SCIENTIFIC WORKFLOWS FOR MCNP6 AND PROTEUS WITHIN THE NEAMS WORKBENCH

Kurt A. Dominesey, Peter J. Kowal, Jonathan A. Eugenio, and Wei Ji*
Department of Mechanical, Aerospace, and Nuclear Engineering
Rensselaer Polytechnic Institute
110 8th Street, Troy, NY 12180 U.S.A.

domink@rpi.edu, kowalp@rpi.edu, eugenj@rpi.edu, jiw2@rpi.edu

ABSTRACT

In this work we integrate MCNP and PROTEUS, two high-fidelity reactor physics codes, into the Nuclear Energy Advanced Modeling and Simulation (NEAMS) Workbench. Specifically, we enhance the user interface of each code by developing full-featured editor services. Next, we re-define MCNP textual outputs in HDF5 formats, at great convenience to the user. Further, these HDF5 outputs are (optionally) coerced to the OpenMC format, enabling powerful and common post-processing workflows. For compatible use of MCNP and PROTEUS, we develop a model unification workflow, converting PROTEUS- to MCNP-compatible meshes and generating a corresponding template MCNP deck. Finally, we investigate cooperative use of MCNP and PROTEUS for hybrid Monte Carlo and deterministic variance reduction through Consistent Adjoint-Driven Importance Sampling (CADIS), demonstrated with a fixed-source shielding problem. Ultimately, we find our integration effort dramatically streamlines pre- and post-processing with MCNP and PROTEUS and enables transformative hybrid Monte Carlo and deterministic workflows.

KEYWORDS: MCNP, PROTEUS, CADIS, scientific workflows, NEAMS Workbench

1. INTRODUCTION

The Nuclear Energy Advanced Modeling and Simulation (NEAMS) Workbench [1] seeks to empower reactor physicists with versatile, comprehensive, and intuitive interfaces to and between scientific software used to simulate nuclear reactors. Primary among these tools are neutron transport solvers, which compute the distribution of neutron flux within a reactor, as well as the multiplication factor. For this purpose, we here integrate MCNP6.2 [2] and PROTEUS [3] (two high-fidelity neutron transport solvers) into the NEAMS Workbench (expanding on previous work [4]). Doing so, we equip Workbench users with powerful neutronics solvers, enhance the productivity of MCNP6 and PROTEUS users, and facilitate novel hybrid Monte Carlo and deterministic analyses.

This integration entails firstly to refactor the user interface (input specification) of both codes, for improved ease-of-use and versatility. Next, we facilitate post-processing and visualization of both

*Corresponding author

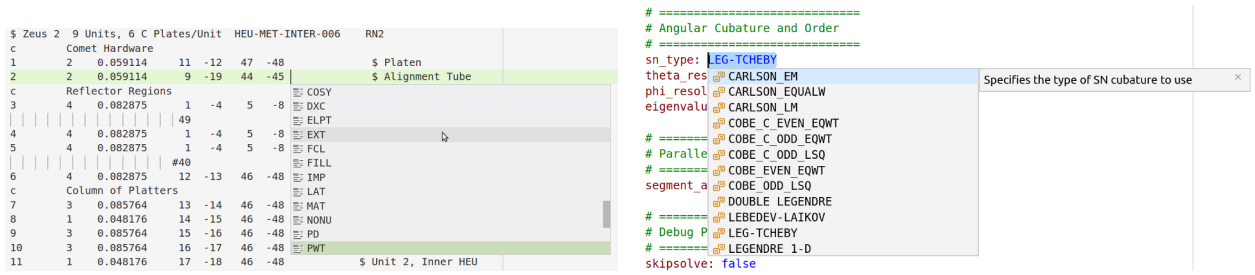
codes by standardizing output formats and leveraging previously-established utilities. Thirdly, we explore the compatible use of MCNP6.2 and PROTEUS as independent analysis tools. Finally, we examine cooperative use, typified here by a hybrid Monte Carlo and deterministic variance reduction scheme known as Consistent Adjoint-Driven Importance Sampling (CADIS) [5].

2. USER INTERFACE AND PRE-PROCESSING

2.1. MCNP Decks

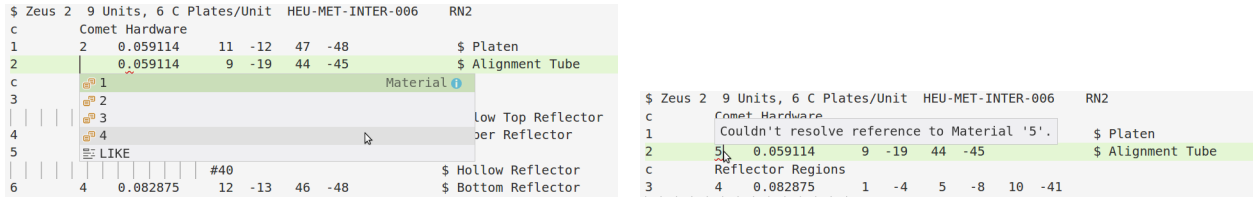
To specify input for Monte Carlo simulations, MCNP6.2 employs a Domain-Specific (Modeling) Language, or DS(M)L. However, despite the ubiquitous use of MCNP among nuclear engineers, no text editors support this custom DSML. Accordingly, users are deprived of common code editing features, such as content-assist, input validation, and reference resolving. Analysts must rely solely on trial-and-error, which can render MCNP input preparation time-consuming and error-prone.

To rectify this hardship, we have developed comprehensive editor services using a language workbench known as Xtext [6]. We show a subset of these features in Figure 1a (content-assistance) and Figure 2 (reference-resolving). Crucially, these editor services enforce DSML syntax and validate MCNP constraints with immediate visual feedback, dramatically enhancing productivity.



(a) MCNP6.2 deck **(b) PROTEUS driver**

Figure 1: Content-assist and autocompletion for MCNP and PROTEUS



(a) Suggesting a reference **(b) Validating a reference**

Figure 2: Reference resolving in the MCNP DSML

To implement these editor services in Workbench, we adopt the Language Server Protocol (LSP) developed by Microsoft, allowing us to encapsulate our software (the "language server") separately from the NEAMS Workbench (the text editor or "language client"). Owing to this generic approach, these editor services could also be deployed in any other editor supporting the LSP.

Likewise, as the underlying language artifacts (such as the parser, serializer, metamodel, etc.) are independent of the text editor, they can easily be re-purposed for future work. Useful examples include defining a standalone API, saving/loading decks to/from an XML (Metadata Interchange) serialization, or even translating decks to and from other similar DSMLs, such as those employed by Serpent [7] and KENO [8] (two similar Monte Carlo neutron transport solvers) or the Consortium for the Simulation of Light Water Reactor's (CASL's) Virtual Environment for Reactor Analysis (VERA) [9].

2.2. PROTEUS Input Files

Correspondingly, the user's primary interface with PROTEUS is the "driver" input file, which details runtime and output options for the solver. These options are specified in an ad hoc textual format, not recognized or supported by text editors. However, as the semantic data consists almost exclusively of key-value pairs, we simply redefine the PROTEUS input file using a new syntax.

Specifically, we employ the YAML (recursively, "YAML Ain't Markup Language") data serialization format, given the similar (yet standardized) syntax and widespread popularity. * An exemplary driver file specified in each format is given in Listing 1. Notably, the YAML format is nearly identical to the native format, such that established users should find no difficulty adopting the former.

```
! Specify quadrature
sn_type LEG-TCHEBY
skipsolve NO
bc_alias side_set_0000010 VOID
bc_alias side_set_0000020 VOID
```

(a) Native PROTEUS

```
# Specify quadrature
sn_type: LEG-TCHEBY
skipsolve: false
bc_aliases:
    side_set_0000010: VOID
    side_set_0000020: VOID
```

(b) YAML

Listing 1: PROTEUS driver formats

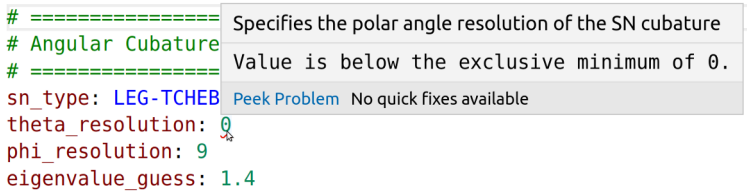
To validate PROTEUS driver files, we have developed an input schema, written in accordance with JSON Schema 7. This schema encodes documentation, default settings, and valid options, as specified in the PROTEUS user manual. As a benefit of YAML's popularity, we leverage an existing language server (supporting JSON Schema) developed by Red Hat [10] to provide editor services, as exemplified in Figures 1b and 3. Again, these editor services provide immediate validation (and assistance) in the text editor, at immense benefit to the productivity of end-users.

3. POST-PROCESSING AND VISUALIZATION

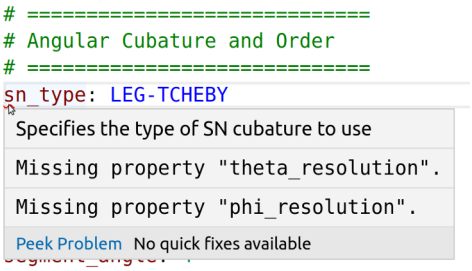
3.1. MCNP Textual Outputs

With the exception of unstructured mesh tallies, all MCNP results are output as plaintext files. To the detriment of end-users, all formats are custom, often forcing users to write ad hoc "text

*Note that as Javascript Object Notation (JSON) is a subset of YAML, it is likewise supported.



(a) Validating a range



(b) Detecting missing properties

Figure 3: Validation in the PROTEUS YAML driver

scraping" functions to programatically post-process their results. Conveniently, with the release of MCNP6.2, official parsers are now available as a standalone module, dubbed MCNPTools [11].

However, despite the tremendous utility of MCNPTools, it does not offer an alternative serialization format. Therefore, the results themselves are not rendered any easier to parse in the absence of MCNPTools. This is problematic, as it forces users to introduce a dependency on MCNPTools (an unfamiliar and restricted-distribution package) in all MCNP projects, in perpetuity.

To remedy this, we have defined Hierarchical Data Format 5 (HDF5) serialization formats for all MCNP outputs supported by MCNPTools. Unlike text files, HDF5 files are simple to access and manipulate, natively supporting metadata, hierarchical data, and multi-dimensional arrays. These files eliminate both the need for text files and purpose-built parsers (like MCNPTools) beyond the initial conversion to HDF5. Moreover, users may employ a familiar HDF5 API, rather than learning the specifics of MCNPTools (if they so choose). From Listings 2 and 3, we find the h5py (HDF5) interface at least as intuitive as that provided by MCNPTools.

```
from mcnptools import Mctal, MctalTally
mctal = Mctal('mctal')
tally4 = mctal.GetTally(4)
tally4_ebins = tally4.GetEBins()
tally4_vals = [tally4.GetValue(-1, -1, -1, -1, -1, -1, e, -1)
               for e in range(len(tally4_ebins))]
```

Listing 2: Reading a MCTAL tally to a Python list using MCNPTools

```
import numpy as np
import h5py as h5
mctal = h5.File('mctal.h5')
tally4 = mctal['4']
tally4_vals = np.squeeze(tally4['value'])
```

Listing 3: Reading a MCTAL tally to a NumPy array using h5py

3.2. MCNP Output Standardization

While comprehensive HDF5 output files for MCNP are advantageous (effectively replacing the textual formats), we now turn to a further goal: standardizing output formats across multiple Monte Carlo neutron solvers. Notably, OpenMC [12], an open-source Monte Carlo neutron transport solver, provides similar functionality to MCNP and already defines HDF5 output file formats. Therefore, we adopt (a subset of) these formats as initial "standard" formats, summarized in Table 1. †

Table 1: MCNP output files and their OpenMC analogs

MCNP Output	OpenMC Analog	Notes
MCTAL	State Point	Here, we focus on tallies and multiplication factors.
MESHTAL	Voxel Plot	MESHTAL tallies can also be represented in a State Point.
PTRAC	Track	-

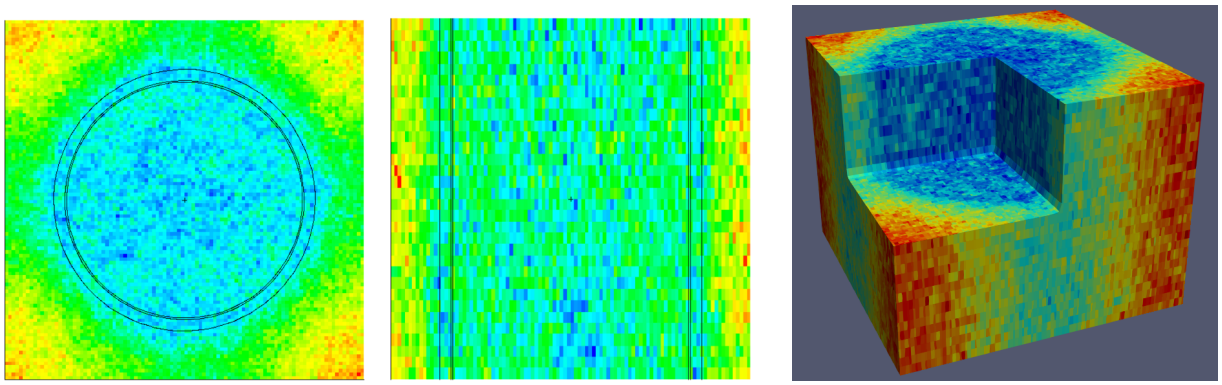
Beyond the convenience of standardization, by coercing (all available) MCNP output to the OpenMC formats, we enable the use of the OpenMC Python API for common post-processing. This is especially valuable for OpenMC "statepoints," which provide comprehensive facilities for tally arithmetic, and mesh tallies, which can be interrogated via a simple GUI provided by OpenMC.

To demonstrate this progress, we have converted a MCNP mesh tally from its native MESHTAL format (to a "MESHTAL.h5" file) into an OpenMC Voxel Plot file. With a utility included with OpenMC, the Voxel Plot can then be converted into a Visualization Toolkit (VTK) file which can be plotted in a variety of programs such as ParaView or VisIt. This provides the user with much more powerful viewing and post-processing capabilities than with the native MCNP utility (MCPLOT). Figure 4 shows a comparision of a MESHTAL file plot using MCPLOT (for the native MCNP file) and ParaView (for the VTK version).

3.3. PROTEUS Post-Processing and Visualization

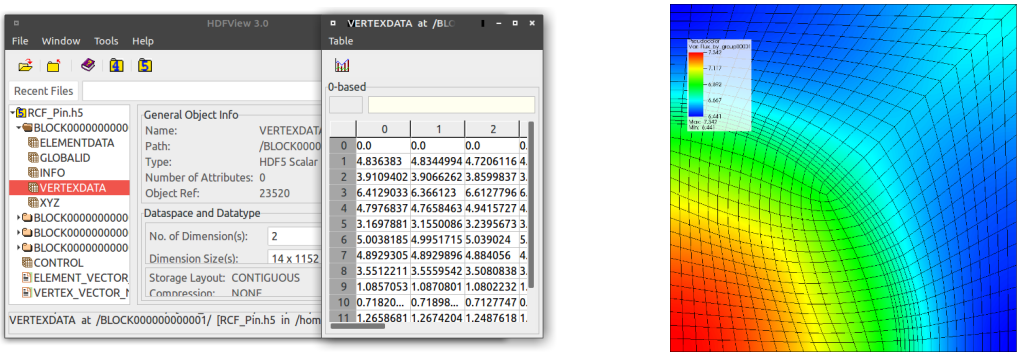
Nearly all PROTEUS output information is contained in "UNIC-"formatted HDF5 files describing the multi-group scalar flux distribution (over an unstructured mesh). As these files are already encoded in HDF5 (as was the goal for MCNP in Section 3.1) and supported by both VisIt and ParaView, we consider the established post-processing capabilities mature. To detail the current state of PROTEUS post-processing, we show in Figure 5 a graphical representation of the HDF5 output file (from HDFView) and a pseudocolor plot of scalar neutron flux rendered by VisIt.

†We expect more appropriate standard formats could be devised by comparing the common capabilities and output quantities of different solvers (such as MCNP, OpenMC, and Serpent). However, such a comparison is beyond the scope of this work.



(a) MCNP MCPLOT **(b) OpenMC Voxel Plot in ParaView**

Figure 4: Plotting MESHTAL files from native and OpenMC formats



(a) HDFView explorer **(b) VisIt plot**

Figure 5: Post-processing and visualization of a PROTEUS flux distribution

4. MODEL UNIFICATION

Crucially, MCNP6.2 and PROTEUS, as Monte Carlo and deterministic solvers, present two models of neutron transport with differing levels of fidelity. While Monte Carlo simulations model exact geometry with continuous-energy cross-sections, deterministic simulations consider a meshed or traced geometry with discrete angles and multigroup-collapsed cross-sections. This enhanced accuracy necessarily incurs additional computational cost (in terms of processors or runtime).

Naturally, users may want to employ multiple levels of fidelity for different tasks. For instance, Monte Carlo calculations may prove useful for validation or to collapse multi-group cross-sections, while determinstic methods would be preferred for transients, design studies, and uncertainty quantification. Therefore, the user should be empowered to compatibly simulate the same model (geometry, sources, and materials) in both MCNP6.2 and PROTEUS. Fortunately, both software support unstructured mesh geometry, suggesting a natural avenue for model unification. Specifically, we accomplish this by developing a utility which converts PROTEUS-compatible to MCNP-compatible meshes and templates a corresponding MCNP deck, depicted in Figure 6 [13].

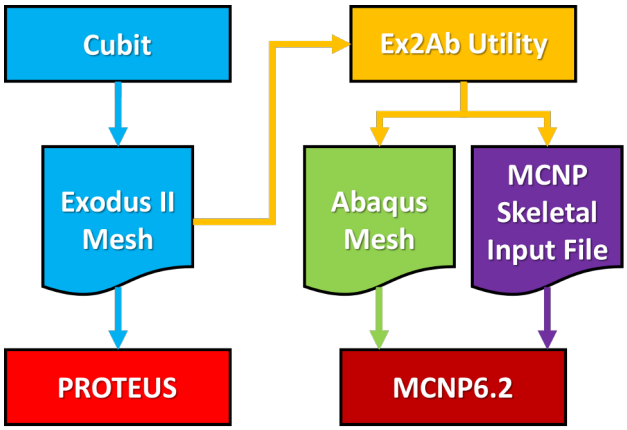


Figure 6: PROTEUS-MCNP model unification

4.1. Unstructured Mesh Conversion

While MCNP6.2 and PROTEUS can both transport particles over an unstructured mesh, MCNP requires meshes to be specified in a subset of the Abaqus ".inp" format, while PROTEUS mesh converters are only available for EXODUS-II meshes. Therefore, we have developed a mesh conversion utility which converts EXODUS-II meshes to an MCNP-intelligible format. Users may also here impose boundary conditions, to be encoded in the corresponding MCNP deck template.

4.2. Templating the MCNP Input Deck

As MCNP requires unstructured mesh geometry to be embedded within a Constructive Solid Geometry (CSG) cell, we additionally create a void region to encompass the user-provided mesh. Moreover, to impose reflecting boundaries on the meshed geometry (as MCNP does not permit

explicitly setting boundary conditions on meshed surfaces) we establish a reflecting CSG boundary at a minuscule offset from each reflective mesh boundary. Finally, we populate the MCNP deck with "pseudocells" corresponding to each element set of the Abaqus (originally EXODUS-II) mesh, such that the users may manually assign cell properties. If provided in the original mesh, pseudocell densities are also set.

Ultimately, this template defines an MCNP model accepting a mesh originally specified in a PROTEUS-compatible format. As such, this utility allows simulations in MCNP and PROTEUS on identical geometries, immensely streamlining hybrid Monte Carlo and deterministic workflows.

5. SCIENTIFIC WORKFLOWS

5.1. Variance Reduction via Consistent Adjoint-Driven Importance Sampling

Monte Carlo simulations are stymied by stochastic uncertainty. Namely, the convergence of the simulation in a region of interest is inversely proportional to the square root of number of histories N contributing to the response in that region. In analog Monte Carlo simulations (where a simulated particle is considered the literal analog of a physical particle) this presents an issue: stochastic uncertainty will be high in all regions where the (relative) neutron population is low.

We can remedy this by employing variance-reduction (VR) techniques. Specifically, our aim is to accelerate the convergence of a response of interest by maximizing the number of histories in that region. However, to do so, we must know whether or not a given particle is likely to contribute to our response of interest (e.g. by streaming to a given location or scattering to a specific energy group). Conveniently, this can be modeled via the adjoint neutron transport equation. While the forward transport equation models a neutron population arising from a given source, the adjoint solves the inverse problem; namely the contribution of a neutron population to a given response.

That being said, simulating the adjoint transport problem at the same fidelity as in the forward problem (that is, Monte Carlo) is likely to increase, rather than decrease, the computational effort. What would be advantageous is to approximate the solution of an adjoint problem, which can be used to cheaply accelerate the convergence of the high-fidelity forward problem. Thus, we find another virtue of having two compatible neutron transport simulators of differing fidelity.

5.1.1. Implementation in MCNP and PROTEUS(-SN)

Consistent Adjoint-Driven Importance Sampling (CADIS) is the means by which we use the adjoint flux to assign importances to the Monte Carlo phase space. To enact this strategy, we must first define an adjoint source, $q^\dagger(\boldsymbol{r}, \boldsymbol{\Omega}, E)$ which represents the response we seek to maximize. In the simplest case, this could be a delta function $q^\dagger = \delta(\boldsymbol{r}-\boldsymbol{r}_p, \boldsymbol{\Omega}-\boldsymbol{\Omega}_p, E-E_p)$, such that we minimize the variance at some point detector p. More generally, we often aim to obtain uniform convergence over the entire phase space. We can represent this using the response function $q^\dagger = 1/\psi$ (where ψ is the forward flux). This is referred to as Forward-Weighted- (FW-)CADIS, which requires an estimate of the forward flux (and is necessarily dependent on the forward problem definition).

Notably, while CADIS has previously been realized and implemented, the use of PROTEUS affords a unique opportunity to compute these low-order solutions on an unstructured mesh repre-

sentative of the true Monte Carlo geometry. This is a departure from implementations such as AutomateD VAriaNce reducTion Generator (ADVANTG) which homogenize materials within a Cartesian grid for use in structured mesh solvers. To avoid homogenization, importances are assigned for each MCNP cell of the model. This guarantees that each distinct material region of the model has its own importance. A single importance value is never assigned across portions of different cells as would occur with mesh-based importances. Using cells as the basis to assign importances also avoids introducing extra boundary crossings into the problem since cell boundaries will exist regardless of applying CADIS to the problem. The use of unstructured meshes can be further leveraged via our unstructured mesh conversion utility. This allows the user to create a single model that is used by both the deterministic and Monte Carlo parts of the simulation. Doing so simplifies model creation and ensures consistence between model features. Our example CADIS case utilizes the unstructured mesh workflow described previously.

5.1.2. Numerical Results

As a test case, we applied our CADIS implementation to a simple neutron shielding problem. The problem is defined within a 80 $\times$ 80 $\times$ 155 cm bounding box with vacuum boundary conditions on all sides. The quantity of interest is defined as the response in a 10 $\times$ 10 $\times$ 10 cm detector region (comprised of helium) on the opposite side of the shield from the source. For the forward source, a 1 MeV isotropic point neutron source is defined at the center of a 50 $\times$ 50 $\times$ 50 cm aluminum source container with a 25 $\times$ 25 $\times$ 25 cm cutout facing the shield. The shield is 80 $\times$ 80 $\times$ 10 cm and made of water. To provide PROTEUS with the necessary cross sections, this model was recreated and run in Serpent2 collapsing to a two-group structure (using a thermal neutron cutoff energy of 1 eV). Due to the shield being fairly substantial, very few particles will reach the detector and consequently variance will be high. This situation makes the problem a natural choice for variance reduction methods such as CADIS.

This geometry (as plotted by Cubit) is shown in Figure 7. In order to assign importances spatially, the problem is subdivided into many smaller cells rather than leaving large material regions as single cells. The individual cells are indicated by the colored blocks in the model. Using uniform blocks to construct the model was done for simplicity, but is not a constraint for MCNP or PROTEUS. Cells can be structured however the user chooses. The model also includes cells in the empty spaces (filled with helium), but these have been made invisible in order to view the important problem features.

The detector shielding problem was run using 1.0E+08 particle histories for both the analog and CADIS implementations and the response was measured using a F4 (flux) tally in the detector cell. Results from the simulations are given in Table 2 where the relative tally uncertainty and Figure of Merit (FOM) are given. Decreasing the tally uncertainty indicates a decreased variance and a higher confidence in the solution. However, it is more important to assess the competing metrics of variance (σ^2) reduction and increased runtime T as a FOM $1/(\sigma^2 T)$. If the FOM increases, it means that the tally uncertainty is reduced more efficiently by CADIS than by simply running the analog problem for more particle histories. Table 2 clearly shows the desired trends of decreased tally uncertainty and increased FOM. This shows that the CADIS implementation was successful with 121% of the FOM from the analog problem. The tally uncertainty decreased by 44%.

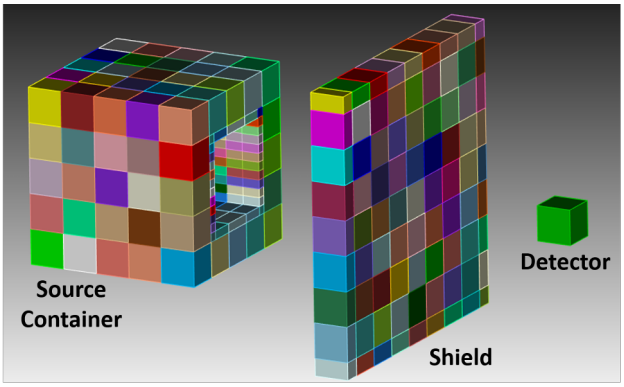


Figure 7: Detector shielding problem geometry

Table 2: Detector shielding problem with CADIS

	σ (Relative Tally Uncertainty)	FOM
Analog MCNP	5.00E-03	1.67E+00
CADIS MCNP	2.80E-03	3.68E+00

6. CONCLUSIONS

In conclusion, we have comprehensively integrated MCNP6.2 and PROTEUS into the NEAMS Workbench, offering Workbench users powerful reactor physics capabilities and MCNP and PROTEUS users an enhanced user interface through full-featured editor services. Meanwhile, by defining HDF5 formats for textual MCNP outputs, we relieve the user's dependence both on inscrutable text files and unfamiliar, purpose-built parsers, beyond an initial conversion to HDF5. We further coerce these MCNP outputs to "standard" formats described by OpenMC, granting users access to OpenMC utilities and facilitating common, comprehensive post-processing workflows.

For the compatible use of MCNP and PROTEUS, we have also developed a model unification utility capable of translating PROTEUS- to MCNP-compatible meshes, as well as generating a template MCNP deck to accept the mesh. Finally, we enact a hybrid Monte Carlo and deterministic workflow for CADIS which (novelly) computes the adjoint flux on unstructured meshes. For a neutron shielding problem, we find our implementation to be effective in applying cell-based weight windows to the MCNP model.

Overall, we find this effort dramatically empowers users of MCNP, PROTEUS, and Workbench, both with new compatible and cooperative modeling capabilities and intuitive, streamlined interfaces for input preparation and post-processing. Moreover, we expect the relevance of this work will only increase as additional simulation tools are "integrated" into the NEAMS Workbench.

ACKNOWLEDGEMENTS

This material is based upon work supported by the Department of Energy Office of Nuclear Energy under Award Number DE-NE0008707. This research was performed under appointment of the first author to the Rickover Fellowship Program in Nuclear Engineering sponsored by the Naval Reactors Division of the National Nuclear Security Administration. The authors would like to thank Robert Lefebvre, Chang-ho Lee, and Forrest Brown for their assistance and insights throughout the integration of MCNP and PROTEUS into the NEAMS Workbench.

REFERENCES

[1] B. T. Rearden, R. A. Lefebvre, B. R. Langley, A. B. Thompson, and J. P. Lefebvre. "NEAMS Workbench 1.0 Beta." Technical Report ORNL/TM-2018/752, Oak Ridge Natl. Lab. (2018).

[2] C. J. Werner. "MCNP Users Manual – Code Version 6.2." Technical Report LA-UR-17-29981, Los Alamos Natl. Lab. (2017).

[3] E. R. Shemon, M. A. Smith, and C. Lee. "PROTEUS-SN User Manual." Technical Report ANL/NE-14/6, Argonne Natl. Lab. (2016).

[4] K. A. Dominesey, M. D. Eklund, P. J. Kowal, and W. Ji. "Preliminary Integration of MCNP6 and PROTEUS into the NEAMS Workbench." *Trans Am Nucl Soc*, **volume 118**, pp. 962–965 (2018).

[5] J. C. Wagner and A. Haghighat. "Automated Variance Reduction of Monte Carlo Shielding Calculations Using the Discrete Ordinates Adjoint Function." *Nucl Sci and Eng*, **volume 128**(2), pp. 186–2018 (1997).

[6] K. A. Dominesey and W. Ji. "Editor Services for MCNP6 in the NEAMS Workbench." *Trans Am Nucl Soc*, **volume 119**, pp. 1098–1099 (2018).

[7] J. Leppänen, M. Pusa, T. Viitanen, V. Valtavirta, and T. Kaltiaisenaho. "The Serpent Monte Carlo Code: Status, Development and Applications in 2013." *Ann Nucl Energy*, **volume 82**, pp. 142–150 (2015).

[8] B. T. Rearden and M. A. Jessee. "SCALE Code System." Technical Report ORNL/TM–2005/39-V-6.2 (2016).

[9] J. A. Turner et al. "The Virtual Environment for Reactor Applications (VERA): Design and Architecture." *J Comput Phys*, **volume 326**, pp. 544–568 (2016).

[10] "YAML Language Server." (2019). URL https://github.com/redhat-developer/yaml-language-server.

[11] C. J. Solomon, C. Bates, and J. Kulesza. "The MCNPTools Package: Installation and Use." Technical Report LA-UR-17-21779, Los Alamos Natl. Lab. (2017).

[12] P. K. Romano et al. "OpenMC: A State-of-the-Art Monte Carlo Code for Research and Development." *Ann Nucl Energy*, **volume 82**, pp. 90–97 (2015).

[13] P. J. Kowal, K. A. Dominesey, M. N. Dupont, J. A. Eugenio, and W. Ji. "Unstructured Mesh Unification for MCNP6.2 and PROTEUS in NEAMS Workbench." *Trans Am Nucl Soc*, **volume 121**, pp. 1541–1544 (2019).