. 314-333, 2016.

[3] G. Reynoso-Meza, et al, "Physical programming for preference driven evolutionary multi-objective optimization," *Applied Soft Computing*, 24, pp. 341-362, 2014.

[4] J. Andersson, "A survey of multiobjective optimization in engineering design," Tech. Rep. LiTH-IKP-R-1097, Department of Mechanical Engineering, Linkoping University, 2000.

[5] D.E. Goldberg. 1989. "Genetic Algorithms in Search, Optimization and Machine Learning," 1st. ed. Addison-Wesley Longman Publishing Co., Inc., USA.

[6] "Nuclear Metal Fuel: Characteristics, Design, Manufacturing, Testing, and Operating History," White Paper 18-01, United States Nuclear Regulatory Commission, June 2018.

[7] B.M. Adams, et al, "Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.10 User's Manual," Sandia Technical Report SAND2014-4253, May 2019.

[8] A. Messac, "Physical programming-effective optimization for computational design." AIAA journal 34.1 (1996): 149-158.

[9] C.J. Werner, et al., "MCNP6.2 Release Notes", Los Alamos National Laboratory, report LA-UR-18-20808 (2018).

[10] R. Stewart, "FRIDGe: Fast Reactor Input Deck Generator," *Journal of Open Source Software*, 4(40), 1486, https://doi.org/10.21105/joss.01486.

[11] M. B. Chadwick, M. Herman, P. Oblozinsky, et al, "ENDF/B-VII.1 nuclear data for science and technology: Cross sections, covariances, fission product yields and decay data", Nuclear Data Sheets, 112(12):2887-2996 (2011).

[12] The HDF Group. "Hierarchical Data Format, version 5," http://www.hdfgroup.org/HDF5/ (1997).

[13] R. Stewart, (2019). Sodium Fast Reactor Database (Version v0.1) [Data set]. Zenodo. http://doi.org/10.5281/zenodo.3464101

[14] W. McKinney. "Data Structures for Statistical Computing in Python", *Proceedings of the 9th Python in Science Conference*, 51-56 (2010).

[15] E. Jones, E. Oliphant, P. Peterson, et al, "SciPy: Open Source Scientific Tools for Python", http://www.scipy.org/ (2001).

[16] N. Touran, "physprog", https://github.com/partofthething/physprog (2017).

[17] R. Stewart and T.S. Palmer, "Geometric Design Space for Sodium Fast Reactors," *Transactions of the American Nuclear Society*, Washington, D.C., November 17-21, Vol. 121, (2019).

[18] R. Stewart and T.S. Palmer, "Metallic Fuel Design Space for Sodium Fast Reactors," *Transactions of the American Nuclear Society*, Orlando, Florida, November 11-15, Vol. 119, (2018).