**Methods and Tools for SNAP Multiphysics Benchmark Evaluation**

Isaac Naupa, \* Samuel Garcia, † Ben Lindley, † Dan Kotlyar \*

*\*Georgia Institute of Technology, 770 State St NW, Atlanta, GA 30313, iaguirre6@gatech.edu*

*†University of Wisconsin Madison, 1500 Engineering Dr, Madison, WI 53706, sgarcia9@wisc.edu*

*[leave space for DOI, which will be inserted by ANS]*

INTRODUCTION

Advanced nuclear reactors and microreactors are receiving significant attention in the nuclear industry as a potential to alter the traditional paradigm for the technology. These systems can be promptly deployable with a fraction of the capital cost of large nuclear systems. To promote and license this technology there is a need to understand the limitations and major challenges. For example, there is a need to capture the loss and redistribution of hydrogen in the system that directly affects cycle length, power and temperature distributions. Therefore, multiphysics simulations are needed to capture the inter-correlation between the different multiphysics. Such Multiphysics capabilities are not limited only to microreactors but are general to other advanced systems.

The Nuclear Energy Advanced Modeling and Simulation (NEAMS) program aims to develop high-fidelity tools to aid the design and deployment of advanced systems, such as microreactors. As part of this program the Multiphysics Object Oriented Simulation Environment (MOOSE) framework has been developed to enable an integrated multiphysics approach [1]. However, with the development of these tools and frameworks arises the need for verification and validation. Many of these systems can become complicated when several physics are introduced and hence necessitate benchmarking to measure performance and fidelity. Benchmarking requires meticulous compilation and documentation of all experimental data and computational models. Among the goals of the current work is to present computational tools that we have developed specifically for the purpose of streamlining the collection and integration of data into multiphysics sequences.

This work is part of an ongoing joint effort between Georgia Tech, University of Wisconsin-Madison, BWX Technologies, and Idaho National Laboratory that leverages extensive experimental data from the (Systems for Nuclear Auxiliary Power) SNAP program to validate the performance of specific NEAMS tools in modeling effects that are unique to microreactor technology [2, 3, 4].

Computational Frameworks

A multiphysics benchmark evaluation compliant with the IRPhEP evaluation guide requires systematic assessments of experimental data with meticulous compilation and documentation. Therefore, we are developing a SFCOMPO [5] styled database and an automated consistent computational sequence for modeling purposes. Both developments take the form of python-based packages with proper documentation, error checking, and unit testing. The SFCOMPO styled databased package is called ‘snapReactors’ while the package supporting the automated multiphysics sequence is called ‘SIMBA’ (Serpent Interface for Multiphysics Based Applications). These packages are open source and are hosted on public repositories for use of the nuclear engineering community [6, 7].

The SFCOMPO styled database ‘snapReactors’ stores all relevant data for the benchmark for various SNAP systems. It is structured in object orientated fashion allowing for clean organization and data handling. The database is inherently dynamic such that removal and/or addition of information can be done easily. Post processing methods are offered that translate collected data into common structures used in computational reactor physics workflows.

The automated multiphysics sequence ‘SIMBA’ provides a set of functions and utilities for common workflows in reactor based multiphysics analysis. SIMBA also aids in modeling sensitivity studies used for demonstrating the accuracy and quantifying the error or bias uncertainty of reactor physics calculations. The automated sequence uses the Monte Carlo neutronics code Serpent [8] for benchmarking purposes as the reference solution generator for other software such as Griffin [9].Below is an example workflow of the sequence that will be detailed in the following sections.

Diagram

Description automatically generatedFig. 1. Snapshot of database object oriented hierarchical

**snapReactors**

The database is meant to be structured in a similar fashion to SFCOMPO, aiming to facilitate the search and visualization of experimental data from spent fuel [5].  
The snapReactors package aims to do this as well but at a much larger scale, including data from isotopic compositions of materials, material thermophysical and mechanical properties, dimensions, and operational conditions. All data stored in the database is documented appropriately with references and verified calculations. In this way the database can be used for validation purposes. The database is built as a python package and structures the data in a collection of containers, aimed at providing easy access and navigation of data. The database also has a suite of functions for added utility such as exporting data into useful formats such as HDF5 or XML formats. A sample image of the database hierarchical structure is shown in Fig. 2.

The snapReactors package takes different containers representing general reactor attributes and organizes them into hierarchical fashion. The highest level being a reactor object, *i.e.*, SNAP8ER, then reactor states *i.e.*, cold zero power, then components, *i.e.*, upper grid plate, then materials, *i.e.*, Hastelloy – N, and finally properties and dimensions, *i.e.*, thermal conductivity or active fuel length. The containers are meant to be general enough to be reactor type agnostic as well as provide a variety of utility functions such as being able to store and access the reference of each container/data set, the appropriate units, descriptions, etc. These features could then be used in the automated sequence for the creation of thorough documented models.

Text

Description automatically generated

Fig. 2. Snapshot of database hierarchical structure.

**SIMBA**

Serpent input files can be become intricate and extensive when dealing with large problems that require multiple physical phenomena to be captured. Traditionally this requires for the user to tailor the construction of the input file in a manner that would capture the appropriate phenomena, *i.e.*, discretization of fuel pins for cross-section generation, imposing temperature and density fields, imposing material grading distributions., etc. If not done manually, often “back of the envelope” scripting is done to facilitate the creation of the input file. While these scripts are sufficient, they are frequently focused on a specific problem and are not well documented and/or readily available for the broader community. Typically, most of the effort associated to such analyses is spent on setting up the tools and scripts. SIMBA aims to reduce the time spent setting up the workflow by providing a set of functions and utilities that can be easily adopted for common workflows in reactor based multiphysics analysis.

The SIMBA package takes traditional universe based Boolean geometry and organizes it in an object-oriented fashion into distinct containers with their appropriate attributes and methods. A set of functions called builders use these objects to create typical universe-geometry objects used in computational reactor analysis. A sample figure of a fully automated model of the SNAP 8 Experimental Reactor S8ER with applied temperature and density fields is presented below.

Chart, bar chart, histogram

Description automatically generated

Fig. 3. Axial view of S8ER core with applied density and temperature fields.

USE CASES AND DEMONSTRATION

This section will focus on demonstrating the typical use case of data collection and exporting data into typical computational reactor physics workflows. The benefits of having the package on the python platform will also be demonstrated by highlighting the ease of use of the software and easy integration with other python libraries such as ‘serpentTools’, a rising in popularity post processing tool for Serpent applications[10]. The snapReactors package offers both an input file and memory-based approach. For users with limited python experience the input file-based approach is more convenient and significantly reduces the command count.

**SNAP 8 Experimental Reactor Demonstration**

The workflow is split into three stages where the input data will be loaded into the python environment with snapReactors, the data will be used to create a serpent model with SIMBA, and the data will be post processed using serpentTools. First the input file data is read in using the reactor state reader. A snapshot of the input file is in Fig. 4, where the data is organized hierarchically with reference, units, uncertainty, and descriptions can be given. Text

Description automatically generated with medium confidence  
Fig. 4. Snapshot of reactor state data input file.

The data is neatly organized into pythonic dictionaries where the user can simply access objects by their name. The component objects are accessed from the cold power reactorState object and are saved to be used in the following SIMBA template. Text

Description automatically generated  
Fig. 5. Coding block 1 of snapReactors-SIMBA workflow.

The SIMBA template is a configured model of the SNAP 8 Experimental Reactor. The template is made using builder functions from SIMBA to automate the creation of the serpent model from input data. The template takes in data from the cold power reactor state object and exports it into serpent format. The script generates a material, geometry, and main file each with the corresponding model data and parameters extracted from the database including descriptions and references listed at the top of each file. The resulting serpent models are displayed below in Fig. 7.

Text

Description automatically generated  
Fig. 6. Coding block 2 of snapReactors-SIMBA workflow.

A picture containing accessory

Description automatically generatedChart, bar chart, histogram

Description automatically generated  
Fig. 7. Snapshot of the Serpent radial and axial models of the SNAP 8 Experimental Reactor generated by SIMBA.

The python environment allows for easy integration with other libraries, allowing for flexible and custom workflows. The serpentTools library is used here to facilitate the creation of post processing scripts for convenience of the user.  
Text

Description automatically generated

Fig. 8. Coding block 3 of snapReactors-SIMBA workflow.

As an example, the user can readily generate plots of history effects as well as output material data into organized tables. Both are shown in Fig. 9.

Table

Description automatically generated

Chart

Description automatically generated

Fig. 9. Snapshot of post processing history effects and material data using serpentTools.

Conclusion

The snapReactors-SIMBA workflow offers a novel approach for traditional computational reactor physics calculations. The packages excel in an environment where large collections of data need to be compiled and documented thoroughly such as a benchmark. For large simulations where multiple physics need to be accounted for, SIMBA facilitates this through automating repetitive and extensive tasks. The python environment which both packages are based on allows for well-integrated workflows with other libraries and allows for power users to take advantage with great flexibility. This is demonstrated for the case of the S8ER where the use of multiple packages is exploited.

Both packages are still in early constant development with updates being deployed frequently. For those who wish to use the codes and help development or report bugs, all documentation and guidelines are found through the GitHub repositories [6, 7].

Acknowledgements

This work was supported by funding received from the DOE Office of Nuclear Energy’s Nuclear Energy University Program under contract number DE-NE0009217. Serpent is developed and maintained by the VTT Technical Research Centre of Finland, LTD, which does not endorse nor financially support SIMBA nor the content of this paper.

**REFERENCES**

1. A. LINDSAY, ‘2.0 - MOOSE: Enabling Massively Parallel Multiphysics Simulation’. SoftwareX, vol. 20, 2022, <https://doi.org10.1016/j.softx.2022.101202>.

2. S. GARCIA, “Validation of SNAP8 Criticality Configuration Experiments using SERPENT,” In Proceedings of ANS Winter Conference (2022).

3. I. NAUPA, “Verification of the Serpent-Griffin Workflow using the SNAP 8 Experimental Reactor,” In Proceedings of ANS M&C Conference (2023).

4. G. GIUDICELLI, ‘The Virtual Test Bed (VTB) Repository: A Library of Reference Reactor Models Using NEAMS Tools’, Nuclear Science and Engineering (2023), DOI: 10.1080/00295639.2022.2142440

5. A. F. MICHEL-SENDIS, "SFCOMPO-2.0: An OECD NEA Database of Spent Nuclear Fuel Isotopic Assays, Reactor Design Specifications and Operating Data", Annals of Nuclear Energy 110, 779-788 (2017), [www.sciencedirect.com/science/article/pii/S0306454917302104](http://www.sciencedirect.com/science/article/pii/S0306454917302104).

6. COMPUTATIONAL REACTOR ENGINEERING LAB, Serpent Interface for Multiphysics Based Applications (SIMBA), (2022), GitHub Repository, <https://github.com/DanKotlyar/SERPENT-INPUT-GENERATOR>

7. COMPUTATIONAL REACTOR ENGINEERING LAB, snapReactors, (2022), GitHub Repository, <https://github.com/CORE-GATECH-GROUP/SNAP-REACTORS>

8. J. LEPPÄNEN, ‘The Serpent Monte Carlo Code: Status, Development and Applications in 2013’. Annals of Nuclear Energy, vol. 82, 2015, pp. 142–150, <https://doi.org10.1016/j.anucene.2014.08.024>.

9. Y. WANG, ‘Rattlesnake: A MOOSE-Based Multiphysics Multischeme Radiation Transport Application’. Nuclear Technology, vol. 207, no. 7, Taylor & Francis, 2021, pp. 1047–1072,

https://doi.org10.1080/00295450.2020.1843348.

10. A. JOHNSON, ‘SerpentTools: A Python Package for Expediting Analysis with Serpent’. Nuclear Science and Engineering, vol. 194, no. 11, Taylor & Francis, 2020, pp. 1016–1024, <https://doi.org10.1080/00295639.2020.1723992>.