INL/RPT-22-68865

Reduced Order Models Generation for HTGRs Pebble Shuffling Procedure Optimization Studies

September 2022

M3RD-22IN0501012

Zachary M. Prince[1], Colin Brennan[2], Mehmet Turkmen[1], Paolo Balestra[1], and Gerhard Strydom[1]

[1]*Idaho National Laboratory*
[2]*University of Texas – Austin*

Idaho National Laboratory

INL is a U.S. Department of Energy National Laboratory operated by Battelle Energy Alliance, LLC

INL/RPT-22-68865

Reduced Order Models Generation for HTGRs Pebble Shuffling Procedure Optimization Studies

M3RD-22IN0501012

Zachary M. Prince[1], Colin Brennan[2], Mehmet Turkmen[1], Paolo Balestra[1], and Gerhard Strydom[1]

[1]**Idaho National Laboratory**
[2]**University of Texas – Austin**

September 2022

Idaho National Laboratory
Advanced Reactor Technologies
Idaho Falls, Idaho 83415

http://www.art.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Page intentionally left blank

ART Program

Reduced Order Models Generation for HTGRs Pebble Shuffling Procedure Optimization Studies

INL/RPT-22-68865

September 2022

Technical Reviewer: (Confirmation of mathematical accuracy, and correctness of data and appropriateness of assumptions.)

Joshua T. Hanophy Nuclear/Reactor Engineer	Date

Approved by:

Michael E. Davenport ART Project Manager	Date
Travis R. Mitchell ART Program Manager	Date
Michelle T. Sharp INL Quality Assurance	Date
Name Title	Date

SUMMARY

This report provides an initial study for producing reduced-order models (ROMs) of pebble-bed high-temperature gas-cooled reactor (HTGR) models for the purposes of design optimization. As an initial study, this work is meant to be exploratory—identifying useful workflows and methods for ROM generation and not meant to be a catch-all analysis of HTGR ROM generation and usage for optimization. This report summarizes three tasks performed in Fiscal Year 2022: 1) the creation of HTGR model, 2) the sensitivity analysis of model design parameters, and 3) an introduction to ROM generation techniques. The representative HTGR model created in this work is a multiphysics equilibrium-core using the BlueCRAB (comprehensive reactor analysis bundle) reactor analysis application, coupling four physical phenomena: neutronics, streamline depletion, porous flow thermal hydraulics, and pebble heat conduction. Part of the model creation was identifying some design parameters and quantities of interest that are relevant in an optimization analysis and adjustable in the model. The sensitivity analysis utilized a polynomial chaos expansion methodology to compute global sensitivity metrics. This analysis showed that thermal hydraulics parameters and quantities of interest had a relatively small impact on the equilibrium core, compared to parameters such as shuffling velocity or discharge burnup. Finally, the ROM generation work involved exploring three different ROM methodologies: polynomial regression, a Gaussian process, and artificial neural networks. Using a cross-validation technique to characterize ROM performance, the Gaussian process and single-layer artificial neural networks showed the most promising results. Overall, this study was insightful and the lessons learned will be invaluable for the eventual development of an HTGR design optimization workflow.

Page intentionally left blank

CONTENTS

SUMMARY iv

1 Introduction 1

2 Equilibrium Core Model 2

2.1 Nominal Reactor Parameters 2

2.2 Reference Calculations 3

3 Sensitivity Analysis 7

3.1 Design Space and Quantities of Interest 7

3.2 Global Sensitivity Methodology 8

3.3 Sensitivity Results 10

4 Reduced-Order Modeling 14

4.1 Methodology 14

4.2 Capabilities in MOOSE Stochastic Tools Module 16

4.2.1 K-Fold Cross-Validation 17
4.2.2 Polynomial Regression 19
4.2.3 Gaussian Process 20
4.2.4 Artificial Neural Network 22
4.3 Initial Reduced-Order Model Study 25

5 Conclusions 28

REFERENCES 29

FIGURES

Figure 1. MultiApp and Transfer structure of representative HTGR model. 3
Figure 2. Geometry of representative HTGR model. 4
Figure 3. Core-wide neutronics and thermal-hydraulics field quantities for representative HTGR with nominal parameters. 6
Figure 4. Points for two-dimensional, seventh-order quadratures. 10
Figure 5. Probability density of quantities of interest (QoIs) from polynomial chaos expansion (PCE) sampling. 13
Figure 6. Total Sobol indices for each parameter and QoI calculated from PCE surrogate. 14
Figure 7. Reduced-order model generation process 16
Figure 8. The architecture of the simple feed-forward artificial neural network (ANN) in the MOOSE stochastic tools module (MOOSE-STM). 22
Figure 9. % Relative root mean-square error (RMSE) for several candidate single-layer neural networks. It is apparent that increasing the neuron count from 64 to 128 causes overfitting for some of the responses. 26
Figure 10. % Relative RMSE for several low-width, two-layer neural networks. In all cases, even parsimonious two-layer networks are outperformed by the best single layer network (ANN-64). 27
Figure 11. % Relative RMSE for several low-width, three-layer neural networks. In all cases, even parsimonious three-layer networks are outperformed by the best single layer network (ANN-64). 27
Figure 12. % Relative RMSE for several candidate models. 28

TABLES

Table 1. Nominal reactor parameters of representative HTGR model. 4
Table 2. Design space for HTGR initial sensitivity and ROM study. 8
Table 3. Output space for HTGR initial sensitivity and ROM study. 8
Table 4. Number of points with tensor product and sparse quadratures for various dimensions and quadrature orders. 9
Table 5. Mean and standard deviation calculated from the PCE surrogate of a representative HTGR model 12
Table 6. Currently available ROMs in the MOOSE-STM and their supported data types. 17
Table 7. Learning parameters used for neural networks. 23
Table 8. Neural network configurations considered in first search. 24
Table 9. Candidate neurons per layer used to define grid-search for two- and three-layer models. 24
Table 10. Five-fold RMSE for k_{eff} in pcm. 26

ACRONYMS

ANN	artificial neural network
FOM	full-order model
GP	Gaussian process
GPBR200	200MW General Pebble Bed Reactor
HTGR	high-temperature gas-cooled reactor
MOOSE	Multiphysics Object-Oriented Simulation Environment
MOOSE-STM	MOOSE stochastic tools module
NEAMS	Nuclear Energy Advanced Modeling and Simulation
PCE	polynomial chaos expansion
PR	polynomial regression
QoI	quantity of interest
RMSE	root mean-square error
ROM	reduced-order model
RPV	reactor pressure vessel
TH	thermal hydraulics
TRISO	tristructural isotropic

Page intentionally left blank

1. Introduction

This work addresses the challenges involved with the optimization of the equilibrium core in a pebble-bed high-temperature gas-cooled reactor (HTGR). This optimization is an incredibly intensive task, given the high-dimensionality of the parameter space (number of pebble types, enrichment, recirculation policies, equilibrium conditions, etc.) and the numerous constraints (max temperature, minimum power, pebble burnup, etc.). Optimizing directly on the multiphysics model would be considered intractable because of the shear amount of simulations that are needed. As such, the generation of reduced-order models (ROMs) is a highly desired capability. The idea is to take a limited set of simulation results of a multiphysics full-order model (FOM) and train a lower-fidelity ROM that is both much faster to evaluate and numerically easier to optimize.

This report explores the HTGR model design space and gains insight into ROM production. There are three goals for this initial study:

1. Develop a robust and representative HTGR model and define design parameters and quantities of interest (QoIs)
2. Quantify the impact of perturbing parameters on QoIs of the HTGR model
3. Train various types of ROMs using data from the HTGR model and characterize their performance.

The first goal involves using Multiphysics Object-Oriented Simulation Environment (MOOSE)-based modeling and simulation applications [6], namely the comprehensive reactor analysis bundle (BlueCRAB) application, which includes neutron transport and pebble-bed streamline depletion from Griffin [5], thermal hydraulics (TH) from Pronghorn [10], and MOOSE heat conduction module. Since BlueCRAB is built on MOOSE, it includes capabilities from the MOOSE stochastic tools module (MOOSE-STM) [13]. The MOOSE-STM is used in this work to perform the second and third goals, particularly global sensitivity analysis and ROM generation.

This report is organized as follows:

- Section 2 describes the representative HTGR model built using BlueCrab along with the defined design space and QoIs

- Section 3 presents the sensitivity study performed in this work
- Section 4 explores ROM generation, which characterizes various methods
- Section 5 summarizes the report and details avenues of future work.

2. Equilibrium Core Model

The design selected for this study is a 200MW General Pebble Bed Reactor (GPBR200), built based on open source literature of current and past designs, such as the HTR-PM, Xe-100, HTR-Modul, PBMR400, and HTTR [1, 2, 8, 14]. The model was developed thanks to a collaboration with the Nuclear Energy Advanced Modeling and Simulation (NEAMS) program with the scope to test the new equilibrium core calculation module developed in Griffin.

The model is multiphysics with four different phenomena, including:

1. Criticality using neutron kinetics (Griffin),
2. Streamline depletion (Griffin),
3. Core-wide—pebble bed, cavity, reflector, gaps, barrel, reactor pressure vessel (RPV) —thermal hydraulics (Pronghorn),
4. Pebble and tristructural isotropic (TRISO) kernel heat conduction (MOOSE heat conduction module).

These physics are coupled using the MOOSE MultiApp system [3]. The MultiApp structure and transfer mechanisms are shown in Figure 1.

2.1 Nominal Reactor Parameters

Table 1 lists the some of the nominal reactor properties. These properties form the basis of the reference design, and some of them will be perturbed later in the sensitivity study and ROM generation. Based on the reference design features in Table 1, Figure 2 shows the 2D cylindrical (RZ) geometry of the HTGR model along with the mesh used for the neutronics and porous media TH. As seen from the figure, the design comprises upper reflector, cavity, barrel, RPV, (pebble bed) core region, and discharging chute. Pebbles are loaded through the cavity, move downward along

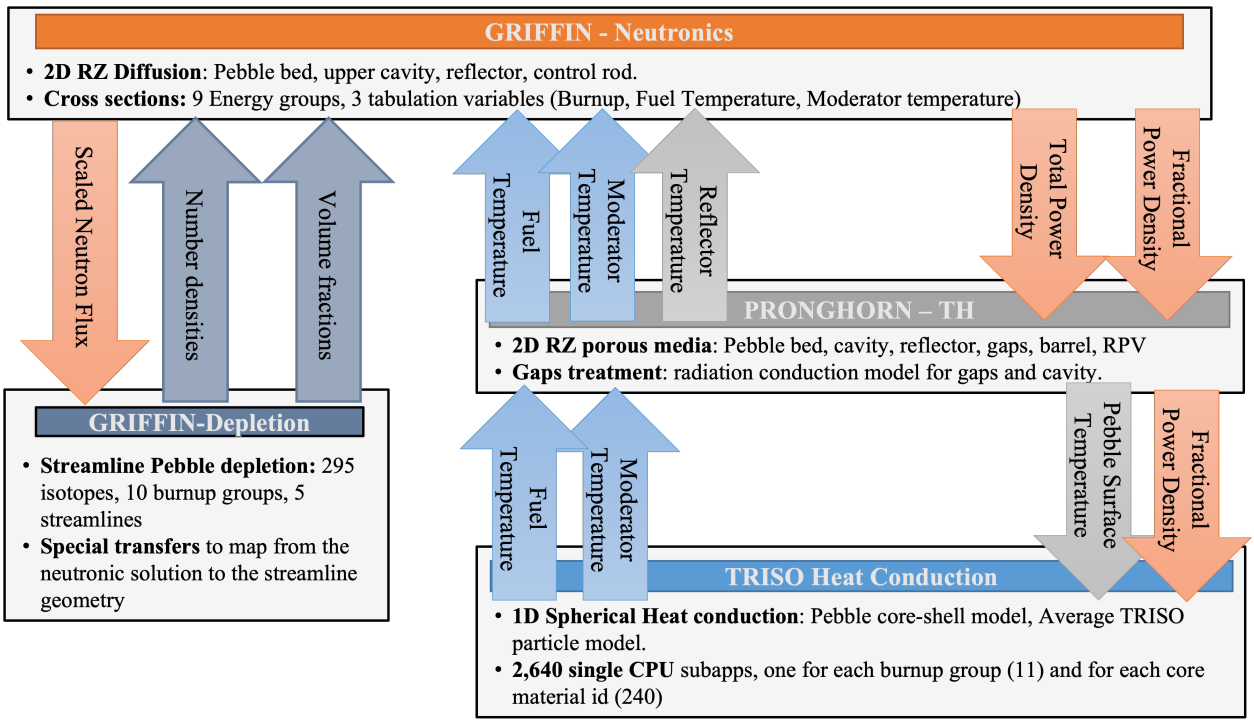


Figure 1: MultiApp and Transfer structure of representative HTGR model.

streamlines, and are taken out from the discharging chute at the end of cycle. For the nominal case, pebbles are unloaded at a rate of one pebble per minute, with pebbles reaching a burnup of 147,600 MWd/t_{HM} removed from circulation.

2.2 Reference Calculations

Based on the nominal parameters, Figure 3 shows the resulting spatial profiles of some of the field quantities in the model, such as power density, $^{235}U/^{135}Xe$ density distribution, normalized thermal and fast flux, solid surface temperature, coolant temperature, and coolant pressure. The resulting multiplication factor (k_{eff}) for this model is 1.01344. We see that the ^{235}U density is at a maximum at the top of the core because fresh pebbles are inserted at the top. Consequently, the maximum power density is observed near the top but at a slightly lower position. This is because, although the ^{235}U density is maximum at the top, the thermal neutron leakage is a prominent factor in decreasing the number of fission reaction at the periphery of the core. The power density distribution has the same distribution as the fast flux, which has a direct relationship with the number of fission reactions, thus power density. In the case of the distribution of ^{135}Xe density,

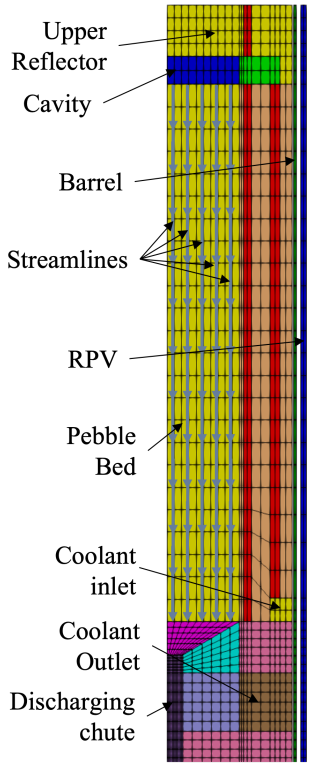


Figure 2: Geometry of representative HTGR model.

Table 1: Nominal reactor parameters of representative HTGR model.

Parameter	Value	Units
Power	200	MW
Core height	1.20	m
TRISO kernels per pebble	18687	—
Fuel enrichment	15.5%	—
Coolant (He) inlet temperature	533	K
Coolant (He) outlet pressure	5.8	MPa
Burnup limit	147,600	MW-days/t_{HM}
Pebble unloading rate	1	pebbles/min

maximum ^{135}Xe density, is shifted downward respect to the power peak for a combination of effects such as pebble moving downward, reduction of the destruction term (proportional to thermal flux), time needed by the ^{135}Xe and its precursor ^{135}I to decay. From the thermal hydraulics and heat conduction, we see a max pebble surface temperature of 1065 K, a similar max fluid temperature of 1060 K, and a total pressure drop of 226 kPa.

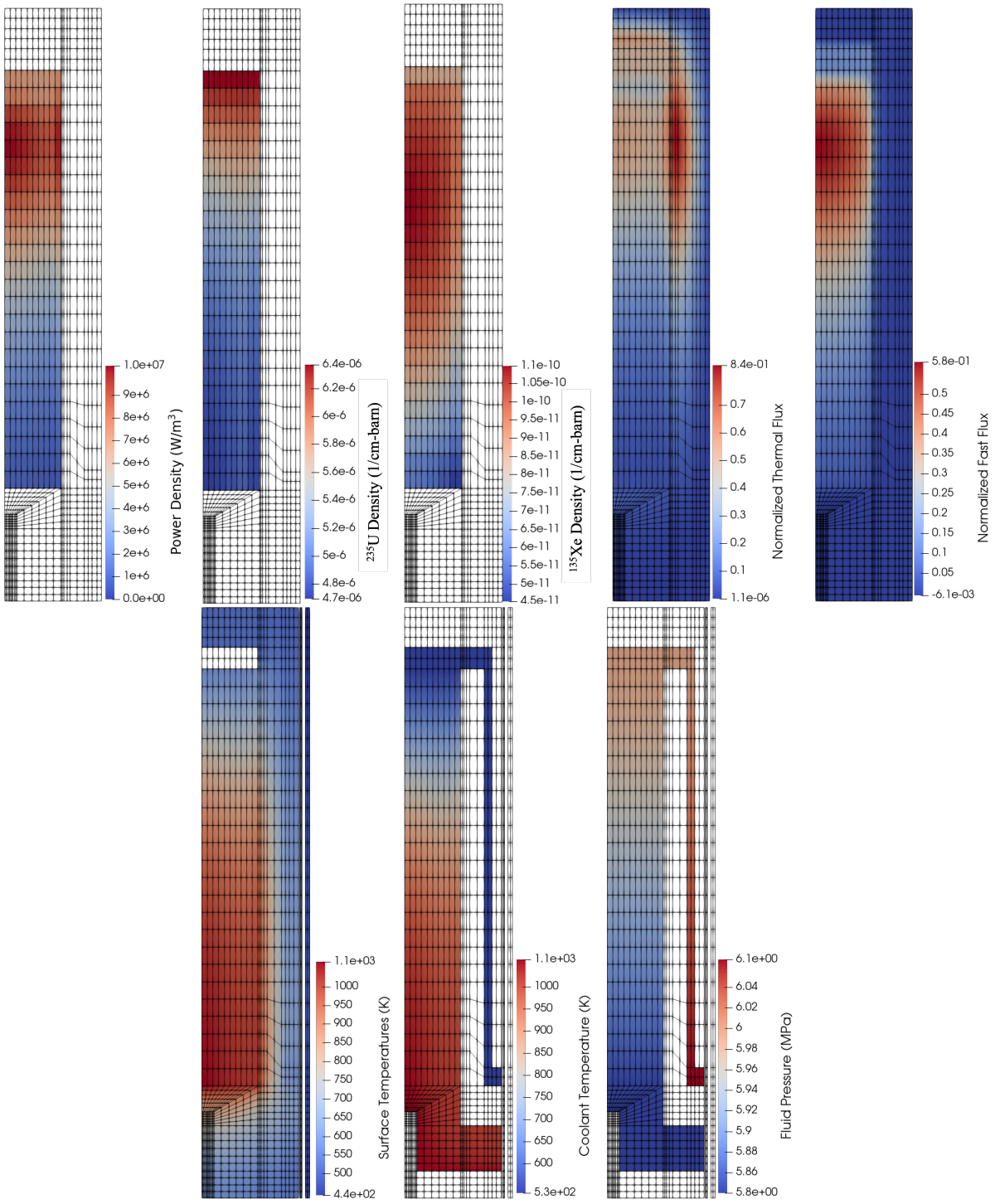


Figure 3: Core-wide neutronics and thermal-hydraulics field quantities for representative HTGR with nominal parameters.

3. Sensitivity Analysis

This section presents the initial sensitivity study performed for the representative HTGR model. Sensitivity analysis, by definition, searches for the variables that have a significant effect on the analysis outcome and quantifies the effect of perturbed parameters in a model. A parameter with a small sensitivity metric on a QoI typically means that the QoI is not significantly affected by the perturbation of the parameter. Higher-order sensitivity metrics can also quantify how the parameter affects the QoI, for example, if it is a nonlinear dependence, covariance with other parameters, etc. Sensitivity metrics can be classified into two categories: local and global. Local sensitivity is essentially the derivative of the QoI with respect to the parameter at a certain location in the parameter space. Local sensitivity can be useful for cases where the design space has been localized to relatively small domain in order to gain insight on the parameterized system in that region. This work will focus on global sensitivity analysis, which computes metrics that give a sense of the effect of parameters for the entire parameter space. In the context of this work, global sensitivity is useful for filtering parameters from the design space. Since the ROM accuracy and performance suffer heavily from large parameter spaces, removing less impactful parameters from ROM generation can help significantly.

The rest of the section will describe the design space and QoIs in the equilibrium core model, global sensitivity methodology used in the MOOSE-STM, and results from the initial study on the representative HTGR model.

3.1 Design Space and Quantities of Interest

Before performing any type of sensitivity study or ROM generation, the design, uncertainty, and output space must be defined. The design and uncertainty space involves model parameters that are either part of a design decision or parameters that are not known exactly. These parameters are defined with a probability distribution describing the likelihood of a certain value occurring. Design parameters are typically defined as uniform distribution, whereby the likelihood of a value is equal between two bounds. There are many different types of distributions for uncertain parameters, a common choice is a normal distribution specified by a mean value with a standard deviation. For this study, we will only focus on design parameters. Table 2

lists the parameters studied in this work with their specified bounds. The output space involves specifying QoIs, which are typically global outputs from the simulations that are relevant to analysis. In the context of optimization, these quantities will be used as part of the function defining the design performance. Table 3 lists the QoIs studied in this work.

Table 2: Design space for HTGR initial sensitivity and ROM study.

Parameter	Symbol	Lower Bound	Upper Bound	Units
Pebble unloading rate	v_p	0.5	2.0	pebbles/min
Burnup limit	$Bu^{\max}$	131,200	164,000	MW-days/t_{HM}
Mass flow rate	$\dot{m}$	74.7	82.5	kg/s
Coolant inlet temperature	T_{in}	506.6	559.9	K
Coolant outlet pressure	P_{out}	5.55	6.13	MPa
Reactor power	P	180	220	MW

Table 3: Output space for HTGR initial sensitivity and ROM study.

QoI	Symbol	Units
Multiplication factor	k_{eff}	—
Max pebble power	$P_k^{\max}$	W
Max TRISO kernel temperature	$T_k^{\max}$	K
Max moderator temperature	$T_m^{\max}$	K
Max reflector temperature	$T_{\text{ref}}^{\max}$	K
Power peaking factor	f_P	—
Axial power peaking factor	f_P^z	—
Radial power peaking factor	f_P^r	—
^{135}Xe peaking factor	f_{Xe}	—
Axial ^{135}Xe peaking factor	f_{Xe}^z	—
Radial ^{135}Xe peaking factor	f_{Xe}^r	—
Fuel max fast flux	$\phi_{1,k}^{\max}$	1/cm^2-s
Reflector max fast flux	$\phi_{1,\text{ref}}^{\max}$	1/cm^2-s
Fuel average fast flux	$\bar{\phi}_{1,k}$	1/cm^2-s
Reflector average fast flux	$\bar{\phi}_{1,\text{ref}}$	1/cm^2-s

3.2 Global Sensitivity Methodology

The MOOSE-STM includes several different methods for global sensitivity analysis: Sobol sampling [12], Morris screening [7], and polynomial chaos expansion (PCE) [15]. PCE is chosen for this work since the method is robust and efficient for relatively small parameter spaces, as is the case with this study. It might prove advantageous in later studies with the more parameters to use Morris screening as it is much more efficient in high-dimensional spaces.

PCE is a type of surrogate modeling technique where a QoI that is dependent on input parameters is expanded as a sum of orthogonal polynomials. Given a QoI Q dependent on a set of parameters $\vec{\xi}$, the PCE expansion is:

$$Q(\vec{\xi}) \approx \hat{Q}(\vec{\xi}) = \sum_{i=0}^{P} q_i \Phi_i(\vec{\xi}), \tag{1}$$

where P is the multidimensional polynomial order and q_i are coefficients that are to be computed. Φ_i are the multidimensional polynomials, which are a monomial product of one-dimensional polynomials based on the parameters' probability distributions:

$$\Phi_i(\vec{\xi}) = \prod_{d=1}^{D} \phi_i^d(\xi_d), \quad i = 0, ..., P, \tag{2}$$

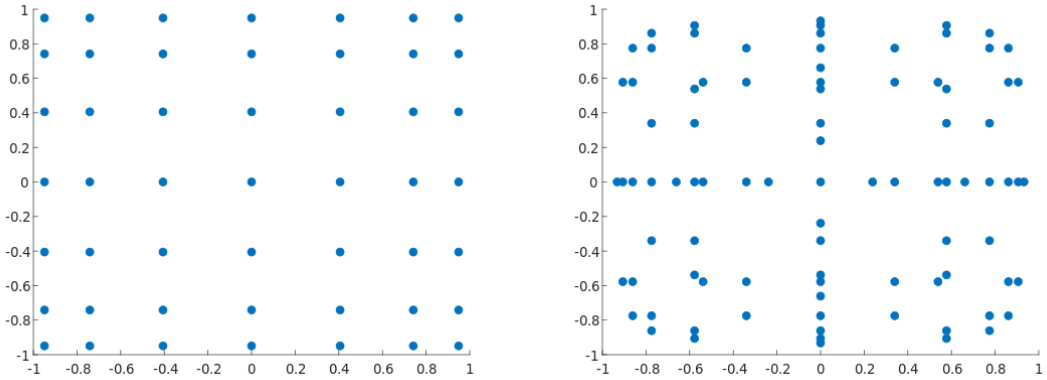
where ϕ is the one-dimensional polynomial and D is the number of parameters. Since the polynomial basis is orthogonal, a nonintrusive technique is developed where the coefficients are found by performing a Galerkin projection and weighted integration. This integration is performed by sampling the FOM to obtain a training data set with N samples $\mathbf{Q} \equiv [Q(\vec{\xi}_1, ..., Q(\vec{\xi}_N)]$. To perform this integration efficiently, a multidimensional quadrature is produced with corresponding weights and points. The coefficients are then evaluated as:

$$q_i = \sum_{n=1}^{N} w_n Q(\vec{\xi}_n) \prod_{d=1}^{D} \phi_i^d(\xi_{d,n}), \quad i = 0, ..., P. \tag{3}$$

Since the multidimensional basis functions are a monomial product of one-dimensional functions, performing a naive tensor product of one-dimensional quadratures is wasteful, even with modest dimensionality. Thus we utilize a Smolyak sparse quadrature to obtain the points and weights [4]. The difference between a tensor product and sparse quadrature is shown in Figure 4 and Table 4.

Table 4: Number of points with tensor product and sparse quadratures for various dimensions and quadrature orders.

Quadrature	Tensor product			Smolyak sparse		
Order	$D = 2$	$D = 5$	$D = 8$	$D = 2$	$D = 5$	$D = 8$
2	5	11	17	5	11	17
3	9	243	6,561	14	66	153
5	25	3,125	390,625	55	1,001	4,845
7	49	16,807	5,764,801	140	7,997	74613



(a) Tensor product quadrature (b) Smolyak sparse quadrature

Figure 4: Points for two-dimensional, seventh-order quadratures.

Because the basis functions (Φ_i) are orthogonal, the expansion in Equation (1) has several convenient properties including the ability to compute statistical moments, like mean and standard deviation, and Sobol indices analytically:

$$\text{Mean:} \quad \mu = q_0, \tag{4a}$$

$$\text{Standard deviation:} \quad \sigma = \sqrt{\sum_{i=1}^{P} q_i^2 \prod_{d=1}^{D} \left\langle \phi_i^d, \phi_i^d \right\rangle}, \tag{4b}$$

$$\text{Total Sobol index:} \quad S_{\mathrm{T},d} = \sum_{i=0}^{P} q_i^2 \left\langle \phi_i^d, \phi_i^d \right\rangle, \quad d = 1, ..., D, \tag{4c}$$

where the $\langle \cdot, \cdot \rangle$ notation is the inner product of the functions:

$$\langle a(\xi), b(\xi) \rangle \equiv \int_{-\infty}^{\infty} a(\xi) b(\xi) f(\xi) d\xi, \tag{5}$$

where $f(\xi)$ is the probability distribution specific to the dimension and parameter.

3.3 Sensitivity Results

To train the PCE surrogate for the representative HTGR model, a polynomial order of 4 was chosen, which is a order high enough to prevent underfitting while still tractable to compute. Using a Smolyak sparse quadrature of order 4 and six parameters, this training required a total of 1,820 model evaluations. Due to the change in the design space parameters defined in Table 2, the

resulting mean and standard deviation from the surrogate for each QoI are shown in Table 5. To further clarify this, Figure 5 visualises the distribution of each QoI and sampled the PCE surrogate with 100,000 random points to obtain probability distributions.

From the results, the least-affected QoI by the parameters' change is f_p^r with a variation of 0.53% while the most-affected QoI is $\phi_{1,k}^{\max}$ with a variation of 15.97%. In other words, $\phi_{1,k}^{\max}$ is more sensitive to change in design space parameters than any other QoI, followed by $P_k^{\max}$, $\phi_{1,ref}^{\max}$, and f_p.

An outlook to each QoI indicates that k_{eff} varies around 1.0190 with a standard deviation of about 2,500 pcm. k_{eff} has a relatively flat probability, meaning values between 0.994 and 1.050 have a seemingly equal likelihood of occurring. The distribution of designs on k_{eff} (upper left) does not resemble any well-known distribution like Normal. While some designs produce a sub-critical core, which needs to be disregarded from the reactor designing process, a great deal of designs achieve higher excess reactivity (up to 1.08) compared to the reference design, recalling that the representative equilibrium core model has a k_{eff} value of 1.01344. This implies that there are noteworthy designs that need to be taken into account depending on the operational requirements.

As for the temperature quantities ($T_x^{\max}$), deviation from the mean value is very limited ($\sim$3.5%) compared to the other QoIs. The temperature distributions have a Gaussian-like shape, meaning they have a linear dependence between two or more parameters, which makes generating a ROM easier for these quantities.

The magnitude of variation in ϕ values depends on whether the flux is maximum or average but seems independent from whether the flux is computed in the fuel or reflector. This is because average flux is calculated over the entire volume of the region. This mitigates both increasing and decreasing effects due to changes in design parameters. On the other hand, maximum flux is taken at a particular location where the flux in a specified region reaches its peak value.

f_{Xe}^z is more sensitive to any change in design parameters than f_{Xe}^r. The reason is that the change of Xe concentration in z-direction is more acute than that in the r-direction, as can be clearly seen from Xe density distribution in Figure 3. f_p is related to the multiplication of f_{Xe}^z and f_{Xe}^r with corresponding propagated standard deviation. Likewise, f_p^z, f_p^r, and thus f_p can be associated with the power density distribution, illustrated in Figure 3. The pebble power has very little

chance of reaching over 4.5 kW. The power and Xe offsets are highly skewed, which indicates a lower limit for their potential value but no upper limit.

Table 5: Mean and standard deviation calculated from the PCE surrogate of a representative HTGR model

QoI	μ	σ **(%)**
k_{eff}	1.019	2.494 ($\sim$ 2,500 pcm)
$P_k^{\max}$	3,076	14.98
$T_k^{\max}$	1,080	3.467
$T_m^{\max}$	1,071	3.485
$T_{\text{ref}}^{\max}$	1,035	3.439
f_p	2.03	12.05
f_p^z	1.848	11.42
f_p^r	1.08	0.5335
f_{Xe}	1.263	6.314
f_{Xe}^z	1.18	5.753
f_{Xe}^r	1.073	1.032
$\phi_{1,k}^{\max}$	7.171e+12	15.97
$\phi_{1,ref}^{\max}$	3.055e+12	14.95
$\bar{\phi}_{1,k}$	3.077e+12	6.608
$\bar{\phi}_{1,\text{ref}}$	1.024e+11	7.15

Finally, Figure 6 shows the results of global sensitivity analysis from computing total Sobol indices from the PCE surrogate. In general, the burnup limit and pebble unloading rate have a significant impact on the QoIs, except temperature. Power has little impact on the dimensionless quantities like power peaking and Xe peaking factors. A perturbation on the TH parameters have a relatively small impact on the neutronics QoIs but are significant to the temperature-related QoIs. This analysis seems to indicate that we can reduce the parameter space by removing the parameterization of the TH parameters. However, the HTGR model is continuing to be improved and a transient model is planned in the future; therefore, this type of analysis expected to be repeated.

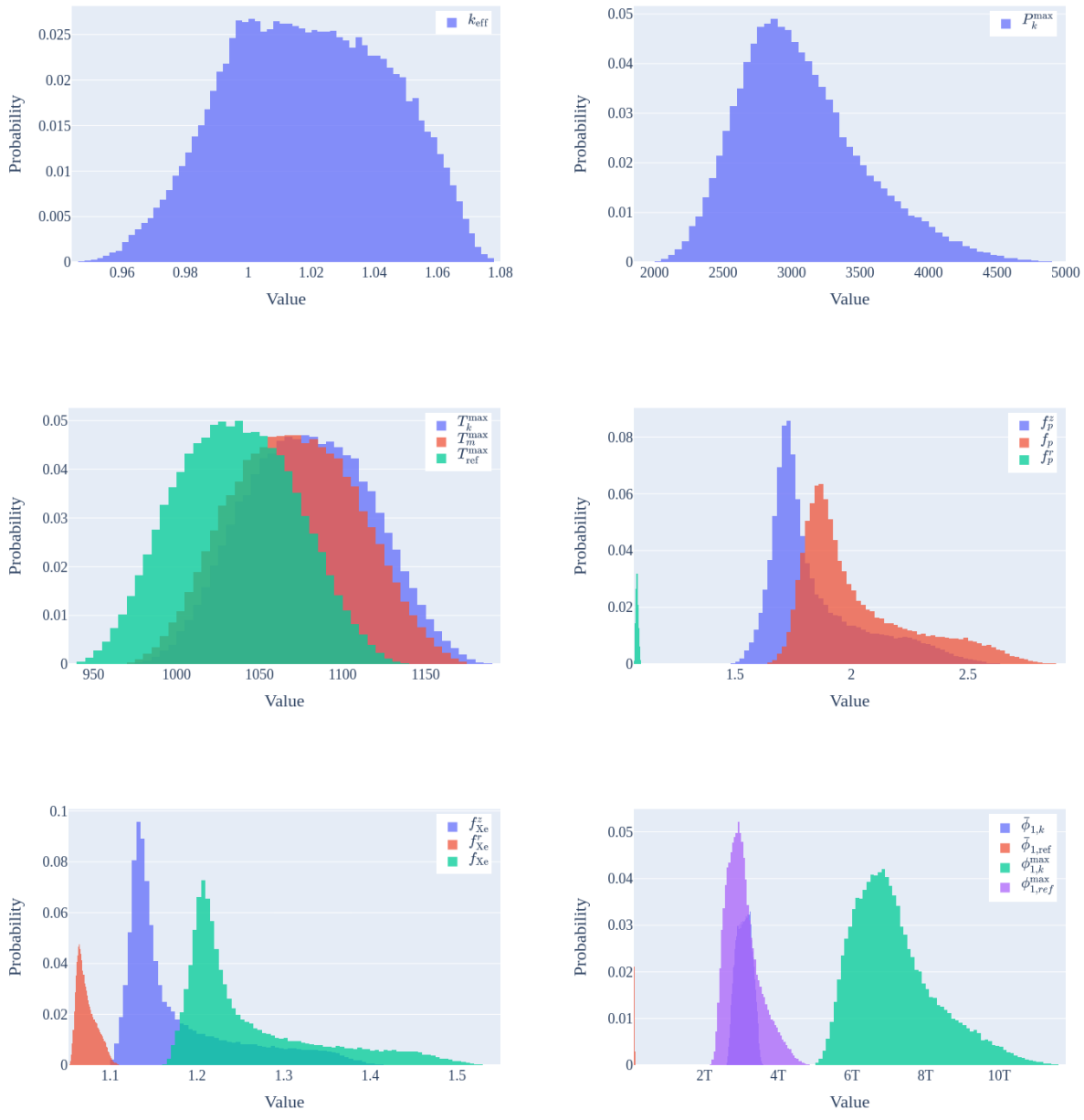


Figure 5: Probability density of QoIs from PCE sampling.

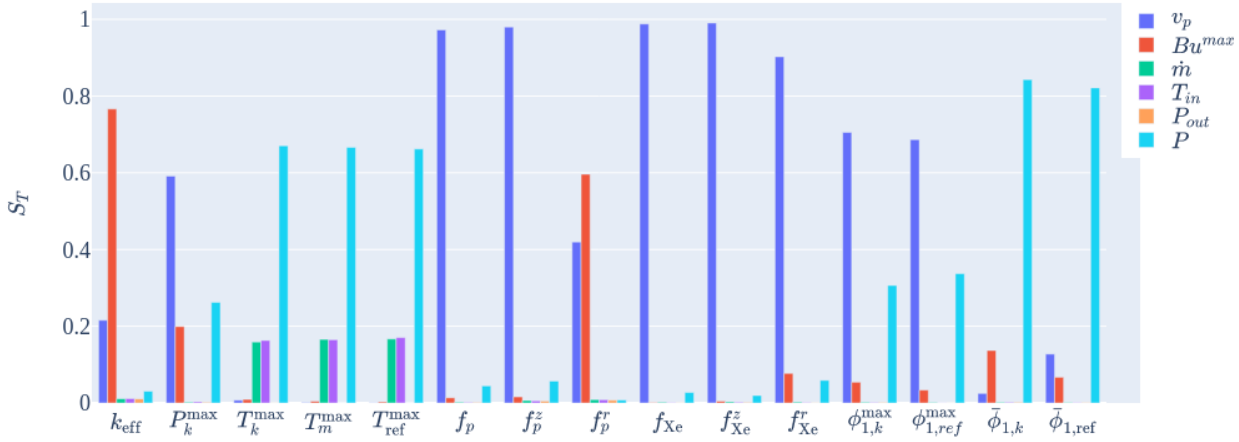


Figure 6: Total Sobol indices for each parameter and QoI calculated from PCE surrogate.

4. Reduced-Order Modeling

4.1 Methodology

The construction of many ROMs have the same basic procedure. First is to define the FOM, the parameters that are either uncertain or unknown, and the QoIs. These three entities can be described by the following highly simplified equation:

$$\vec{Q}(\vec{\xi}) = \mathcal{A}(\vec{\xi}), \tag{6}$$

where $\mathcal{A}$ is an operator representation of the FOM, $\vec{\xi}$ is the set of parameters, and $\vec{Q}$ is a set of QoIs. For clarity, a simple 1D one-group diffusion kinetics example is presented (basically a transient diffusion-reaction problem):

$$\frac{1}{v}\frac{\partial \phi}{\partial t} - \frac{\partial}{\partial x}\left(D\frac{\partial \phi}{\partial x}\right) + \Sigma\phi = q''', \tag{7a}$$

where

$$\phi(x, t=0) = \phi_0, \quad D\left.\frac{\partial \phi}{\partial x}\right|_{(x=0,t)} = q_0'', \quad \phi(x=L, t) = \phi_L. \tag{7b}$$

Equation (7) represents the FOM. The parameters ($\vec{\xi}$) can be defined as numerous things, including v, D, Σ, q''', L, ϕ_0, etc. The parameter space can also include spatial and temporal dependence by splitting the parameters into multiple parameters for different regions and time-ranges. However, including too many parameters causes a dimensionality increase in the overarching problem, complicating the task significantly. Therefore, it is important on the analysis side to determine which parameters are actually important (see Section 3). Finally, the QoIs ($\vec{Q}$) can also be a number of things, basically they should be whatever is important from the model evaluation. From the context of optimization, these are the quantities that are supposed to be optimized. For instance, it might be important to maximize average flux ($\bar{\phi}$) while minimizing maximum flux ($\phi_{\max}$). Performing an optimization or stochastic analysis at this point is possible and necessary; however, these tasks will mostly likely require many FOM ($\mathcal{A}$) evaluations, which may become intractable given the multiphysics nature of HTGR simulations. This is the impetus for creating a ROM.

Now that the FOM, parameters, and QoIs have been defined, the process of generating a ROM (or ROMs) can start. First is to define a "training" set of parameter points $\boldsymbol{\xi} = [\vec{\xi}_1, \vec{\xi}_2, ..., \vec{\xi}_N]^T$. There are numerous ways to define this set depending on the type of ROM, the number of parameters, constraints on the number of model evaluations, etc. There are also methods that adaptively choose parameter points ($\vec{\xi}_i$) during the generation process. Once the training set has been defined, the FOM is evaluated N times, which results in a QoI data set $\mathbf{Q} = [\vec{Q}_1, \vec{Q}_2, ...\vec{Q}_N]$. This data, along with the training set, is then used to generate the ROM ($\hat{\mathcal{A}}$), which obtains a modified version of Equation (6):

$$\hat{\vec{Q}}(\vec{\xi}) = \hat{\mathcal{A}}(\vec{\xi}) \tag{8}$$

The benefit of generating the ROM stem from the fact that N is much smaller than the number of FOM evaluations needed for the optimization procedure, and $\hat{\mathcal{A}}$ is much easier and faster to evaluate than $\mathcal{A}$. However, the extent of these benefits is largely correlated to how well the ROM emulates the FOM; we will call this the ROM error. A well formed ROM will be able to estimate this error so the analyst can determine if more training is needed to reduce it.

Finally, the generated ROM can be used for a variety of purposes, which is categorized below:

- Prognosis—Simply evaluating the ROM with a set of parameters to get an estimate of the

QoIs: $\hat{\vec{Q}} = \hat{\mathcal{A}}(\vec{\xi})$, which is useful for ROM validation and error estimation by comparing the ROM evaluation result with the FOM evaluation result.

- Diagnosis—Finding a set of parameters that best match a given set of QoIs: $\xi = \hat{\mathcal{A}}^{-1}(\vec{Q})$, which is useful if the the QoIs are known and a set of parameters is needed. This is less relevant in the context of optimization, but very important for digital twin development.

- Optimization—Finding a set of parameters that minimizes a form function dependent on the QoIs: $\vec{\xi} = \arg\max_{\vec{\xi}} f(\vec{Q}(\vec{\xi}))$.

Figure 7 summarizes the process described in this section.

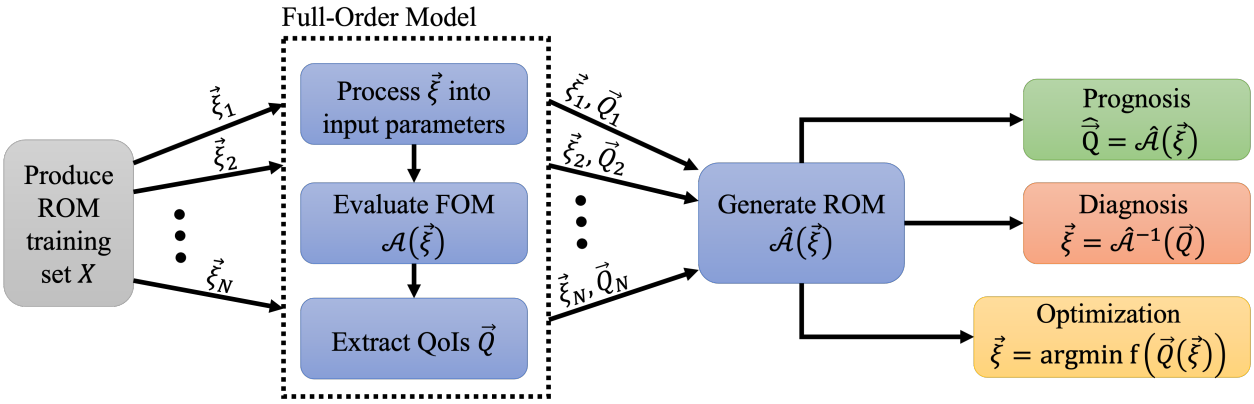


Figure 7: Reduced-order model generation process

4.2 Capabilities in MOOSE Stochastic Tools Module

This section gives an overview of the current ROM capabilities in the MOOSE-STM. The MOOSE-STM Surrogate system provides a means of training and evaluating ROMs. The system involves a two-step procedure with a different type of objects for each step: *Trainers* for using predictor and QoI data to generate model data (training) and *Surrogates* for using model data to build a functional representation between the parameters and QoI (evaluation). The training step usually involves sampling a MOOSE model using a stochastic `MultiApp` to generate the QoI. This data, along with the predictor data from the `Sampler`, is then used to generate the model data. The model data can then either be directly used by a `Surrogate` object or be outputted to a MOOSE readable file to be used later by the `Surrogate` object. The second step is simply to use

the `Surrogate` object as surrogate of the original MOOSE FOM. A common practice is to train a surrogate with a limited amount of samples, then use the surrogate to sample with many more perturbations for statistical analysis. However, `Surrogates` were designed to be pluggable with any other object tied to the MOOSE framework. The following sub-sections describe the currently available ROMs in MOOSE-STM. Table 6 gives an overview of these models. The following subsections will describe the techniques specifically used in this work, including polynomial regression (PR), Gaussian process (GP), and artificial neural networks (ANNs).

Table 6: Currently available ROMs in the MOOSE-STM and their supported data types.

Model Type	Scalar	Vector	Solution	Recommend
Nearest point	Y	N	N	N
PR	Y	Y	N	Y
PCE	Y	N	N	N
GP	Y	N	N	Y
ANN	Y	N	N	Y
Proper orthogonal decomposition	N	N	Y	N

4.2.1 K-Fold Cross-Validation

A primary concern in surrogate design is minimizing generalization error—that is, the error of the model in predicting QoIs for parameters not in the training set. k-fold cross-validation is a useful technique for estimating the generalization of candidate surrogate models using known data.

Consider the problem of training a surrogate model $\hat{Q}(\vec{\xi})$, where $\vec{\xi}$ is a vector of predictors and $\hat{Q}(\vec{\xi})$ is an approximation to a true response function $Q(\vec{\xi})$ to be determined by regression on a set of training data $\boldsymbol{R} = \{\vec{\xi}_i, Q_i\}_{i=1}^{N}$. k-fold cross-validation attempts to characterize the generalization error of $\hat{Q}(\vec{\xi})$ by splitting the data D into k random, non-overlapping training sets ($\boldsymbol{R}_{\text{train}}$) of size $\frac{k-1}{k}N$ and testing sets (D_{test}) of size $\frac{1}{k}N$. The surrogate is trained on $\boldsymbol{R}_{train}$ and used to make predictions for $\boldsymbol{R}_{\text{test}}$. True responses for $\boldsymbol{R}_{\text{test}}$ are known, so errors $e_i = Q_i - \hat{Q}_i$ can be calculated. The procedure is summarized in Algorithm 1.

In k-fold cross-validation, each data-point is used as a test case exactly once. The implementation in MOOSE-STM accumulates the root mean-square error (RMSE) across all splits to assign a score for the model, see Equation (9a). Commonly, we will report the relative

Algorithm 1 K-fold cross-validation simplified algorithm.

Randomly shuffle the data, then partition it into k splits.
for $j = 1, ..., k$ **do**
  Take the split as a test set: $\boldsymbol{R}_{\text{test}} \leftarrow \{R_i\}_{i=\frac{j-1}{k}N}^{\frac{j}{k}N}$.
  Train a regression model $\hat{Q}(\vec{\xi})$ using the other $k-1$ splits of data: $\boldsymbol{R}_{\text{train}} \leftarrow \boldsymbol{R} - \boldsymbol{R}_{\text{test}}$.
  Use the regression model to evaluate predictions for $\boldsymbol{R}_{\text{test}}$.
  Evaluate and retain prediction errors.
end for
Sum error contributions from each split.

RMSE, as in Equation (9b), which is the RMSE normalized by the mean response. This is useful for making comparisons of fit quality between response types.

$$\text{RMSE} = \sqrt{\sum_{i=1}^{N} \frac{\left(Q_i - \hat{Q}_i\right)^2}{N}} \tag{9a}$$

$$\text{RRMSE} = \frac{\text{RMSE}}{\frac{1}{N}\sum_{i=1}^{N} Q_i} \tag{9b}$$

Because the data is randomly partitioned, the result of a single trial of k-fold cross-validation can be an uncertain measure of model fitness. To average statistical noise, it is common to repeat cross-validation several times with a new random seed to obtain a mean RMSE. For all cases in this study, cross-validation was repeated several times (typically $N = 10$ trials), and reported values are an average.

For the purpose of this study, we will claim that *a model performs better than another model if a repeated cross-validation produces a lower RMSE*. Although not foolproof, RMSE scores from cross-validation can also help diagnose over- and under-fitting for a model type when varying hyper-parameters.

- If increasing the complexity of a model causes an increase in RMSE, it is likely that the new model is over-fit.
- If decreasing the complexity of the model causes an increase in RMSE, it is likely that the new model is under-fit.
- If repeated trials of cross-validation show high variance in RMSE, it is likely that the new model is over-fit.

4.2.2 Polynomial Regression

The PR ROM in the MOOSE-STM is essentially a multivariate linear regression reliant on ordinary least squares method (with the option to perform ridge regression). This ROM expresses the QoI dependence on a set of parameters with a linear expansion of multidimensional polynomials:

$$\hat{\vec{Q}}(\vec{\xi}) = \sum_{i=1}^{P} \vec{q}_i \Phi_i(\vec{\xi}), \tag{10}$$

where P is a user-defined polynomial order, $\vec{q}$ are scalar coefficients the size of the number of QoIs, and Φ_i are separable multidimensional polynomials. The PR ROM technique is simple, versatile, and scalable. Although not implemented in the MOOSE-STM, PR also has convenient properties that allow for intrinsic error estimation. However, PR does make assumptions that reduce its applicability in a general context. First, the data from the training set are assumed to be fully known (without any uncertainty), so the QoI error from running the FOM is not considered. Second, the parameters are assumed to be uncorrelated, this can cause issues when two parameters are not independent, i.e. if the pebble unloading rate has some correlation with the burnup limit. As such, the actual parameters to fit the ROM must be chosen carefully. Third, the variance in the ROM (error estimation) is assumed constant in the parameter space, so regions in the training set with lots of data are assumed to have the same error as regions with little data. Finally, because of its generality and simplicity, PR may require a large training set to generate an accurate ROM.

PR is a good point of reference for evaluating the performance of other methods. It is significantly cheaper to train, evaluate, and store than other model types. However, it can also be very limited—PR can perform poorly for responses with nonlinear parameter dependence and can easily fall victim to over-fitting for high-dimensional inputs. Due to its simplicity, there are not many parameters to tune. Cross-validation results were obtained for 3rd-degree and 4th-degree polynomial regressions. Although the MOOSE-STM supports ridge regression, this data could be handled well by ordinary least-squares. Ordinary least-squares is preferred, as ridge regression requires additional tuning for the penalty parameter.

4.2.3 Gaussian Process

GP regression is a nonlinear ROM technique, which is significantly different than the PR and PCE methodologies. The formulation of a GP ROM is quite complex, but in overly simplistic terms, GP modeling is driven by the idea that training points that are "close" in their parameter space will be "close" in their QoI space. Closeness in the parameter space is driven by the covariance function $k(\vec{\xi}, \vec{\xi}')$. The GP model consists of an infinite collection of functions, all of which agree with the training and observation data. Importantly, the collection has closed forms for second-order statistics (mean and variance). When used as a surrogate, the nominal value is chosen to be the mean value. The method can be broken down into two steps: defining the prior distribution then conditioning the observed data. A GP is a (potentially infinite) collection of random variables, such that the joint distribution of every finite selection of random variables from the collection is a Gaussian distribution.

$$\mathcal{GP}(\mu(\vec{\xi}), k(\vec{\xi}, \vec{\xi}')). \tag{11}$$

In an analogous way that a multivariate Gaussian is completely defined by its mean vector and its covariance matrix, a GP is completely defined by its mean function and covariance function. The (potentially) infinite number of random variables within the GP correspond to the (potentially) infinite points in the parameter space that our ROM can be evaluated at. While the only apparent decision in the above formulation is the choice of covariance function, most covariance functions will contain hyper-parameters of some form, which need to be selected in some manner. Determining the value of these hyper-parameters manually is a terribly inefficient task; therefore, the MOOSE-STM uses a tuning procedure to set these values automatically during the training process. GP is a powerful ROM technique with its versatility, flexibility, and generality.

The choice of covariance function is an essential component in defining a GP regression. This work uses the common squared-exponential covariance kernel:

$$k(\vec{\xi}, \vec{\xi}') \equiv \sigma_f^2 \exp\left(-\frac{r_l(\vec{\xi}, \vec{\xi}')^2}{2}\right) + \sigma_n^2 \delta_{\vec{\xi}, \vec{\xi}'}, \tag{12a}$$

where,

$$r_l(\vec{\xi}, \vec{\xi}') \equiv \sqrt{\sum_{i=1}^{N} \left(\frac{\vec{\xi}_i - \vec{\xi}_i'}{l_i} \right)^2}. \tag{12b}$$

Some important remarks about this function:

- The function is *translation invariant* - it only depends on the difference between points $\vec{\xi} - \vec{\xi}'$.
- The function is *isotropic* - it only cares about the magnitude of $\vec{\xi} - \vec{\xi}'$, and not the direction.

Many common covariance functions have these properties (including all of the options in the MOOSE-STM). Although these properties are advantageous in quantifying similarity, they have the side effect that predictors near each other will have highly correlated covariances with the rest of the data. This is an issue because several operations in training and evaluation with GPs involve inverting the dense matrix:

$$\mathbf{K}(\boldsymbol{\xi}, \boldsymbol{\xi}) \equiv \begin{bmatrix} k(\vec{\xi}_1, \vec{\xi}_1) & \cdots & k(\vec{\xi}_1, \vec{\xi}_m) \\ \vdots & & \vdots \\ k(\vec{\xi}_m, \vec{\xi}_1) & \cdots & k(\vec{\xi}_m, \vec{\xi}_m) \end{bmatrix}. \tag{13}$$

Because Equation (12) is translation-invariant and isotropic, this matrix can easily become very-near-singular if there are too many samples near each other. To handle this, the implementation in the MOOSE-STM performs all necessary linear solves with a stable direct method using the Cholesky factorization. Computing this factorization is an $\mathcal{O}(n^3)$ operation, making it difficult to perform regression for large data-sets.

The covariance function, see Equation (12), has a number of hyper-parameters $\boldsymbol{\theta}$ that must be optimized:

- $\vec{l} \in \mathbb{R}^n_{>0}$, the length scales for each predictor
- $\sigma_n^2 \in \mathbb{R}_{>0}$, the noise variance in the inputs
- $\sigma_f^2 \in \mathbb{R}_{>0}$, signal variance in the response.

The problem of choosing these parameters can be posed as an optimization problem, where the objective function is the *marginal likelihood*. This gradient is incredibly expensive to calculate—it

requires computing a Cholesky factorization ($\mathcal{O}(N^3)$) for the current hyper-parameters and then performing a linear solve for each hyper-parameter ($\mathcal{O}(LN^2)$). When optimizing, this gradient may need to be calculated hundreds or thousands of times for each response. For the sample set produced in this study ($N = 3,000$), the cost of hyper-parameter tuning by direct gradient-based optimization was prohibitively expensive. Although, small-batch optimization methods are being considered for implementation in the MOOSE-STM to reconcile this issue.

4.2.4 Artificial Neural Network

The ANN capability in the MOOSE-STM is relatively new and only simple feed-forward neural networks have been implemented. The implementation uses the underlying objects imported from `libtorch` (C++ API of `pytorch`) [11]. For a more detailed introduction to neural networks, we refer the reader to Reference [9]. The architecture of a simple feed-forward neural network is presented in Figure 8. The first layer from the left to the right are referred to as input and output layers, while the layers between them are the hidden layers.

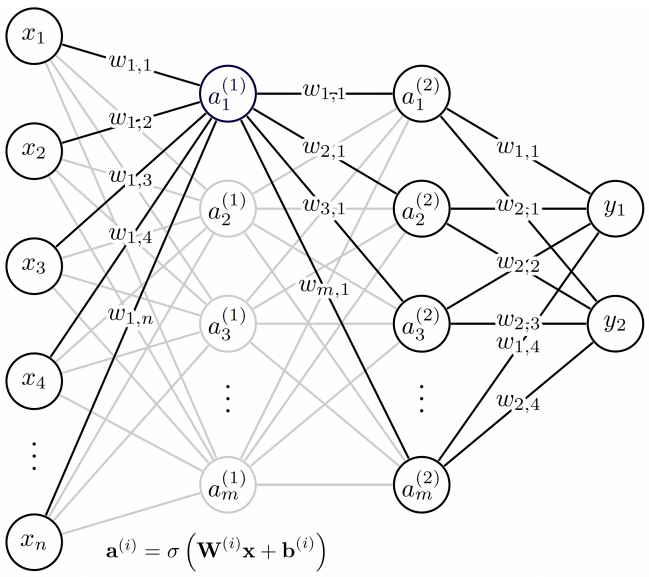


Figure 8: The architecture of the simple feed-forward ANN in the MOOSE-STM.

We see that the outputs ($\mathbf{y}$) of the neural net can be expressed as function of the inputs ($\mathbf{x}$) and the corresponding model parameters (weights $w_{i,j}$, organized in weight matrices $\mathbf{W}$, and biases, b_i

organized in the bias vector (**b**) in the following nested form:

$$\mathbf{y} = \sigma(\mathbf{W}^{(3)}\sigma(\mathbf{W}^{(2)}\sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \mathbf{b}^{(3)}), \tag{14}$$

where σ denotes the activation function. Currently, the MOOSE-STM implementation supports `relu`, `elu`, `gelu`, `sigmoid`, and `linear` activation functions. In this implementation, no activation function is applied on the output layer. It is apparent that the real functional dependence (target function) between the inputs and outputs is approximated by the function in Equation (14). As in most cases, the error in this approximation depends on the smoothness of the target function and the values of the model parameters. The weights and biases in the function are determined by minimizing the error between the approximate outputs of the neural net corresponding reference (training) values over a training set.

Node biases and weights are determined by stochastic gradient descent. There are a number of parameters associated with the learning process that can affect the quality of the resulting model. For this work, the learning parameters were selected to ensure stable and consistent convergence. These values were identified by fixing a model configuration (single hidden layer with 64 neurons) and varying each learning parameter. Table 7 summarizes the values found to work well for the fixed model that were thus used when training all other model configurations.

Table 7: Learning parameters used for neural networks.

Parameter	Value
Number of training epochs (N_e)	3,000
Relative loss stopping criteria (ϵ)	1e-4
Learning rate (l_r)	1e-3
Number of training batches (N_b)	10

There are several hyper-parameters to choose from when designing a neural network.

- Number of hidden layers (N_l)
- Number of neurons per hidden layer (N_n)
- Type of activation function to use in each layer.

In this work, the `relu` activation function was used in all cases. To search for an optimal model, a search over possible network configurations was performed. This search was performed in

two steps. First, a naive approach was attempted in which neurons were added to a single-layer network until over-fitting occurred (identified by an increase in cross-validation RMSE). The number of neurons in the first layer was fixed, and neurons were then progressively added to a second layer. This procedure was continued for a third layer. Table 8 summarizes the network configurations tested in this study.

Table 8: Neural network configurations considered in first search.

$\boldsymbol{N_l}$	$N_{n,1}/N_{n,2}/N_{n,3}$
1	$\{8, 16, 32, 64, 128\}/0/0$
2	$64/\{8, 16, 32, 64\}/0$
3	$64/32/\{8, 16, 32, 64\}$

This procedure was poor—although a good single-layer model was identified (64 neurons), freezing the first layer to contain a large number of neurons resulted in all two- and three-layer models being overly complex and caused over-fitting. A second search was performed over two- and three-layer models with fewer neurons per layer. This was performed using a full grid search, with the following procedure:

- Choose the numbers of hidden neurons in each layer to consider.
- Denote $\boldsymbol{N_{n,i}}$ as a vector of candidate neuron counts in layer i.
- Form the Cartesian product of all possible combinations of candidate neuron counts. For two-layer models, this is $\boldsymbol{N_{n,1}} \otimes \boldsymbol{N_{n,2}}$. For three-layer models, this is $\boldsymbol{N_{n,1}} \otimes \boldsymbol{N_{n,2}} \otimes \boldsymbol{N_{n,3}}$.
- For each combination, train a neural network, with each model scored by their cross-validation RMSE.

The neuron counts considered in this step are summarized in Table 9. Performing a grid-search is straightforward using the MOOSE-STM cross-validation capability.

Table 9: Candidate neurons per layer used to define grid-search for two- and three-layer models.

$\boldsymbol{N_{n,i}}$	Candidates (2-layer)	Candidates (3-layer)
$\boldsymbol{N_{n,1}}$	4, 8, 16, 32	8, 16, 32
$\boldsymbol{N_{n,2}}$	4, 8, 16, 32	8, 16, 32
$\boldsymbol{N_{n,3}}$	0	4, 8, 16

4.3 Initial Reduced-Order Model Study

In this initial ROM study, a number of ROMs were trained using various methodologies and hyper-parameter selection, see Section 4.2 for descriptions. For this study, a training set was produced consisting of 3,000 sets of parameters from random Latin hyper-cube sampling and a corresponding number of QoI sets generated from running the FOM described in Section 2. Three types of regression models were considered in this study—PR, GP, and ANN. Each of these models is specified by hyper-parameters $\boldsymbol{\theta} = \{\theta_1, ..., \theta_L\}$ that define how the model learns a functional representation of the data. A primary research goal in this study was identifying good practices for tuning the hyper-parameters for each model.

In all cases, a five-fold cross-validation was used to assign RMSE scores for each model type, for each response. Figure 9, Figure 10, and Figure 11 cover the grid-search cross-validation results for single-, double-, and triple-layer networks respectively. Some brief observations:

- In all cases, the data series ends with a model configuration (ANN-128, ANN-32/32, and ANN-16/16/8) that appears to be overly complex for some of the responses, as evidenced by the increase in RMSE.
- For the two- and three-layer models, the score for the best fit single-layer network (ANN-64) is included for comparison.
- In all cases, ANN-64 has a lower cross-validation RMSE than any of the candidate two- and three-layer models.

Figure 12 contains cross-validation results for the best-fit models from each type and the most pertinent takeaways from this study. Results are provided for third- and fourth-degree polynomial models to serve as a point of reference for the other model types. A single result is provided for the best-fit neural network model identified from the cross-validation grid-search—the single-layer, 64 neuron network. For GP regression, results are provided for the best-fit models identified from small-batch tuning and cross-validation, using batch sizes of m=100, 300, and 600. As a general observation, all model types explored in this study had a relative RMSE of less than 1% for all response types—in some cases, this value was even less than 0.1%.

Interestingly, there were several cases where fourth-order polynomial regression outperformed the nonlinear models. This occurred for T_{mod}, T_{ref}, and f_p. This result is somewhat unexpected but may indicate these responses are "mostly linear" in the predictors. The nonlinear models (GP and ANN) appear to outperform the polynomial for all other response types. A common trend is that the single-layer neural network outperforms the best GP, sometimes with a significant margin. That said, the GP models were hampered due to difficulty in hyper-parameter tuning.

For the eigenvalue specifically, it is useful to convert the relative RMSE to an absolute error in pcm. These values are reported in Table 10. As expected, the nonlinear models significantly outperformed polynomial models for the eigenvalue.

Table 10: Five-fold RMSE for k_{eff} in pcm.

Model	PR-3	PR-4	ANN-64	GP-100	GP-300	GP-600
RMSE (pcm)	221	146	23	45	35	33

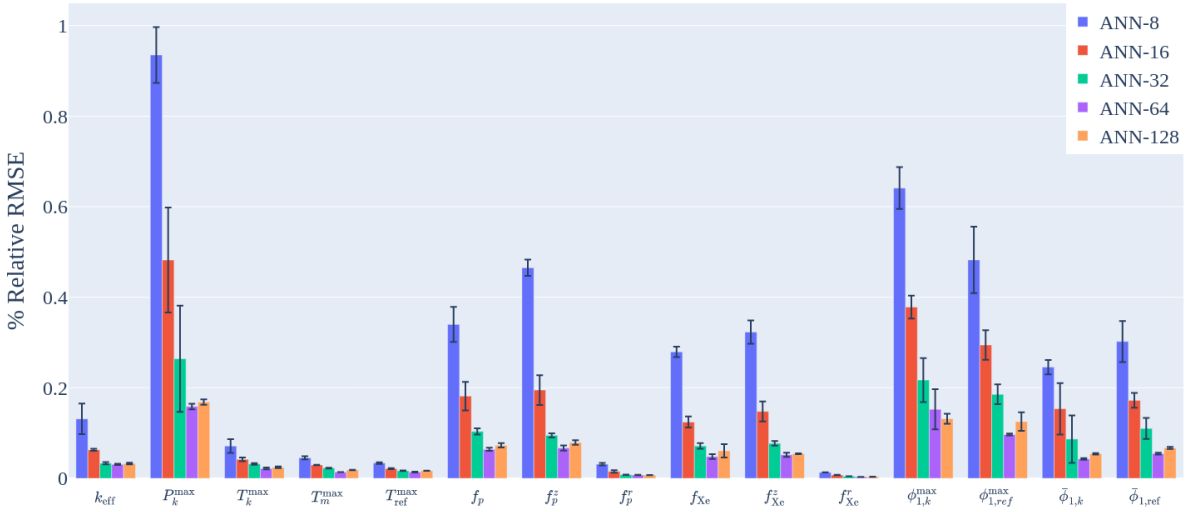


Figure 9: % Relative RMSE for several candidate single-layer neural networks. It is apparent that increasing the neuron count from 64 to 128 causes overfitting for some of the responses.

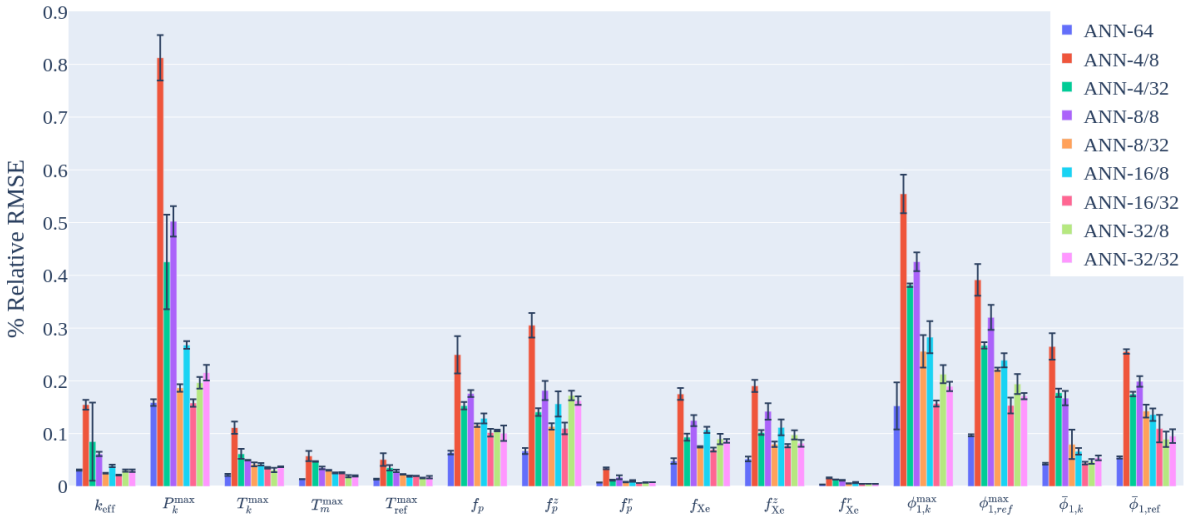


Figure 10: % Relative RMSE for several low-width, two-layer neural networks. In all cases, even parsimonious two-layer networks are outperformed by the best single layer network (ANN-64).

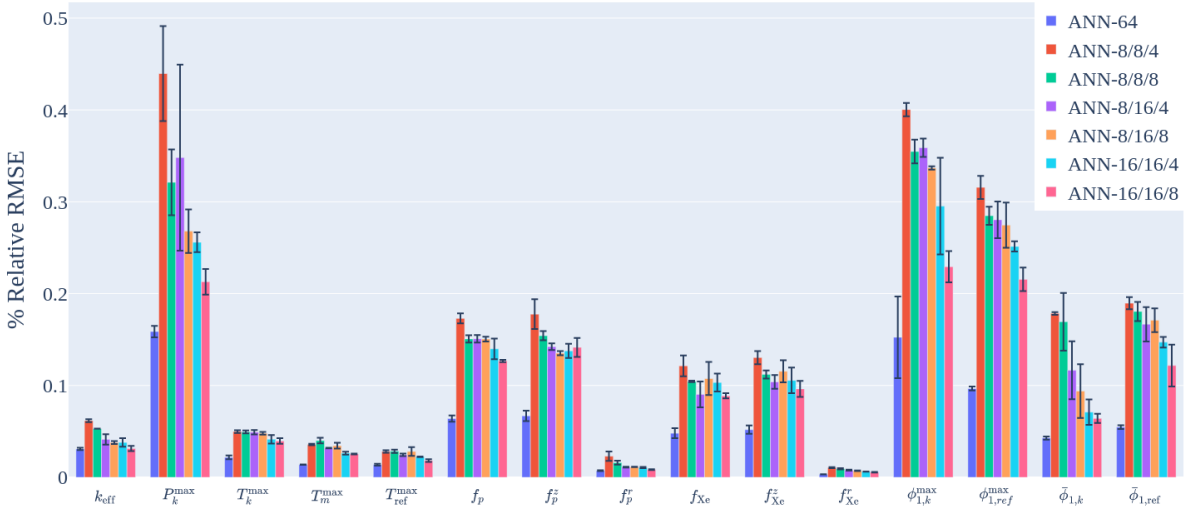


Figure 11: % Relative RMSE for several low-width, three-layer neural networks. In all cases, even parsimonious three-layer networks are outperformed by the best single layer network (ANN-64).

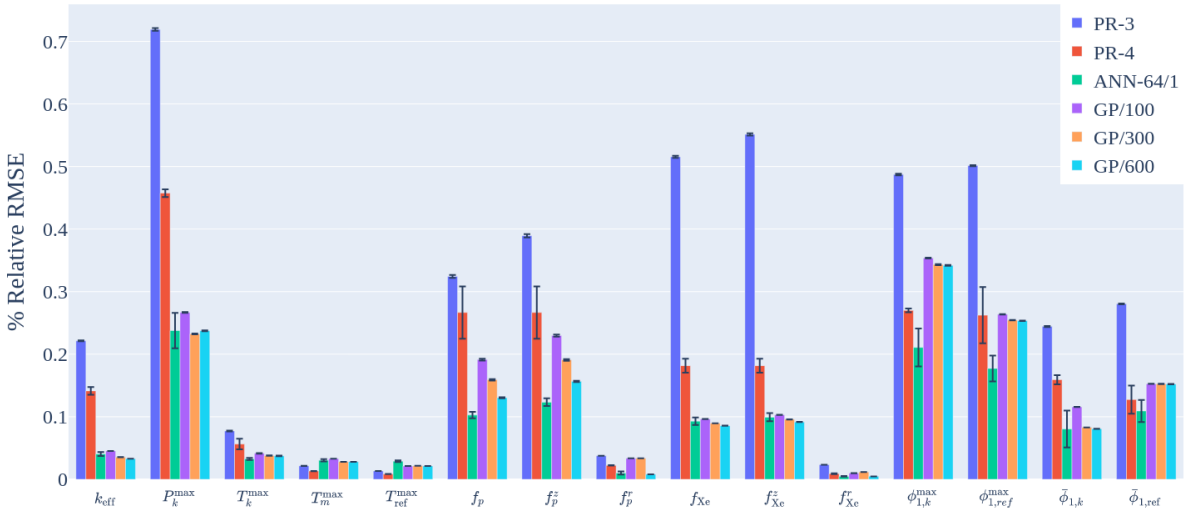


Figure 12: % Relative RMSE for several candidate models.

5. Conclusions

In this work, an initial study was conducted on the ROM generation of pebble-bed HTGRs. This included: the development of a representative equilibrium-core HTGR model, a sensitivity analysis of model parameters and relevant QoIs, and a characterization of the ROM techniques included in the MOOSE-STM.

The representative HTGR model is a steady-state equilibrium core built using the reactor analysis application, BlueCRAB. It is a multiphysics model with four different phenomena: neutronics, streamline depletion, porous-flow TH, and pebble and TRISO kernel heat conduction. All of these physics were coupled using a fixed-point iteration strategy with the MOOSE MultiApp system. Along with the development of the multiphysics model, parameters and QoIs were identified for the study, which are necessary for sensitivity analysis and ROM generation. These quantities included design parameters and quantities across all the physical mechanisms of the model.

To perform the sensitivity study, a PCE methodology was used to compute statistical moments and Sobol global sensitivity metrics for each parameter and QoI. From the study, we found that TH parameters had a relatively small impact on the model behaviour. Although this conclusion might

change in future model iterations, this analysis indicates that these quantities could be removed from the ROM generation. Filtering out these quantities will make training a ROM less intensive and error-prone, making the resulting ROM more robust.

The purpose of the ROM study was to characterize various ROM techniques and gain insight on which technique to recommend and determine a proper specification of hyper-parameters. Several different ROM techniques were explored, including PR, GP, and ANN. The performance of each generated ROM was quantified using k-fold cross-validation. From the results, it appears that either a single-layer ANN or GP would be strong candidates for an HTGR surrogate. However, this conclusion may change with future model iterations.

There is much work to be done to develop a workflow for HTGR ROM generation for optimization. For the equilibrium core model, more design parameters can be included, such as fuel enrichment, pebble and kernel sizes, and reactor dimensions. We were not able to include these parameters since it requires a re-tabulation of cross sections and the development of advanced mesh generation techniques. Another task is to develop a transient model, specifically a running-in, approach to asymptotic core model. This model may require more complexities, such as a criticality search routine to obtain control rod positions, and the incorporation of fuel performance physics. Furthermore, a transient model will have a different design and output space, which needs to be identified. Once these model improvements are made, the sensitivity and ROM study presented in this report can be repeated. The penultimate goal of this work is to create an optimization workflow using ROMs. As such, another study must be performed to investigate the use of these ROMs in the context of optimization.

REFERENCES

[1] Balestra, P., S. Schunert, R. W. Carlsen, A. J. Novak, M. D. DeHart, and R. C. Martineau, Pbmr-400 benchmark solution of exercise 1 and 2 using the moose based applications: Mammoth, pronghorn, in *Proceedings of PHYSOR 2020: Transition to a Scalable Nuclear Future*, Cambridge, United Kingdom, 2020.

[2] Brits, Y., F. Botha, H. van Antwerpen, and H. W. Chi, A control approach investigation of the xe-100 plant to perform loadfollowing within the operational range of 100–25–100%, *Nuclear*

Engineering and Design, *329*, 12–19, doi:https://doi.org/10.1016/j.nucengdes.2017.11.041, 2018.

[3] Gaston, D. R., et al., Physics-based multiscale coupling for full core nuclear reactor simulation, *Annals of Nuclear Energy*, *84*(1), 45–54, doi:10.1016/j.anucene.2014.09.060, 2015.

[4] Gerstner, T., and M. Griebel, Numerical integration using sparse grids, *Numerical Algorithms*, *18*(3), 1572–9265, doi:10.1023/A:1019129717644, 1998.

[5] Lee, C. H., Y. Jung, H. Park, E. Shemon, J. Ortensi, Y. Wang, V. Labouré, and Z. Prince, Griffin Software Development Plan, *Research Report INL/EXT-21-63185, ANL/NSE-21/23*, Idaho National Laboratory, Argonne National Laboratory, 2021.

[6] Martineau, R. C., The moose multiphysics computational framework for nuclear power applications: A special issue of nuclear technology, *Nuclear Technology*, *207*(7), iii–viii, doi: 10.1080/00295450.2021.1915487, 2021.

[7] Morris, M. D., Factorial sampling plans for preliminary computational experiments, *Technometrics*, *33*(2), 161–174, doi:10.1080/00401706.1991.10484804, 1991.

[8] Mulder, E. J., and W. A. Boyes, Neutronics characteristics of a 165 mwth xe-100 reactor, *Nuclear Engineering and Design*, *357*, 110,415, doi:https://doi.org/10.1016/j.nucengdes.2019.110415, 2020.

[9] Müller, B., J. Reinhardt, and M. T. Strickland, *Neural networks: an introduction*, Springer Science & Business Media, 1995.

[10] Novak, A., R. Carlsen, S. Schunert, P. Balestra, R. Slaybaugh, and R. Martineau, Pronghorn: A multidimensional coarse-mesh application for advanced reactor thermal hydraulics, *Nuclear Technology*, *207*(7), 1015–1046, doi:10.1080/00295450.2020.1825307, 2021.

[11] Paszke, A., et al., Pytorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, pp. 8024–8035, Curran Associates, Inc., 2019.

[12] Saltelli, A., Making best use of model evaluations to compute sensitivity indices, *Computer Physics Communications*, *145*(2), 280–297, doi:10.1016/S0010-4655(02)00280-1, 2002.

[13] Slaughter, A. E., Z. M. Prince, P. German, I. Halvic, W. Jiang, B. Spencer, S. Dhulipala, and D. Gaston, Moose stochastic tools: A module for performing parallel, memory-efficient in situ stochastic simulations, *SoftwareX*, *Manuscript in review*, 2022.

[14] Stewart, R., P. Balestra, D. Reger, and E. Merzari, Generation of localized reactor point kinetics parameters using coupled neutronic and thermal fluid models for pebble-bed reactor transient analysis, *Annals of Nuclear Energy*, *174*, 109,143, doi:https://doi.org/10.1016/j.anucene.2022.109143, 2022.

[15] Sudret, B., Global sensitivity analysis using polynomial chaos expansions, *Reliability Engineering & System Safety*, *93*(7), 964–979, doi:10.1016/j.ress.2007.04.002, 2008.