ORNL/TM-2016/579

WARTHOG: Progress on Coupling BISON and PROTEUS

Shane W. D. Hart

September 2016

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2016/579

Reactor and Nuclear Systems Division

WARTHOG: Progress on Coupling BISON and PROTEUS

Shane W. D. Hart

September 2016

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-Battelle, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

ACRONYMS

ANL	Argonne National Laboratory
DOE	U.S. Department of Energy
DTK	Data Transfer Kit
FPL	Fuels Product Line
FY	Fiscal Year
INL	Idaho National Laboratory
IPL	Integration Product Line
MOOSE	Multi-physics Object-Oriented Simulation Environment
NEAMS	Nuclear Energy Advanced Modeling and Simulation
ORNL	Oak Ridge National Laboratory
RPL	Reactor Product Line
SGAPI	Sub-Group Application Programming Interface

ACKNOWLEDGEMENTS

The author would like to acknowledge the previous work on Warthog performed by Alexander McCaskey. He also provided much needed support to ease the transition to get Warthog running. In addition, the support provided by Stuart Slattery in troubleshooting DataTransferKit has been greatly beneficial to this and previous efforts.

This work is supported by the U.S. Department of Energy's Nuclear Energy Advanced Modeling and Simulation (NEAMS) Program under the NEAMS Integration Product Line.

EXECUTIVE SUMMARY

The Nuclear Energy Advanced Modeling and Simulation (NEAMS) program from the Office of Nuclear Energy at the Department of Energy (DOE) provides a robust toolkit for modeling and simulation of current and future advanced nuclear reactor designs. This toolkit provides these technologies organized across product lines, with two divisions targeted at fuels and end-to-end reactor modeling, and a third for integration, coupling, and high-level workflow management. The Fuels Product Line (FPL) and the Reactor Product Line (RPL) provide advanced computational technologies that serve each respective field effectively. There is currently a lack of integration between the product lines impeding future improvements of simulation solution fidelity. In order to mix and match tools across the product lines, a new application called *Warthog* was produced. Warthog is built on the Multi-physics Object-Oriented Simulation Environment (MOOSE) framework developed at Idaho National Laboratory (INL).

This report details the continuing efforts to provide the Integration Product Line (IPL) with interoperability using the Warthog code. Currently this application strives to couple the BISON fuel performance application from the FPL using the PROTEUS Core Neutronics application from the RPL. Warthog leverages as much prior work from the NEAMS program as possible, enabling interoperability between the independently developed MOOSE and SHARP frameworks, and the libMesh and MOAB mesh data formats. Previous work performed on Warthog allowed it to couple a pin cell between the two codes. However, as the temperature changed due to the BISON calculation, the cross sections were not recalculated, leading to errors as the temperature got further away from the initial conditions. XSProc from the SCALE code suite was used to calculate the cross sections as needed.

The remainder of this report discusses the changes to Warthog to allow for the implementation of XSProc as an external code. It also discusses the changes made to Warthog to allow it to fit more cleanly into the *MultiApp* syntax of the MOOSE framework. The capabilities, design, and limitations of Warthog will be described, in addition to some of the test cases that were used to demonstrate the code. Future plans for Warthog will be discussed, including continuation of the modifications to the input and coupling to other SHARP codes such as Nek5000.

1. INTRODUCTION

The Nuclear Energy Advanced Modeling and Simulation (NEAMS) program from the Office of Nuclear Energy at the Department of Energy (DOE) has funded two comprehensive frameworks that developers can build on to provide advanced simulation technologies for various aspects of simulating a nuclear reactor. The Fuels Product Line (FPL) at Idaho National Laboratory (INL) has developed the Multi-physics Object-Oriented Simulation Environment (MOOSE) framework [1] upon which the fuel performance code BISON is based, while the Reactor Product Line (RPL) at Argonne National Laboratory (ANL) has developed the SHARP framework [2] consisting of various codes such as PROTEUS [3]. MOOSE focuses on providing a multi-physics framework which uses a “top-down” approach, while SHARP provides a multi-physics framework focused on “bottom-up” development. Both are valid approaches to multi-physics problems, but their indpendent development has resulted in framework components that are difficult to mix and match.

Previous work [4] was performed to develop a coupling code called *Warthog* to couple these two different frameworks together. Work focussed originally on coupling the fuel performance code BISON [5], which was built on the MOOSE framework, with the neutronics code PROTEUS, which is part of SHARP. This previous work was successful at coupling a simple pin cell between the codes, but due to the lack of temperature feedback in the cross section data provided, this coupling was not complete. Coupling between these codes is desired because BISON lacks high-fidelity fission source feedback, relying primarily on experimentally validated power profile piece-wise functions to stand in for a high-fidelity neutronics power calculation. Warthog’s goal is to enable interoperability of components from MOOSE/libMesh with components from SHARP/MOAB.

This report documents work that was performed to build on Warthog’s previous abilities and to enable it to perform full coupling with temperature feedback of the cross sections. This report also details the addition of the XSProc module from the SCALE [6] code suite as a Warthog external code. The report provides examples of current cases that were run, and it provides a discussion of the current coupling status. The report concludes with a look at future work to move Warthog into the MOOSE “herd” of codes and coupling it with other codes of the SHARP framework.

2. PREVIOUS WORK

Warthog was previously capable of performing some simple coupling between PROTEUS and BISON. When the code was handed off to the new maintainer, a basic coupling scheme was in effect. This section describes the previous work and the limitations that were present.

2.1 COUPLING

The Warthog code, as delivered to the new maintainer, had the ability to run PROTEUS alone or as a coupled run with BISON. However, because there was no way to calculate new cross sections based on temperatures calculated by BISON, minimal effort was placed into demonstrating coupling by the previous maintainer. In fact, when obtained, coupling did not work out of the box, and some slight code modifications were needed.

The heart of the coupling scheme involves using the Data Transfer Kit (DTK) to move data from the MOAB mesh used by PROTEUS to the libMesh used by MOOSE/BISON. The process is more fully described in previous Warthog documentation [4], and an overview of the process is given in Figure 1. In essence, a master Warthog MOOSE application sets some initial conditions (namely temperature) and moves those to the MOAB mesh for PROTEUS using DTK. PROTEUS then runs and calculates the power density on the MOAB mesh. This power density is then transferred back to the Warthog application using DTK where MOOSE handles transferring the required data to the BISON run using a MultiApp transfer. As mentioned previously, due to there being no temperature correction for the cross sections, every iteration of the coupled problem would use the same cross section data and the power density would never change.

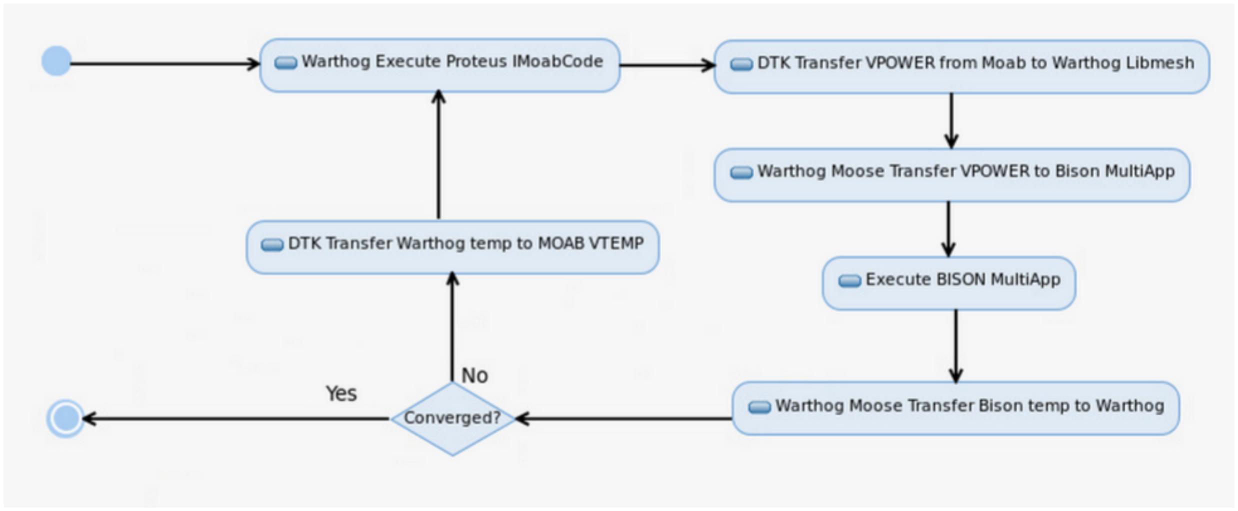

Warthog Execute Proteus IMoabCode
DTK Transfer VPOWER from Moab to Warthog Libmesh
Warthog Moose Transfer VPOWER to Bison MultiApp
DTK Transfer Warthog temp to MOAB VTEMP
Execute BISON MultiApp
No
Yes
Converged?
Warthog Moose Transfer Bison temp to Warthog

Figure 1. High level flow for a Warthog execution.

2.2 EXAMPLES

Several previous examples of how Warthog can be used given with the Warthog repository. These examples are only discussed here for completeness, as they were used as a base for the current work.

2.2.1 Simple Hexagonal Assembly (SAHEX) PROTEUS Example

This example demonstrates how Warthog can be used as a simple wrapper around a PROTEUS input. It exists as a proof-of-concept in the mesh transfer between the Warthog libMesh mesh and the PROTEUS MOAB mesh using DTK. The mesh used in the problem is shown in Figure 2. In this example, the initial conditions are all set on the PROTEUS input files, and the power density is calculated. The power density is returned to Warthog using DTK and output on the output mesh. With no coupling, these data are not passed to BISON, and the calculation can either continue (running the same PROTEUS problem again) or end based on user input. The power density is shown in Figure 3. Note that the mesh is hidden so that the power density is more easily seen.

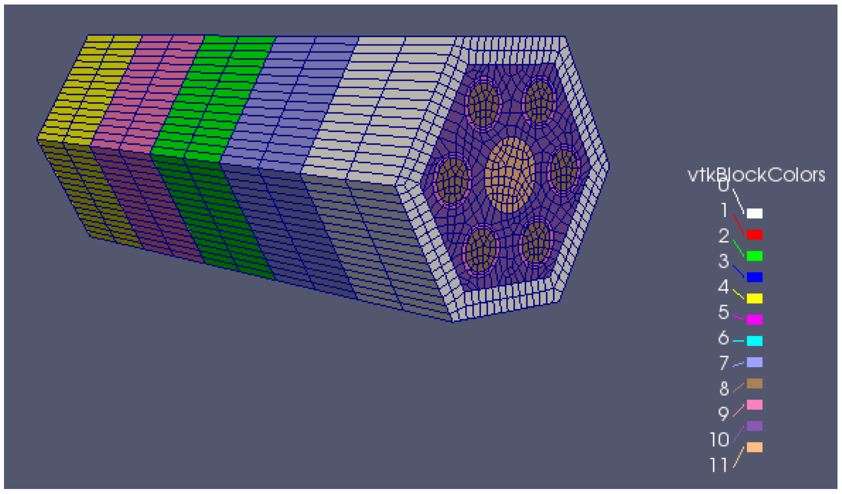


Figure 2. PROTEUS mesh for SAHEX example.

2.2.2 ReactorMesh Example

Some previous effort was made to simplify the creation of a fuel assembly by creating a custom mesh for MOOSE called a ReactorMesh. The ReactorMesh construct extends the FileMesh class of MOOSE with some additional parameters to specify the geometry of a reactor assembly. In particular, the user can specify the number of fuel rods in the x and y directions. By setting the number of fuel pins in each direction to 1, the user can mimic a normal single pin cell run. The ReactorMesh class simplifies set-up by automating creation of some of the MultiApp transfers from PROTEUS to BISON. The overarching goal is to allow the user to simulate the assembly as a whole in PROTEUS and then run each pin of the assembly in separate BISON MultiApp runs. Other convenience functions allow for the automatic creation of BISON and PROTEUS inputs based on a small number of template parameters given by the user in the input file.

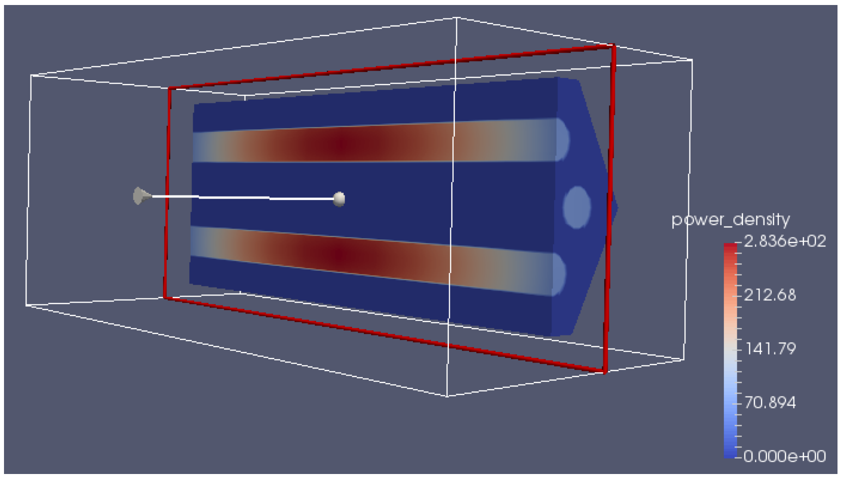


Figure 3. PROTEUS-calculated power density for SAHEX example.

Currently this work is incomplete and has only been tested briefly by the previous maintainer. This testing was done only on a 1 x 1 assembly. It is uncertain at this time if this work will be maintained. Since the framework is in place, it should be easily fixed as work on Warthog is completed.

3. CURRENT WORK

This section will discuss work that occurred during the past Fiscal Year (FY). Most work involved taking over maintenance of the code from the previous maintainer and becoming familiar with the Warthog and MOOSE suite of codes and tools. Some work had to be done to enable the coupling between PROTEUS and BISON, as this capability had declined slightly prior to the transition in maintainers. New work involved adding XSProc to the Warthog toolset and working on adding examples that show the coupling in action.

3.1 XSPROC INTEGRATION

One of the main goals for this FY was to enable cross section generation inline of Warthog. Multi-Group (MG) cross sections are required for the PROTEUS neutronics code. Cross sections can be given to PROTEUS through a ASCII text file in the ANLXS format, or as a binary ISOTXS-formatted file. In addition, previous versions of Warthog rely on the Sub-Group Application Programming Interface (SGAPI) packaged with some versions of PROTEUS to generate the self-shielded cross sections [7]. However, this approach had the same limitations as providing stand-alone cross section files: namely that it could not be modified as the temperature changed.

Based on familiarity with the SCALE code package, it was decided that XSProc would be used to generate the cross sections for PROTEUS. XSProc is a modern module in SCALE that uses AMPX [8] libraries to generate self-shielded cross sections for user-defined geometries, compositions, and temperatures. Unlike most of the legacy applications in SCALE, XSProc is written using C++.

XSProc was added to Warthog as a new ExternalCode. As with other ExternalCodes, the user controls the operation of XSProc through an options block in the Warthog input file. Currently the options present for an XSProc run are fairly basic. The user can set such parameters as lattice pitch, clad radius, fuel radius, and gap radius. Warthog constructs a dummy input file (in-memory) for XSProc and runs XSProc with that input. Typically this can be done all in memory, but PROTEUS does not use the same AMPX format that is used by SCALE codes. Therefore, the AMPX cross section library must be written to disk, and an external converter must be used to convert the cross section data to the ANLXS format. PROTEUS can then convert this ANLXS format to the ISOTXS format used by PROTEUS.

After every PROTEUS/BISON run, XSProc can then be re-run with the updated temperatures to calculate new cross sections. The overall flow of the process is shown in Figure 4.

3.2 COUPLING PIN CELL

Another goal for this FY was to demonstrate coupling of a pin cell between PROTEUS and BISON. As described above, in Section 2.2.1, Warthog had the ability to run PROTUES as a standalone code. Also, there was an automated mechanism in place to transfer the fission power solution to BISON as part of the ReactorMesh work (Section 2.2.2). However, to couple just a single pin cell, the ReactorMesh interface was not used. As this interface is still very much a work in progress, it was decided that for simplicity, the transfers would be set up manually in the Warthog input file.

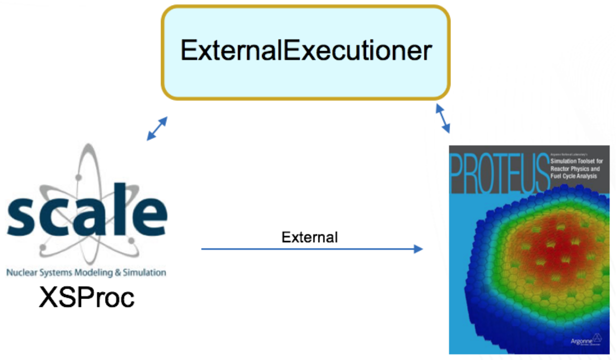


Figure 4. Flow of XSProc to PROTEUS execution.

The mesh used for this pin cell is shown in Figure 5. PROTEUS has support for both tetrahedral and quad meshes, as well as higher order elements. However, a relatively coarse tetrahedral mesh has been used. The figure shows is a full 3D mesh with a cut through the center of the pin cell. The fuel, cladding, and moderator material can all be clearly seen.

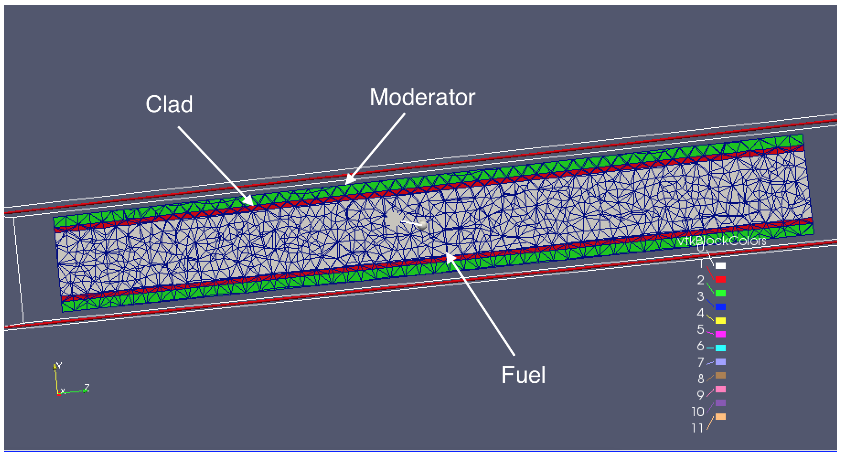
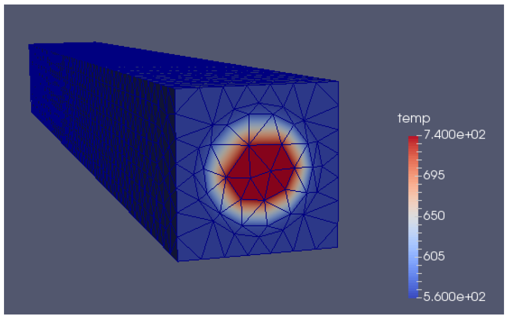


Figure 5. Mesh created in CUBIT for pincell coupling.

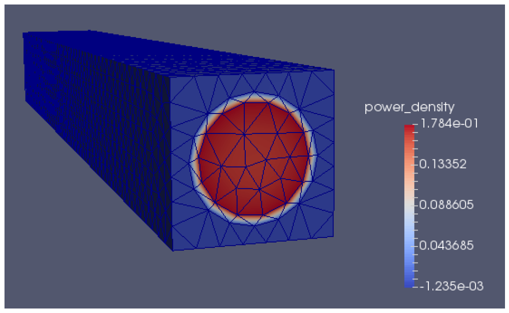
The input temperatures for the mesh are shown in Figure 6a, and the fission power density calculated by PROTEUS for this pin cell is shown in Figure 6b.

As shown in the figures above, the power density and the temperature are both highest in the fuel regions and lowest in the moderator. This is expected for this case. The power density can then be transferred to a BISON MultiApp case using the transfer blocks in the Warthog input file. A BISON mesh and input file that corresponds to the mesh used in PROTEUS must be given. Generally this is the same mesh as used for PROTEUS, but it is lacking the moderating material. BISON will solve for the temperature on the mesh, among other things. This temperature can then be used to run XSProc and then PROTEUS. The coupling can continue until some predefined end point is reached or until the temperature reaches a steady state.

Currently the main output desired from BISON is an updated temperature map. The temperatures on the mesh after one BISON run can be seen in Figure 7. As can be seen from the figure, the temperature has increased very slightly from this problems initial temperature of 500 C. This slight increase is due to the



(a) Initial temperature for pincell mesh.



(b) Calculated fission density for pincell mesh.

Figure 6. Pincell mesh data.

small time step taken. As the time from the initial start increases, the temperature should increase and eventually reach a steady state.

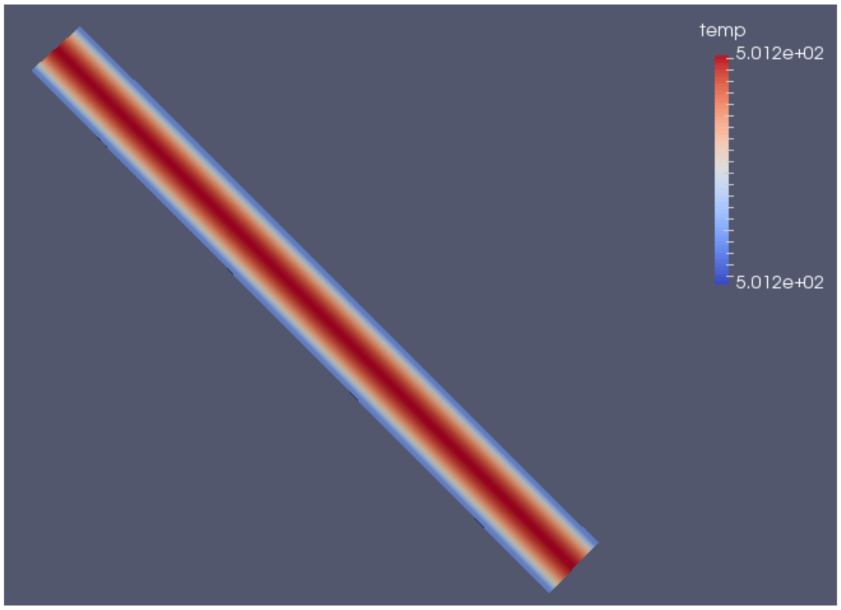


Figure 7. Temperature of pin cell after BISON run.

4. FUTURE WORK

Many avenues are available for future work including:

- expanding on the coupling interface (including the ReactorMesh) to allow more user options
- allowing more granular control of the XSProc execution sequence,
- looking into coupling 2D problems due to faster run time, and
- exploring coupling of other SHARP based codes such as Nek5000.

Currently the Warthog coupling structure is somewhat input intensive. The user must know the appropriate PROTEUS options and input, as well as the appropriate BISON input. It would be beneficial to make this more general, allowing the user to specify certain attributes that can be used to generate a mesh and the associated input as needed. This has been explored some with the ReactorMesh mesh facility inside of Warthog, but it must be developed further and made more accessible. Likewise, the XSProc module offers many options that are not presented to the user. Currently XSProc in Warthog assumes that the user wants cross sections processed for a square lattice with a cylinder fuel pin. XSProc supports many more options that could be useful in the future, such as different lattice types and different solver options. It is possible that the access to XSProc from outside Warthog must be revisited.

As shown in the examples discussed, all problems run at this stage have been 3D models. In the future, it may prove beneficial to move to simple 2D reflected models to decrease run time, particularly in the PROTEUS code. Some investigation has been done to explore 2D meshing, but there have been issues with getting PROTEUS to run successfully with most of the 2D meshes created. This problem must be investigated further, but it probably arises from the fact that there are many different meshing schemes used through all of the tools encompassed in this report.

Finally, in the coming year, coupling to other SHARP based codes will be examined. Nek5000 is the next target for coupling, and like PROTEUS, it uses the MOAB mesh format. Because MOAB is used for both, the framework for coupling should remain the same. Coupling Nek5000 with Warthog will hopefully be part of establishing the DTK transfers and providing users with options to control the simulation.

One task was discussed with the MOOSE team was to slightly reorganize the Warthog external executioners. Currently there is one Warthog “MasterApp” with all of the external executioners. This MasterApp transfers the power density solution to BISON. It was suggested that each external executioner be moved into its own “MultiApp.” This means that there would be a MultiApp for XSProc and a separate MultiApp for PROTEUS. This would allow MOOSE to handle the parallelism, and it would make Warthog fit more into the requirements for a MOOSE “animal.” This work will require some refactoring and movement of the code, but it should not take a long period of time.

5. REFERENCES

[1] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandie, “MOOSE: A parallel computational framework for coupled systems of nonlinear equations,” *Nuclear Engineering and Design*, 239(10), October 2009, 1768-1778.

[2] T. J. Tautges, et al., *SHARP Assembly-Scale Multiphysics Demonstration Simulations*, ANL/MCS-NE-13-9, March 2013.

[3] E. R. Shemon, M. A. Smith, and C. Lee, *PROTEUS-SN User Manual*, ANL/NE-14/6 (Rev 3.0), February 2016.

[4] A. J. McCaskey, S. Slattery, and J. J. Billings, *Warthog: A MOOSE-Based Application for the Direct Code Coupling of BISON and PROTEUS*, ORNL/TM-2015/532, September 2015.

[5] D. Perez, R. Williamson, S. Novascone, G. Pastore, J. Hales, and B. Spender, *Assessment of BISON: A Nuclear Fuel Performance Analysis Code*, INL/MIS-13-30314, 2013.

[6] S. M. Bowman, et al. “SCALE 6: Comprehensive Nuclear Safety Analysis Code System,” *Nuclear Technology*, 174, 126, 2011.

[7] C. H. Lee, A. Marin-Lafleche, and M. A. Smith, *Development of Cross Section Library and Application Programming Interface (API)*, ANL/NE-13/15, September 2013.

[8] M. Dunn, N. Greene, “AMPX-2000: A cross-section processing system for generating nuclear data for criticality safety applications”, *Transactions of the American Nuclear Society*, 86, 118-119, 2002.