ANL-ARC-290

End-User Evaluation of SHARP for Analysis of Sodium-Cooled Fast Reactors

Nuclear Engineering Division

About Argonne National Laboratory
Argonne is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC under contract DE-AC02-06CH11357. The Laboratory's main facility is outside Chicago, at 9700 South Cass Avenue, Argonne, Illinois 60439. For information about Argonne and its pioneering science and technology programs, see www.anl.gov.

DOCUMENT AVAILABILITY

Online Access: U.S. Department of Energy (DOE) reports produced after 1991 and a growing number of pre-1991 documents are available free via DOE's **SciTech Connect** (http://www.osti.gov/scitech/)

Reports not in digital format may be purchased by the public from the National Technical Information Service (NTIS):

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandra, VA 22312
www.ntis.gov
Phone: (800) 553-NTIS (6847) or (703) 605-6000
Fax: (703) 605-6900
Email: **orders@ntis.gov**

Reports not in digital format are available to DOE and DOE contractors from the Office of Scientific and Technical Information (OSTI):

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
www.osti.gov
Phone: (865) 576-8401
Fax: (865) 576-5728
Email: **reports@osti.gov**

Disclaimer
This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor UChicago Argonne, LLC, nor any of their employees or officers, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of document authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, Argonne National Laboratory, or UChicago Argonne, LLC.

End-User Evaluation of SHARP for Analysis of Sodium-Cooled Fast Reactors

prepared by
J.W. Thomas, H. Connaway, J. Grudzinski, Y. Tang, S.N.P. Vegendla
Nuclear Engineering Division, Argonne National Laboratory

September 30, 2014

SUMMARY

The DOE Nuclear Energy Advanced Modeling and Simulation (NEAMS) program is developing the SHARP toolkit to support growing reactor analysis needs in other programs. Although SHARP includes physics modules—especially the computational fluid dynamics code Nek5000 and the structural mechanics tools in Diablo—that have an existing user-base in other fields, most of the analysis and validation efforts for nuclear reactor applications have been performed by members of SHARP's own development team. The SHARP development team includes engineers and computational scientists from a diverse set of backgrounds, including nuclear engineering. However, before the SHARP tools can be adopted by the Advanced Reactor Technology (ART) program to support the analysis of sodium-cooled fast reactors (SFRs), it would be beneficial for an outside set of analysts and engineers to begin using SHARP and providing feedback to both programs. The authors of this report were charged with learning to use SHARP, applying SHARP to SFR analyses of interest to ART, evaluating the tools, and providing a set of recommendations and requirements to guide future development efforts by the NEAMS program. The authors are engineers (nuclear, mechanical, and civil) by training; all have experience in modeling and simulation for nuclear energy applications; all have either supported ART directly or report to a supervisor who does. This report is a formal summary of the findings of this team of engineers—the potential end-users of SHARP for ART activities. In an informal way, this effort also opened up communication lines between the SHARP developers and ART engineers, which will continue beyond the scope of this effort.

The selected application for SHARP is that of core structural deformation (i.e. radial expansion) caused by thermal gradients in the core during normal operating conditions, and the associated influence on core reactivity and the neutron flux distribution. SHARP's capabilities may be useful to support both quasi-static analysis needs for core restraint system design, and future needs for transient analysis of reactor safety performance. Because of careful design of the core restraint system, radial core expansion is often the dominant negative reactivity feedback mechanism in postulated fast transient events for SFRs under consideration by our program. Unfortunately, it is arguably the most difficult to predict accurately with conventional tools, and the available experimental dataset for validation is scarce. The modeling capabilities being developed under the NEAMS program may support ART's efforts to reduce capital costs in SFRs while maintaining safety performance, and to promote these design concepts with our international partners. To this end, the SHARP development team successfully performed a demonstration simulation featuring integrated neutronics, thermal-hydraulics, and structural mechanics simulation with sufficient geometric detail to represent the key phenomena. The ART end-users have been learning to apply the SHARP models themselves, and evaluate the SHARP team's efforts in this report.

Furthermore, the ART end-users were able to use the SHARP physics modules to develop and analyze models of other SFR applications and include this experience in their evaluation. The particular results of these simulations should be viewed as preliminary, and the authors understand that further work is required to improve the accuracy of the models. The results of SHARP predictions by the ART end-users reported here should in no way be used to evaluate the accuracy of the SHARP tools themselves. Rather, the preliminary results of these

exercises are included here to provide examples of ART end-users efforts to understand the SHARP tools.

The NEAMS developers have also been working on supporting elements intended to facilitate the process of developing new SHARP models and analyzing the predicted results. This includes the ORNL computing environment NiCE and the ANL mesh generation tool RGG, among others. These tools are also evaluated by the ART end-users in this report.

The report includes the ART end-users evaluation of the SHARP tools, including the SHARP team efforts on the core structural deformation demonstration problem, the SHARP neutronics module PROTEUS, the SHARP thermal-hydraulics module Nek5000, the SHARP structural mechanics module Diablo, as well as the supporting elements NiCE and RGG. Specific recommendations are provided for each code module in dedicated sections in this report. The report concludes with a summary of recommendations for the SHARP package.

Table of Contents

Summary i
Table of Contents iii
List of Figures v
List of Tables vi
1 Objectives and Scope 1
1.1 Technology Transfer 1
1.2 Collaborative Working Relationship 1
1.3 SHARP Workflow Improvements 3
2 Modeling Structural Core Deformation 4
2.1 Conventional Approach 4
2.1.1 Simulations for Core Restraint Design 4
2.1.2 Modeling Safety Performance 6
2.2 The SHARP Multi-Physics Approach 7
2.2.1 Background on SHARP 7
2.2.2 The SHARP Model of Core Deformation 9
2.3 Evaluation of the SHARP Team's Multi-Physics Demonstration 11
3 Evaluation of SHARP Neutronics (PROTEUS) 13
3.1 Analysis of EBR-II with PROTEUS 13
3.2 Feedback on Utility of PROTEUS for Reactor Analysis Needs 17
3.2.1 Ability to Meet User Needs 18
3.2.2 Documentation 19
3.2.3 General Usability 20
3.3 ART's Requirements for PROTEUS 21
4 SHARP Thermal-Hydraulics (Nek5000) 22
4.1 Overview of Ongoing Application of Nek5000 to Reactor Analysis 22
4.2 Feedback on Utility of Nek5000 for Reactor Analysis Needs 24
4.3 ART's Requirements for Nek5000 26
5 SHARP Structural Mechanics (Diablo) 27
5.1 Application to Soil-Structure Interaction Analysis 27
5.1.1 Introduction 27
5.1.2 System Considered 28
5.1.3 Modeling and Approach 29
5.1.4 Results 30
5.2 Feedback on Utility of Diablo for Reactor Analysis Needs 37
5.2.1 User Manual 38
5.2.2 Technical Support 38
5.2.3 Software 39
5.3 ART's Requirements for Diablo 39
6 Computing Environment for SHARP (NiCE) 41
6.1 NiCE for Nek5000 41
6.2 NiCE for PROTEUS 41
6.2.1 PROTEUS Model Builder 41

6.2.2 PROTEUS Launcher 43
6.3 NiCE for SHARP 44
6.3.1 SHARP Model Builder and Launcher 44
6.3.2 Reactor Analyzer and VisIt Interface 45
6.4 Documentation and Code Developer Support 46
6.5 ART's Requirements for NiCE 47
7 Mesh Generation Tools (RGG) 49
7.1 RGG and MeshKit 49
7.1.1 Documentation and Code Developer Support 49
7.1.2 Usability and Ability to Meet User Needs 51
7.2 RGG GUI 52
7.3 ART's Requirements for RGG 53
8 Summary of ART's Requirements for SHARP 54
9 References 57

LIST OF FIGURES

Figure 2-1 Limited Free Bow Core Restraint System 5
Figure 2-2 Illustration of Limited Free Bow Core Restraint Concept. 5
Figure 2-3 Power Distribution for the ABTR full core 10
Figure 2-4 k_{eff} as a Function of the Global Iteration 11
Figure 2-5 Magnified (100x) Displacements Colored by the Displacement in the *y*-direction 11
Figure 3-1 Examples of the PROTEUS EBR-II Assembly Mesh, (a) Axial Mesh and Material Regions and (b) Radial Mesh 14
Figure 3-2 PROTEUS Full-Core EBR-II Mesh 15
Figure 3-3 Power Distribution in the EBR-II Core at 55% Core Height, (a) DIF3D and (b) PROTEUS (Preliminary Results) 17
Figure 3-4 Group 8 (E = 302-498 keV) Flux Distribution in the EBR-II Core at 55% Core Height, (a) DIF3D and (b) PROTEUS (Preliminary Results) 17
Figure 4-1 Model (left) and Velocity Predictions (right) for the PLAJEST Benchmark (Preliminary Results) 22
Figure 4-2 Contour Plot of Predicted (i) Temperature, (ii) Vapor volume fraction and (iii) Velocity Magnitude Profile of Heated Flow in a Vertical Pipe (Preliminary Results) 23
Figure 4-3 Comparison of Experimental Measurements to Nek5000 Predictions using the Homogeneous Flow Model (Preliminary Results) 24
Figure 5-1 System Considered for Soil-Structure Interaction 28
Figure 5-2 FEM Model for Soil-Structure Interaction Test Problem 30
Figure 5-3 Filtered Time History of Input Displacement for *f*=1 Hz 35
Figure 5-4 Time History of Reaction Force for Square Footing with *B*=10 ft, *f*=1 Hz (Lysmer Boundary Condition) 35
Figure 5-5 Time History of Reaction Force for Square Footing with B=10 ft, f=33 Hz (Fixed Boundary Condition) 36
Figure 5-6 Time History of Reaction Force for Square Footing with B=10ft, f=33 Hz (Lysmer Boundary Condition) 36
Figure 5-7 Time History of Reaction Force for Square Footing with B=10ft, f=33 Hz (Fixed Boundary Condition) 37
Figure 6-1 PROTEUS Input Structure: (a) Example (Human-Created) Input File Distributed with PROTEUS, compared to (b) Example Input File Written by NiCE 42

LIST OF TABLES

Table 3-1 Calculated Eigenvalue of the EBR-II SHRT-45 Run 138B Core Loading (Preliminary Results) 16
Table 5-1 Vertical Impedence Function for Square Footing with *B*=10ft 32
Table 5-2 Vertical Impedence Function for Square Footing with *B*=20ft 33
Table 5-3 Comparison of Vertical Impedance Functions for Square Footings with B=10ft, L=200ft and Element Sizes of 5ft and 2.5 ft. 34

1 Objectives and Scope

The DOE-NE Advanced Reactor Technology (ART) program has supported an initiative with the following objectives:

1. To initiate transfer of available technology developed under the DOE-NE Nuclear Energy Advanced Modeling and Simulation (NEAMS) program to end users in the DOE-NE Advanced Reactor Technology (ART) program.
2. To establish a working relationship between advanced methodology developers working under the NEAMS program and end users of NEAMS products in the ARC program.
3. To initiate the development of the ARC specific workflow components and modules that facilitate adoption of the NEAMS Toolkit by ARC designers and analysts.

1.1 Technology Transfer

The complete SHARP [1] nuclear reactor modeling and simulation toolkit for analysis of nuclear reactors is now available to ART end-users at ANL, i.e. the team of engineers who support sodium-cooled fast reactor (SFR) analysis and design. The SHARP structural mechanics module Diablo [2], which was developed at LLNL primarily with NNSA support, was transferred to ANL for the purpose of this collaboration. This transfer required significant efforts in export control review, and technical efforts to install and test on ANL computing platforms. The SHARP neutronics module PROTEUS [3, 4] and thermal-hydraulics module Nek5000 [5] were both primarily developed at ANL, as were the tools to couple the codes under the MOAB [6] framework and to facilitate mesh generation with RGG [7].

In addition to providing access to the NEAMS tools themselves, this collaboration supported training for the ART end-users on employing the SHARP physics modules to SFR analyses. One-day training sessions were held for PROTEUS and Nek5000 in April and July 2014, respectively [8,9]. Each of these training sessions were attended by experts supporting ART and other reactor analysis programs. Because of the existing close interaction of the Diablo developers at LLNL and ANL engineers during the process where Diablo was exported to ANL, the ANL engineers decided that personal communication with the Diablo development team was preferable to such a one-day training session. Besides these training sessions, the SHARP development team provided continuous support to the ART end-users as they test out the tools. In addition, the collaboration helped incentivize the development of documentation, including revisions to the PROTEUS User's Manual and Methodology Manual in August 2014 [3, 4] and a March 2014 updated draft of the Diablo User's Manual [2].

1.2 Collaborative Working Relationship

To achieve the second goal, establishing the working relationship between developers and potential end-users, the decision was made to have the teams collaborate on a problem of interest to the ART program. The selected problem was the neutronics response to core geometry changes associated with structural deformation due to thermal expansion. For some SFR cores under consideration by the ART program, such core expansion provides the dominant negative reactivity feedback effect, helping to stabilize the reactor during postulated transient scenarios. In part due to a perceived lack of experimental evidence and reliable predictive methods, there is

considerable skepticism in the international community of the potential to take advantage of the core expansion phenomenon to improve reactor safety performance and reduce capital costs. For this reason, the ART program has been investing in recovering experimental data from the Fast Flux Test Facility (FFTF) that demonstrates the potential for core expansion to improve SFR safety performance. Meanwhile, the NEAMS program has been considering the core expansion problem as one of the primary drivers for developing the multi-physics modeling capability in SHARP for the last couple of years, and the tools have recently reached a level of maturity to attempt a meaningful demonstration.

The SHARP developers and ART end-users met to determine which particular problem would be analyzed, and the decision was reached to analyze core expansion in the Advanced Burner Test Reactor (ABTR) [10]. The ABTR was a prototype SFR concept developed by the ART program (and its predecessors), but its design is no longer actively supported. However, it is appealing for several reasons: (1) it employs the limited free bow core restraint system which provides the desired structural deformation, (2) its design is available in open literature which facilitates publication and possible future collaborations, and (3) the core size is comparable to other prototypic concepts that are under consideration, which facilitates estimations of SHARP's computational expense for these analyses. Analyzing the FFTF was not feasible because the data is not yet available. Analyzing the EBR-II would not be of interest, because it did not employ the limited-free bow core restraint system, and thus deforms differently than what is hoped for modern SFR concepts. Furthermore, the simulations performed here were first-of-a-kind and the objective is to perform a preliminary demonstration of the feasibility of continuum-scale explicit multi-physics treatment of the core expansion phenomenon with SHARP. Performing a validation exercise is well beyond the scope of this effort, and there wasconcern that modeling FFTF or EBR-II would provide a false impression of this project's scope.

The SHARP and ART teams collaborated to develop a problem specification for the ABTR core expansion problem. These specifications were actually completed in mid-October 2013, but were reported to DOE-NE in January 2014 [11]. Because this was the first demonstration exercise of its kind, the SHARP development team was responsible for performing the demonstration exercise. For testing and debugging purposes, the SHARP team began with a 7-assembly test problem. Clearly the 7-assembly problem is not representative of actual core physics (the neutron leakage and flux gradients are far too high), but the model enabled them to perform relatively inexpensive simulations while testing the mult-physics simulation capability. In the January 2014 [12] milestone report for this effort, successful tests of the 7-assembly model were described, single-physics models for each of the SHARP components (PROTEUS, Nek5000, and Diablo) of the full 199-assembly ABTR were described, but coupled SHARP multi-physics simulations of the 199-assembly ABTR had yet to be performed. Therefore, this effort continued under NEAMS support, and successful tests of the SHARP multi-physics capability were presented in a follow-up report [13].

Following the demonstration effort, the SHARP team transferred the codes and documentation to the ART end-users, and provided the previously mentioned training. Although the SHARP team successfully demonstrated SHARP's unique capability to explicitly predict structural deformation and neutronics response, the procedure was not sufficiently automated until late in the evaluation period. This made it impractical for the ART end-users to perform meaningful coupled SHARP simulations on their own. Therefore, the scope of the evaluation was broadened to include use of each of the SHARP physics modules—PROTEUS, Nek5000, and

Diablo—as standalone physics codes. For each of the three physics modules, one or two field experts were selected to begin using the SHARP tools and evaluate them in the context of their utility for SFR applications of interest to the ART program. These field experts are the co-authors of this report: H. Connaway for the PROTEUS code; S.V.P. Vegendla and Y. Yu for the Nek5000 code; and J. Grudzinski and Y. Tang for the Diablo code. Each of these engineers were able to work with the SHARP development team to begin learning the codes, perform some simulations, and provide an evaluation in this report. Because the Nek5000 code is already actively being used by ANL nuclear engineers, this effort focused primarily on evaluating the neutronics and structural mechanics modules, and the technical efforts described for Nek5000 applications are supported by other programs.

1.3 SHARP Workflow Improvements

The NEAMS program sponsors the development of supporting elements to facilitate model development and analysis, including the Reactor Geometry (and Mesh) Generator (RGG) [7], the NEAMS Integrated Computing Environment (NiCE) [14], and the VisIt [15] analysis and visualization tool [15], among others. The SHARP team employed RGG to generate the mesh for the ABTR, as discussed in the prior report [12]. An RGG training session was provided to the ART end-users, which highlighted the new graphical user interface developed as part of the SBIR with KitWare. The ART end-users made efforts to employ RGG to generate meshes for SHARP, and an evaluation of RGG is included in this report.

As part of this collaboration, an effort was initiated to develop a NiCE computing environment for SFR applications. Previous efforts in NiCE were focused on the Cartesian core geometries that are characteristic of light-water reactors, and were originally developed for computational tools developed by the CASL program. The NiCE development effort is centered at ORNL. During the early stages of this project, a set of SFR-specific requirements were provided by ANL to ORNL, and the ORNL development team began developing tools to facilitate PROTEUS model development and analysis and Nek5000 mesh generation. Following early stages of evaluation of the SFR-specific module in NiCE, the ART end-user team visited ORNL in April 2014 to provide feedback. The development efforts were reported by ORNL in December 2013 with revisions (including the feedback from the visit) in April 2014 [16]. The ART end-user evaluation is provided in this report.

2 Modeling Structural Core Deformation

This section provides background information on how ART engineers currently use conventional simulation tools to model structural deformation in SFR cores, and compares this with the approach proposed by the SHARP team. The section also includes background information on the core expansion phenomena, the behavior of core restraint systems that ART engineers design, and the SHARP tools themselves. Because the core structural deformation is an inherently multi-physics phenomenon, the simulation tools for modeling structural deformation have been multi-physics tools for decades. The primary contrast is that the SHARP approach provides a multi-physics simulation where all physics are considering the same geometry at the same time, and highly-resolved solution data is shared between codes dynamically throughout the calculation procedure. Conventional approaches tend to employ a multi-stage solution procedure that treats the interaction between the physics in an approximate sense. For instance, perturbation analysis includes a series of neutronics simulations—with assumed and typically uniform temperature profiles and an undeformed geometry—which generate a set of reactivity coefficients that represent the core's sensitivity to changes in geometry and thermal conditions. This is followed by subsequent calculations that predict changes in temperature and core geometry, and essentially multiply these predicted changes by the coefficients in the prior neutronics perturbation analysis. Information from the latter stages of the simulation may update the earlier stages in follow-up simulations, but iterations between the simulation stages are typically quite limited. The SHARP approach eliminates the need for generation of sensitivity coefficients, and the solution data from one physics module is applied directly to another physics module on the same geometric representation. These computing approaches are described in more detail in this section.

2.1 Conventional Approach

SFR analyses may include modeling structural deformations for several purposes, but the primary considerations are for: (1) the design of the core restraint system itself, and (2) analyzing the safety response of the reactor during postulated transient events. Different approaches and different codes are used to model structural deformations for these two applications, and they are both discussed in this section.

2.1.1 Simulations for Core Restraint Design

SFR core designs under consideration by ART feature core restraint systems that are designed to induce limited free bow [17]. The restraint system is characterized by top load pads (TLPs) on the assembly ducts at the top and above-core load pads (ACLPs) in the region above the core, along with restraining rings at the TLP and ACLP axial heights. The rigid restraint rings are attached to the core barrel at the ACLP and TLP locations. The load pads serve as preferential contact points between the ducts. The load pads add only marginal thickness to the main duct body but they are nonetheless thick enough to maintain the desired form under the design loadings. The limited free bow core restraint system is designed to provide inherent protection against over power events by taking advantage of thermally induced bending action of the fuel ducts. This is illustrated in Figure 2-2 which shows a row of three cantilevered ducts located symmetrically about the center of a core and in a radially varying thermal gradient.

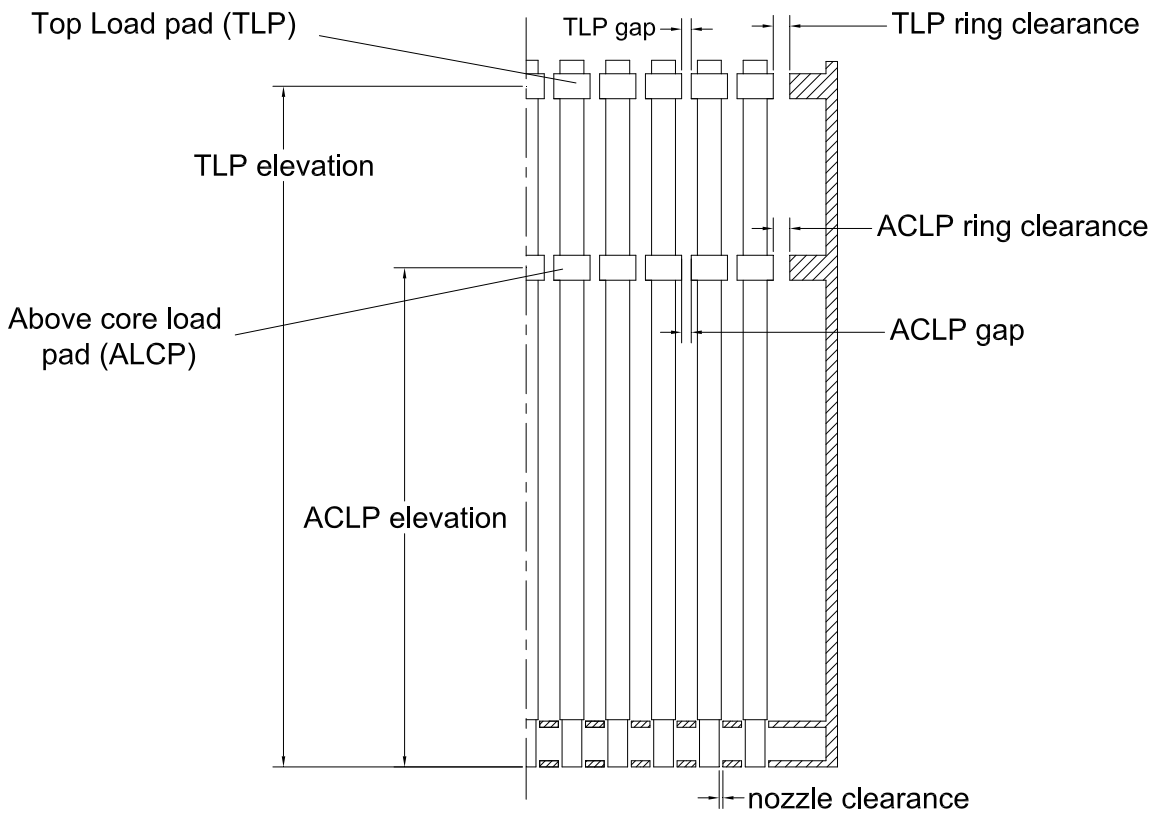


Figure 2-1 Limited Free Bow Core Restraint System

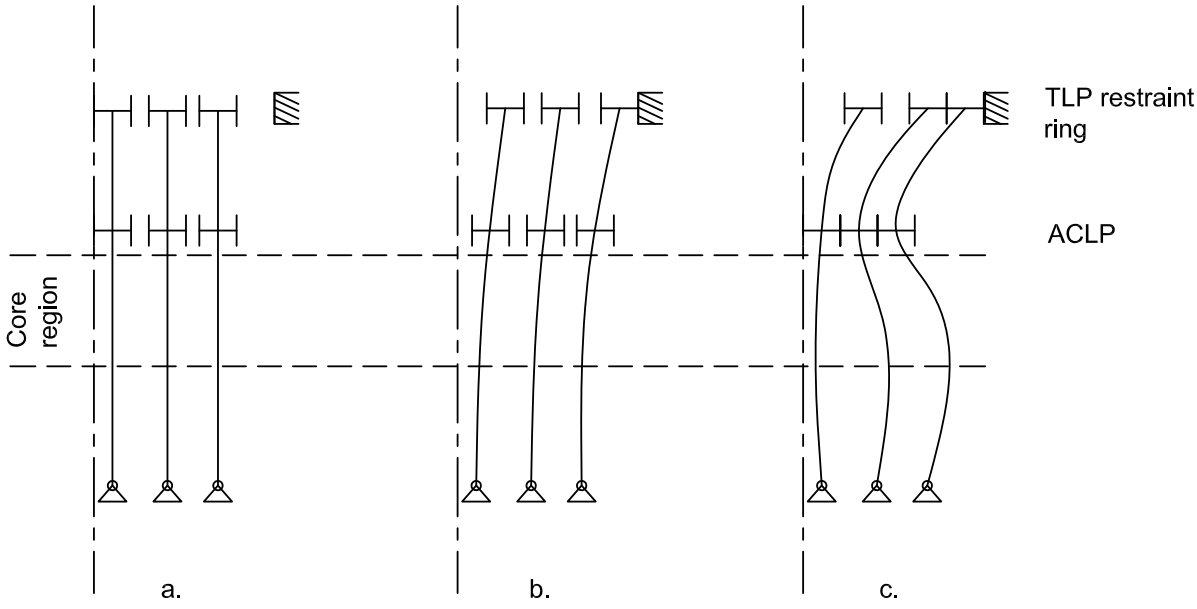


Figure 2-2 Illustration of Limited Free Bow Core Restraint Concept.

Figure 2-2a shows the nominal configuration of the ducts with no temperature gradient. As the radial thermal gradient develops (increasing temperature as distance from centerline decreases), the ducts begin to bow outward as shown in Figure 2-2b. Prior to contact with the top core restraint ring, the duct bends away from the core centerline as the temperature increases and therefore reduces the reactivity insertion. After contacting the top restraint ring and as the temperature gradient increases, the center of the duct bows inward which temporarily increases the reactivity. As the gradient increases, the inward bowing continues until the ducts contact at the ACLP. When the interior ducts all contact at the ACLP, the reactor is 'locked-up' and no further compaction can occur. Subsequent increased thermal gradients cause a reverse bowing below the ACLP moving the core region away from the core center as illustrated in Figure 2-2c. At this point the reactivity generally decreases with constant negative slope as temperature increases. The core restraint system is designed to have this lock-up occur below the nominal operating core outlet temperature.

For the specific problem of fast reactor core restraint design, ANL engineers use the NUBOW software code which is a legacy fast reactor code developed at ANL in 1974 and updated at various points. The objectives of these NUBOW simulations are two ensure that the core restraint system meets the following design objectives [17]:

1. Control core motion such that it is predictable and safe during transients and steady stage conditions. Reactivity insertion due to assembly bowing during transient over-power events must be negative, i.e. must help stabilize the core power.
2. Nominal refueling loads should be below a specified threshold.
3. Core alignment must be maintained to accommodate control rod movements.
4. Keep core assemblies within structural limits.

To accomplish this, NUBOW uses a characteristic fast reactor core description (number of rows, duct dimensions, etc.) and then creates the model (mesh) and predicts the bowing motion of the assemblies using beam elements to model the duct. The analysis includes contact between ducts and incorporates inelastic strains resulting from irradiation creep and swelling. As input, NUBOW uses core temperatures and flux distributions obtained from prior simulations of other fast reactor physics analysis tools. To ensure negative reactivity feedback (objective #1 above), NUBOW includes a post-processing capability to evaluate the reactivity change that results from bowing of the duct assemblies using displacement reactivity worths as an input. These displacement reactivity worths are determined from a prior perturbation analysis with a neutronics code, and provided as input to NUBOW. This displacement reactivity worth calculation procedure lacks a consistent set of best practices, the resulting reactivity estimates may also have significant uncertainty.

2.1.2 Modeling Safety Performance

To design an inherently safe fast reactor, reactivity dependence on radial core expansion must be engineered into the reactor plant to assure a loss of reactivity during transient events. In the advanced SFR concepts currently under consideration by the ART program, the core is designed to bow outward in response to thermal expansion of the structures in any transient where the core is heating. The grid plate and load pads, which support the core from below and restrain it from the top, respectively, also expand outward. Moreover, the core restraint system is designed such

that the fuel assemblies bow outward in the middle, further separating the fuel pins. When controlled correctly, core expansion causes the fuel assemblies to move farther apart from each other, which has a negative reactivity effect and helps to shut down the reactor. Simulation of this expansion, which is essential to the safety of these reactor concepts, necessitates the coupling of structural mechanics, thermal hydraulics, and structural mechanics.

For safety performance analyses supported by the ART program, the radial core expansion phenomena would typically be performed by a multi-step procedure. First, a neutronics perturbation analysis would be performed. The approach here is to analyze the core under two conditions—a reference condition and an expanded condition. The complex geometry of the expanded core cannot be modeled by conventional deterministic neutronics codes; the postulated expansion must be uniform everywhere in the core. In fact, the expansion must be uniform both radially and axially, i.e., all assemblies expand by the same amount, and the deformation does not vary along the length of the assembly. The perturbation analysis on this simplified geometry produces either a single reactivity coefficient to represent the integral response of the core to deformation, or a set of reactivity coefficients to represent expansion along the length. These coefficients are then employed in a separate transient analysis with the system code SAS4A/SASSYS-1 [18]. The system code predicts the core temperature during the postulated transient scenario. SAS4A/SASSYS-1 has two models for predicting changes in radial core geometry due to transient thermal expansion: the “simple” radial core expansion that assumes a linear function for the radial deflection, and the “detailed” radial core expansion model that treats the core as a single beam and permits a step-function axial profile for the radial deflection. Neither of the system code models are capable of modeling assembly bowing, but only include the effect core flowering due to expansion of the load pads and the supporting grid plate. Neglecting assembly bowing should be a conservative approximation, as the prior core restraint design analysis (described above) should have indicated that the assembly bowing would be a negative contribution to the core reactivity.

2.2 The SHARP Multi-Physics Approach

2.2.1 Background on SHARP

SHARP [1], developed under the NEAMS program, is an advanced modeling and simulation toolkit for the analysis of nuclear reactors. SHARP is comprised of several components, including physical modeling tools, tools to integrate the physics codes for multi-physics analyses, and a set of tools to couple the codes within the MOAB [6] framework. Physics modules currently include the PROTEUS [3] neutronics code, the Nek5000 [5] thermal-hydraulics code, and the Diablo [2] structural mechanics code. Development efforts in each of these physics modules were initiated by other programs prior to NEAMS existence (although PROTEUS was quite immature), which allows NEAMS to leverage prior investments. Nek5000 and Diablo were each originally developed for other applications, with sponsorship from DOE Office of Science and the NNSA, respectively, each with 20-30 staff-years of investment. The development philosophy for the physics modules is to incorporate as much fundamental physics as possible, rather than developing tools for specific reactor analysis applications. This empowers designers to analyze transformative reactor concepts with simulation tools that are not limited to available experimental data sets from currently existing reactor designs. By developing the tools to be highly efficient on parallel computing platforms, employing millions of processor cores, engineering-scale simulations become practical on high-performance computers currently

available at the DOE complex. Because of this foundation of high-performance computing, the SHARP tools will grow increasingly practical, rather than increasingly obsolete, as computing power continues to grow rapidly. Development efforts strive to work in tandem with efforts in experimentation, so that the tools are validated to produce accurate results for modeling physical phenomena that have been identified as important for nuclear reactor analysis. By taking this approach, SHARP supports nuclear reactor analysis and design activities for DOE programs and industrial partnerships with trustworthy modeling and simulation tools.

PROTEUS is a high-fidelity deterministic neutron transport code based on the second-order even-parity formulation. The application scope targeted for PROTEUS ranges from the homogenized assembly approaches prevalent in current reactor analysis methodologies to explicit geometry approaches, with the ability to perform coupled calculations to thermal-hydraulics and structural mechanics. The PROTEUS solver has a proven capability of using existing petascale parallel machines to solve problems with demonstrated scalability of over 70% (strong scaling) at over 250,000 processors. These achievements of PROTEUS were made possible by partitioning the space-angle system of equations over the available processors and utilizing established iterative solution techniques from the neutron transport community combined with the parallel algorithms in the PETSc toolbox.

The Nek5000 computational fluid dynamics solvers are based on the spectral element method. Nek5000 supports two different formulations for spatial and temporal discretization of the Navier-Stokes equations. The first is the P_N-P_{N-2} method with velocity/pressure spaces based on tensor-product polynomials of degree N and $N-2$ respectively. The second is a low-Mach number formulation that uses consistent order-N approximation spaces for both the velocity and pressure. The Nek5000 code has been extensively verified and validated for several benchmark problems and has a proven scalability in existing petascale architectures up to 131,072 processors (over a billion degrees-of-freedom).The conjugate heat transfer problems that are typically present in nuclear engineering applications can be solved rigorously using the formulations in Nek5000.

In order to perform thermofluid analyses of homogenized fuel assemblies for the core expansion problem, a porous media model was implemented in the Nek5000 code. Porous media models are typically applied to problems where the fluid flows through a region with many small-scale solid structures, and it would be impractical to resolve the geometry explicitly. Instead, the effect of the small-scale solid structures on the flow is modeled as a momentum sink or resistance in a homogenized fluid domain. In this particular case, we wish to model the influence of the fuel pins on the flow, i.e., drag and pressure drop, as a momentum sink without explicitly representing the geometry of thousands of fuel pins. The model must also account for the energy deposition associated with nuclear fission. Moreover, fuel and cladding temperatures are estimated for each fuel assembly, and may be provided to the neutronics code in future coupled simulations. This enables prediction of the duct wall temperature profile, which controls the thermally-driven structural deformation in the core, with significantly lower computational expense than explicit representation of the wire-wrap fuel bundle geometry.

The Diablo code being developed at Lawrence Livermore National Laboratory uses implicit, Lagrangian finite-element methods (FEM) for the simulation of solid mechanics and multi-physics events over moderate to long time frames. A primary focus is nonlinear structural mechanics and heat transfer. The code provides a venue for applying parallel computation to discretization technologies developed and user-tested in the legacy serial-processor codes NIKE3D and TOPAZ3D. Diablo is built around Fortran 95 data structure objects and a message-

passing programming model. The architecture provides flexibility for the addition of other field problems, such as electromagnetics.

In structural analysis of mechanical assemblies, a key functionality is "contact": capturing the interaction between unbonded material interfaces. The Diablo team has broad experience with contact problems and has created state-of-the-art algorithms for their solution. Their experience with contact motivates the use of low-order spatial discretizations, such as eight-node hexahedra for continua and four-node quadrilaterals for shells. Appropriate formulations are employed to accommodate nearly incompressible material models, such as for metal plasticity and rubber elasticity. Global algorithms include second-order and quasi-steady time integration and a number of approaches for nonlinear iteration: full Newton, modified-Newton, multiple quasi-Newton updates, and line search. Linear solvers are utilized from multiple libraries.

2.2.2 The SHARP Model of Core Deformation

The details of the SHARP model of the core deformation of the ABTR core are described in prior DOE reports [12, 13], and so are just summarized here. The simulations were preliminary, and were intended to demonstrate the capabilities of the SHARP codes to simulate complex multi-physics response. The simulation could be considered to be a quasi-static analysis to determine the initial geometry of the fresh core at the hot operating condition. Rather than going through the complex time-dependent start-up procedure with the required control rod motion, the core configuration is fixed as the inlet flow and power are at the nominal operating conditions throughout the simulation. The core power is fixed throughout, and the fission source is adjusted by k_{eff} (which deviates from unity). The strategy can be summarized as follows:

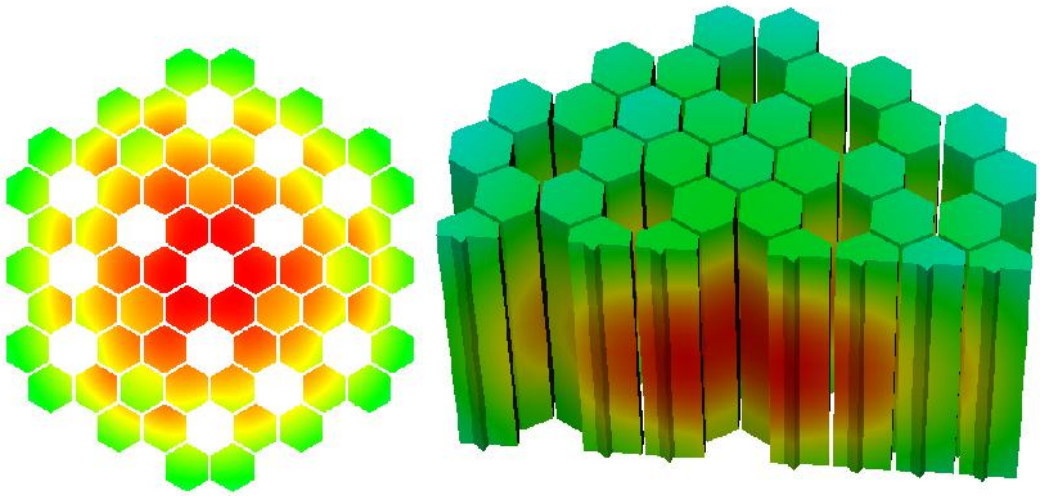
1. At each global iteration of the three physics (SHARP Global iteration), the mesh is updated and the input files are modified with corrected densities.
2. As part of each global iteration, 50 inner iterations are performed between PROTEUS and Nek5000 to converge the neutronics and thermal-hydraulics fields.

The full core ABTR problem has 199 assemblies in total, including 60 fuel assemblies. The total power is set to 250 MWt based on the specification. The fuel assemblies comprise three different types: inner core, outer core, and fuel test assemblies, which differ only by fuel composition.

The full core mesh for PROTEUS has 825k vertices and nearly 800k elements. Combined with 48 modeled angles in PROTEUS and 9 energy groups, the neutronics model has 356.5 million space-angle-energy degrees of freedom. The Nek5000 mesh employs the same 800k quadratic elements with a polynomial order of N=2. The Diablo mesh includes only the solid ducts, load pads, and restraint rings, because the contribution of the fuel pins to resisting structural deformation is negligible. Both the PROTEUS neutronics and Nek5000 thermal-hydraulics models treat the interior of the fuel assembly—which includes the fuel pins and interstitial coolant—as a homogenized media, with Nek5000 employing the porous medium model. Bypass flow between adjacent assemblies is not treated; rather, these regions are represented by solids.

Figure 2-3 depicts the power distribution in the fuel assemblies of the full-core ABTR model from the initial PROTEUS simulation. Figure 2-3a in particular shows the 3D distribution in the entire active core region sectioned along the core centerline to expose interior features. The power distribution is peaked at the radial and axial center of the core, and the z-dependence appears to

be approximately cosine shaped. Figure 2-3b highlights the radial power pattern at the axial midplane of the fuel. The PROTEUS calculation does not account for gamma heating and therefore has negligible power deposition predictions in the outer regions (reflector and shield). To account for these effects, power in these regions has been added based on previous calculations for the ABTR preconceptual design [10].



(a) 3D View (b) Core Mid-Plane

Figure 2-3 Power Distribution for the ABTR full core

The resulting prediction for the core displacements computed by Diablo in the *x*- and *y*-directions is shown in Figure 2-5. Figure 4.7 shows the evolution of the k_{eff} as a function of the global iteration. The result should be considered as preliminary as only three iterations have yet been conducted. Each result point shown in Fig. 4.6 represents the result of a fully converged Nek5000-PROTEUS iteration (50 inner iterations).

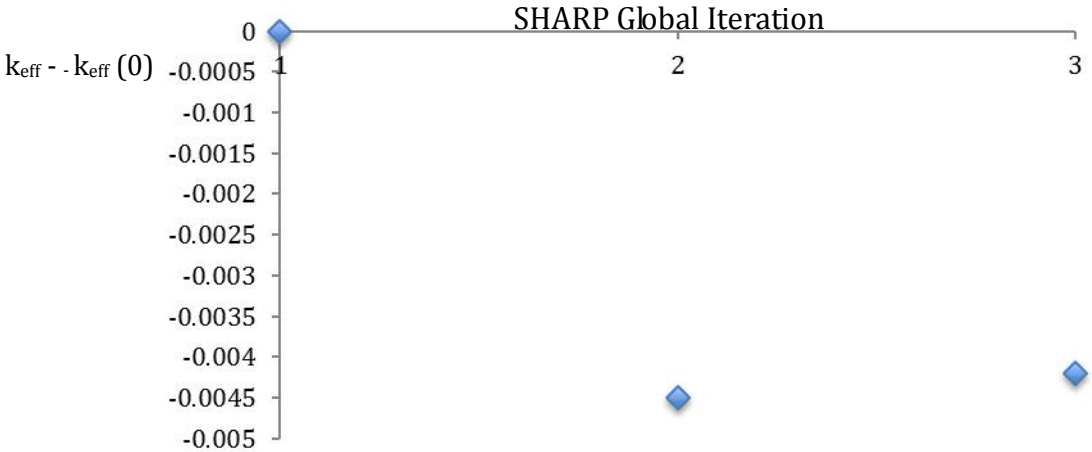


Figure 2-4 k_{eff} as a Function of the Global Iteration

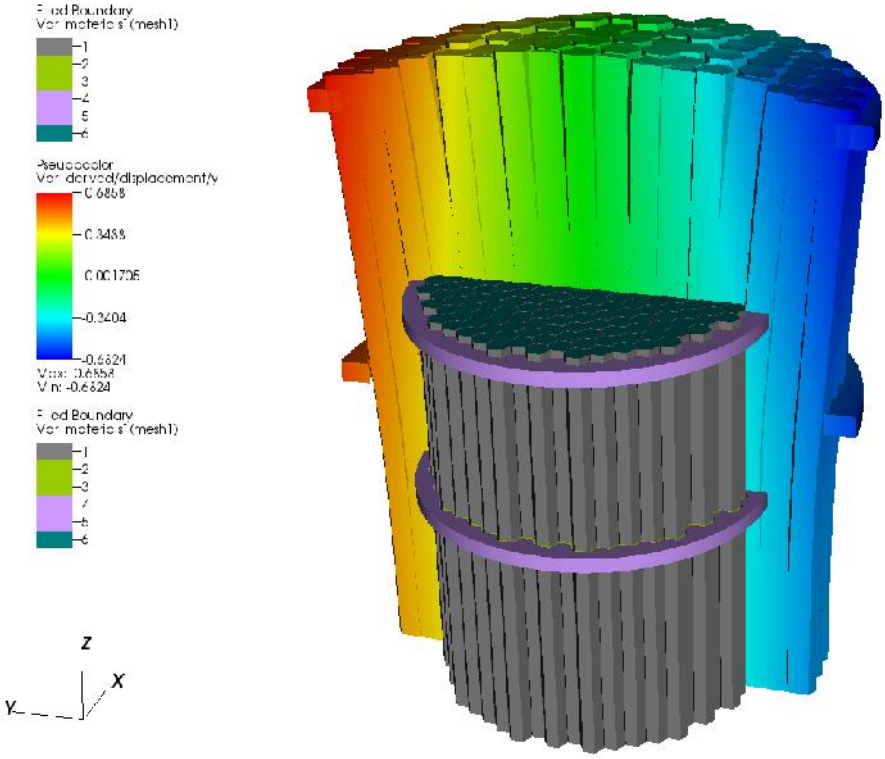


Figure 2-5 Magnified (100x) Displacements Colored by the Displacement in the *y*-direction

2.3 Evaluation of the SHARP Team's Multi-Physics Demonstration

ART end-users had the opportunity to evaluate the effort by the SHARP team to perform a preliminary demonstration of the capability to treat the combined neutronics, thermal-hydraulics, and structural mechanics effects as an SFR core undergoes structural deformation induced by thermal gradients. This was a successful demonstration that shows a unique modeling capability that, after some additional development efforts and a verification and validation campaign, could be employed to ensure SFR safety performance while avoiding expensive and exotic safety systems. A list of additional considerations and necessary improvements to SHARP include:

- PROTEUS lacks the capability to predict photon transport, and the simulations assumed a gamma heating rate that came from a separate calculation by a conventional code. These off-line simulations for gamma heating are not consistent with the fission rate predicted by PROTEUS, which in turn are dependent on the thermal-hydraulics and geometry conditions predicted by Nek5000 and Diablo. The capability to predict photon transport and the corresponding gamma heating directly in PROTEUS would be more accurate and avoid the need for off-line simulations.
- The SHARP simulations were quasi-static, which would be useful to support core restraint design applications, as described in Section 2.1.1. However, in order to support safety analysis (see Section 2.1.2), the simulations would need to be time-dependent. A time-dependent capability requires a significant development effort in

PROTEUS, which includes only an adiabatic kinetics feature, and improved quasi-static is likely necessary. In any case, the transient multi-physics capability has not yet been demonstrated with all three physics modules.

- During the demonstration simulation, if a particular duct deforms such that the fuel assembly volume increases, the mass of solid materials within the homogeneous regions in PROTEUS is conserved by reducing the density. During the density update, it is assumed that the volume of each fuel pin is fixed to their initial values, and sodium coolant rushes in to fill gaps caused by the increased volume. However, the fuel pins would also swell and lengthen under thermal gradients, thus changing the relative volume occupied by solid and coolant. A model for this should be included in these density updates.
- In the SHARP simulations, the focus was on core structural deformation associated with assembly bowing and grid plate expansion, which is controlled by the motion of the subassembly duct walls. However, the fuel pins also lengthen vertically, effectively changing the shape of the core. This motion is independent of the duct motion, and is controlled by the thermo-mechanical properties of fuel and cladding, and their associated temperature changes. This is currently accounted for in SAS4A/SASSYS-1 as a distinct reactivity feedback mechanism referred to as axial expansion. This should also be accounted for in the SHARP simulations, either by adjusting the geometry directly or by another adjustment during the density update process mentioned previously.
- Fuel Doppler feedback—associated with the strong temperature dependence of the fuel absorption rate in fuel isotopes—was not accounted for during the SHARP simulations. There is a capability in PROTEUS to adjust the fuel isotope temperatures in homogenized regions, but it needs to be exercised in the context of these simulations.
- The spatial mesh for all three physics modules was relatively coarse, as the objective was for demonstration rather than validation. Further work should be performed to provide simulations for a spatially-converged mesh, which would also provide insight into the computational expense of running these simulations and test the RGG mesh generation tool.
- The calculation procedure for performing multi-physics SHARP simulations, especially with the structural mechanics code Diablo, is still rather complex and is not automated. In fact only a couple of members of the SHARP team have proven able to manage the simulation process. Now that the demonstration has been performed, the procedure needs to be automated and new staff (on both the SHARP and ART teams) need to be trained.

The majority of these items were already described in the SHARP team's reports [12, 13], and discussions between the SHARP developers and ART engineers have been ongoing.

3 Evaluation of SHARP Neutronics (PROTEUS)

PROTEUS is a three-dimensional neutron transport code targeted for nuclear reactor applications. It can be utilized for both stand-alone neutronics analysis and coupled SHARP simulations. It is highly scalable, and is particularly applicable in situations where a high degree of fidelity is needed to resolve solution gradients or capture spatial details [4].

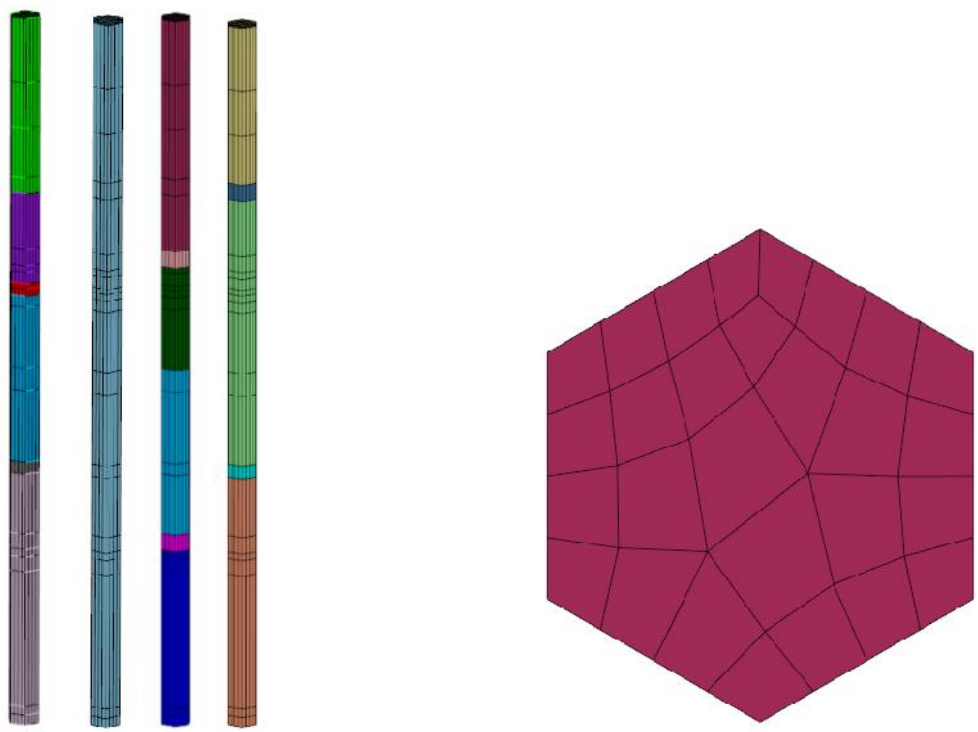
3.1 Analysis of EBR-II with PROTEUS

In order to fully evaluate PROTEUS, technical efforts were also made to use the code. PROTEUS was first used to simulate a series of benchmark problems. For a more detailed evaluation, the code was then used to model the Experimental Breeder Reactor II (EBR-II). The results described in this section should be considered preliminary; the purpose was to learn to use PROTEUS rather than to perform a thorough investigation.

EBR-II is a heterogeneous, sodium-cooled fast breeder reactor which operated from 1964 to 1994. The PROTEUS model of EBR-II was built based on equivalent MCNP and DIF3D models created in support of the SHRT-45R test neutronics benchmark analysis [19]. The particular core loading in these models corresponds to EBR-II run 138B.

Two types of driver subassemblies were used in this core loading. In addition to the driver subassemblies, the central region of the core also contained control, safety, and dummy subassemblies. Select locations contained experimental subassemblies. Reflector subassemblies surrounded the central core region, and these were in turn surrounded by a depleted uranium outer blanket. The entire core configuration was enclosed in a sodium pool. The driver and blanket subassemblies were partially depleted prior to this run. Therefore, each subassembly had a unique composition. However, in the particular DIF3D and MCNP models considered in this analysis, subassemblies of a common type were all modeled with the same composition, which was an averaged composition excluding fission products. This approach was also taken in the PROTEUS model, in order to perform code-to-code comparisons.

The mesh for the PROTEUS simulation was created using the recently-developed meshing tool Reactor Geometry Generator (RGG) [7]. This therefore allowed for further evaluation of RGG as well. Meshes of the unique subassembly types were first created using the RGG utility AssyGen. Each of these sixteen subassemblies was homogenized radially, but axial heterogeneity was preserved. In order for the different subassemblies to merge in the eventual full-core model, it was necessary to apply the same axial meshing scheme in all subassemblies. Each subassembly therefore has an axial segmentation at each elevation where any subassembly type has a change in material. Radially the subassemblies were meshed with four segments per edge, which was specified using the EdgeInterval variable. . Examples are shown in Figure 3-1. All subassemblies were constructed with “CreateSideset=No” to prevent internal sidesets in the final core model.



(a) **(b)**

Figure 3-1 Examples of the PROTEUS EBR-II Assembly Mesh, (a) Axial Mesh and Material Regions and (b) Radial Mesh

The full core mesh was constructed from the individual assemblies using RGG's CoreGen utility. The top, bottom, and collective radial sides were each assigned a sideset value, which is necessary to specify boundary conditions in PROTEUS. Coregen produced a .h5m mesh file, which is compatible with MOAB-enabled PROTEUS. However, because this version of the code is not currently stable, the file was instead converted for use with stand-alone PROTEUS. First, the mesh was converted to the CUBIT [20] exodus file format (.exo, .e) using the MOAB mbconvert utility. This exodus file was then converted to a PROTEUS-compatible .ascii mesh file using a utility recently created by the PROTEUS development team. The EBR-II mesh constructed for PROTEUS simulations is illustrated in Figure 3-2.

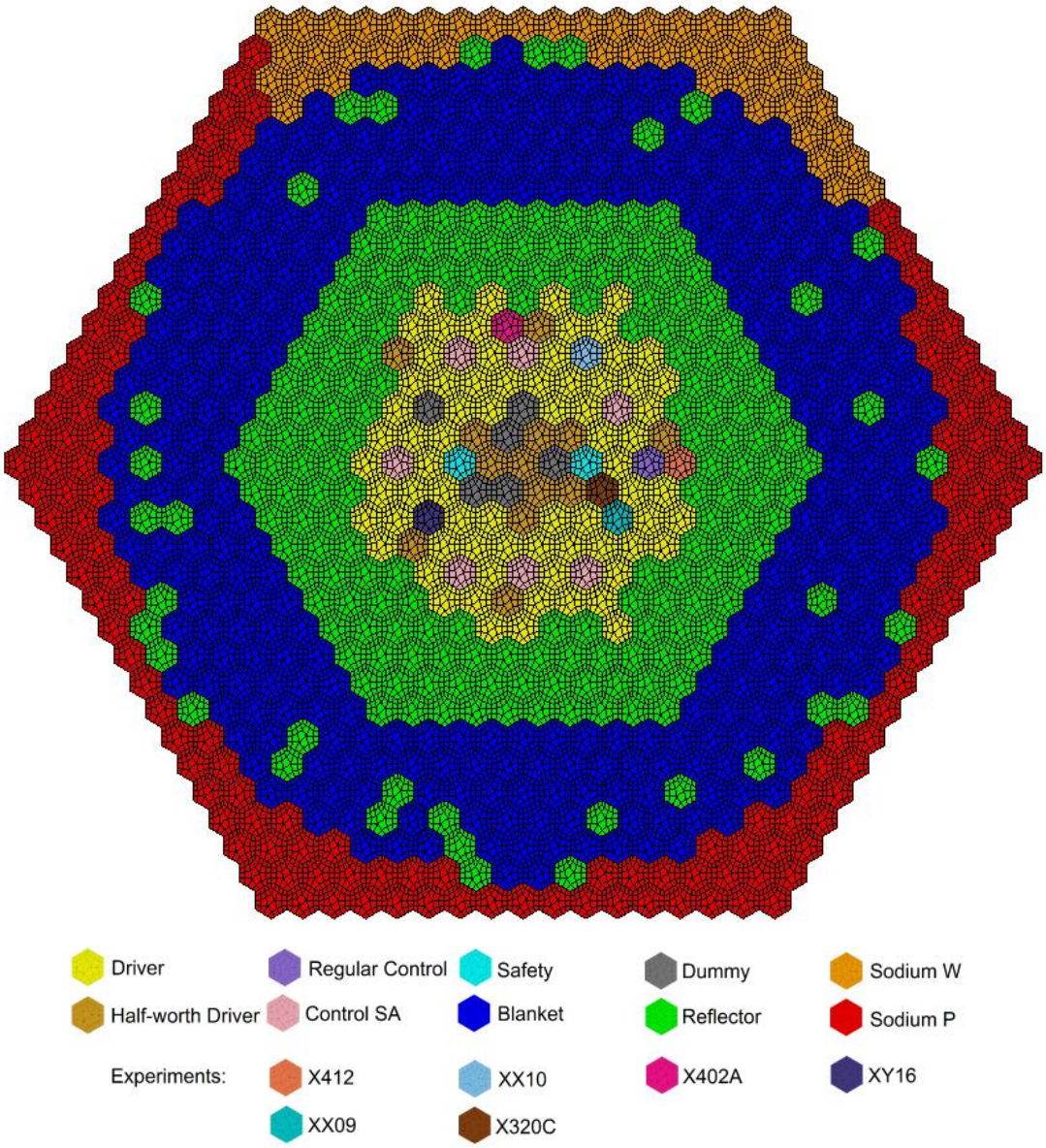


Figure 3-2 PROTEUS Full-Core EBR-II Mesh

The PROTEUS simulations used the ISOTXS cross-section file already developed for simulation of the DIF3D EBR-II model. This file was created using the MC^2-3 code, with the ANL33 energy group structure [21].

The EBR-II material compositions were first assigned to the appropriate geometric regions in the individual RGG assembly input files. Because the MCNP text input is simpler to follow than the DIF3D input, this was done based on the information in the MCNP input file, and the MCNP material numbers were used as the RGG material names. The cross-section file uses the DIF3D model's isotope naming scheme, so the actual composition information for the PROTEUS assignment file was later extracted from the EBRII DIF3D input.

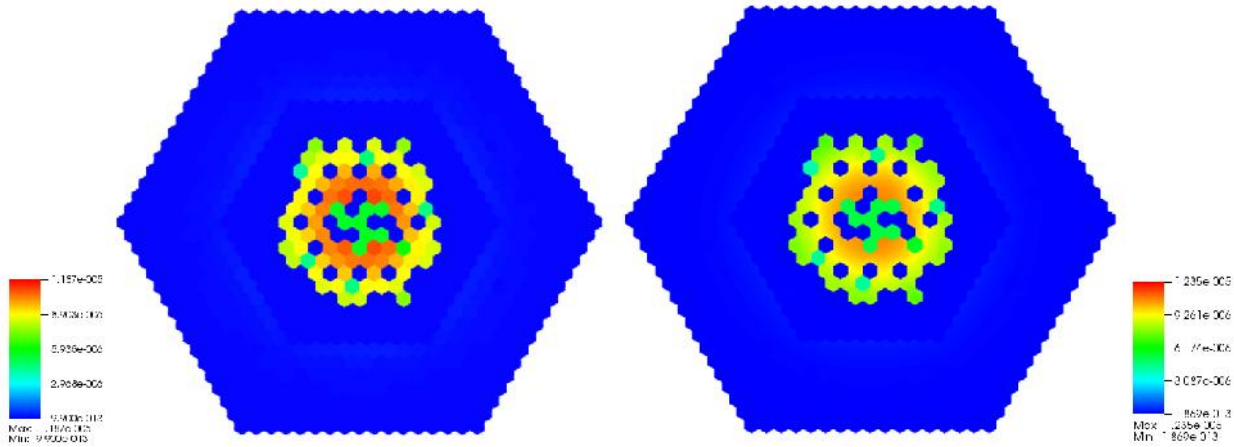
Although the material information was provided in the RGG inputs to generate the mesh, there is not a currently an automated way to provide this information to PROTEUS. PROTEUS requires material specifications via a text (.assignment) file which specifies the material in each region of the mesh, and provides material properties such as density. The isotopic naming scheme used in this file must match the naming scheme used in the creation of the ISOTXS file.

To create the EBR-II assignment file, a simple python script was developed. To determine the isotopic compositions and total density of each material, the script read the DIF3D input file and extracted this information. The script contained definitions connecting the MCNP composition names (which were used in the RGG input files) with the composition names in the DIF3D file. It then read the individual RGG assembly files, determined what the eventual mesh file region numbers will be based on the starting values specified by the MaterialSet_StartId variable, and then linked each region number to its composition (and corresponding density and isotope constituents). Finally, the script wrote all of this information to a text file in the PROTEUS assignment file format. The fixed structure of the RGG input files made it very simple to do this. Essentially, each region number can be predicted because they will be named based on incremental additions to the MaterialSet_StartId value, with a new region beginning anytime there is a change in the material specification. For example, if an AssyGen input has seven different materials axially, and MaterialSet_StartId is 201, the seven regions in the resultant mesh file will be regions 201-207. This basic logic could be utilized to develop a more universal tool for creating a PROTEUS assignment file from RGG inputs.

Simulation of the EBR-II model proved difficult because of the high computing demands of PROTEUS. After failed attempts to simulate the model on smaller machines, it was necessary to obtain an account on one of ANL's Advanced Leadership Computer Facility (ALCF) supercomputers. The model was then simulated using Legendre-Tchebychev cubature, for various resolutions in angle. The results were evaluated and compared to simulations of the MCNP and DIF3D models. Example results are shown in Table 3-1,Figure 3-3 and Figure 3-4.

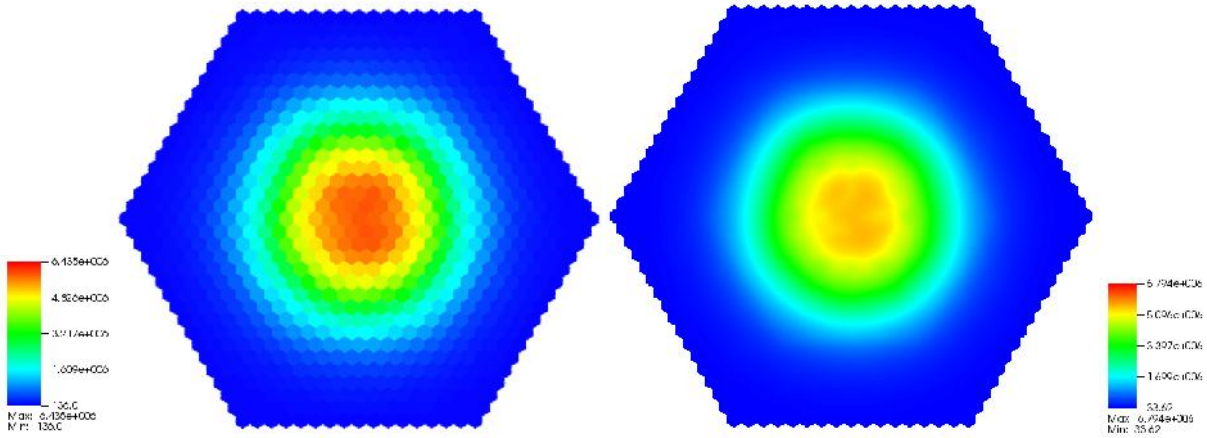
Table 3-1 Calculated Eigenvalue of the EBR-II SHRT-45 Run 138B Core Loading (Preliminary Results)

		k-eff	ρ (pcm)	Difference from MCNP (pcm)	Difference from DIF3D (pcm)
MCNP		0.99501 ± 0.00016	-501.5	-755.1	
DIF3D		0.98759	-1256.6		
PROTEUS	L1T1	0.97477	-2588.0	-2086.5	-1331.4
	L5T5	0.97737	-2315.1	-1813.6	-1058.5
	L7T23	0.97744	-2308.2	-1806.7	-1051.6



(a) **(b)**

Figure 3-3 Power Distribution in the EBR-II Core at 55% Core Height, (a) DIF3D and (b) PROTEUS (Preliminary Results)



(a) **(b)**

Figure 3-4 Group 8 (E = 302-498 keV) Flux Distribution in the EBR-II Core at 55% Core Height, (a) DIF3D and (b) PROTEUS (Preliminary Results)

It is expected that the PROTEUS results would be in better agreement with the other codes if the simulation were repeated with a more refined mesh. According to the code developers, PROTEUS cannot determine an accurate eigenvalue without a good solution spatially. However, since the focus of this analysis was evaluation of the code's usability (rather than a technical study of EBR-II), additional simulations were not pursued at this time. The results should be considered preliminary.

3.2 Feedback on Utility of PROTEUS for Reactor Analysis Needs

The evaluation of PROTEUS has considered how well the code can meet current needs of users, as well as general usability. The current code documentation has also been considered, since documentation plays an important role in usability.

3.2.1 Ability to Meet User Needs

As discussed in Section 2, PROTEUS is uniquely suitable for evaluating complex multi-physics problems such as radial core expansion, because the code is very flexible in the mesh it can use. This flexibility also benefits standalone PROTEUS users. It allows for explicit modeling of unique geometries, such as reactors with unconventional fuel designs or reactors with an in-core experiment sample, which many other codes (other than Monte-Carlo codes) cannot handle.

PROTEUS's role within the SHARP multiphysics analysis toolkit is also very beneficial to potential users. Using standalone neutronics and thermal codes often requires a great deal of work, since the output from one code must be manually translated into the information needed by the other code. Having an automated method of doing this within the framework of the coupled SHARP package will save users a great deal of time.

As mentioned in Section 2, PROTEUS lacks a photon transport model and is limited to only an adiabatic model for neutron kinetics. Both photon transport and advanced neutron kinetics models would be useful for standalone neutronics simulations, as well as the previously mentioned multiphysics problems. . Although gamma heating can be approximated with a fixed value, there are other instances where users may be interested more specific information, such as estimating the flux at some detector location. Many users would also benefit from the implementation of more advanced time-dependent simulation capabilities. The current adiabatic approximation is more accurate than point kinetics, but not as accurate as improved quasi-static or full kinetics methods.

Recently, the code developers have introduced a cross-section API, which allows for on-the-fly processing of cross-sections for the exact true reactor geometry. Verification of this utility is not yet completed. This new feature offers benefit to users, both from the ability to use the true geometry and from the fact that it eliminates the need to also construct an MC^2-3 model for cross-section generation. Both the PROTEUS cross-section API and MC^2-3 are only equipped to use ENDF-VII cross-sections; in the future, it could be beneficial to users if they also had the option to use other sets of evaluated nuclear data, such as the JEFF or JENDL libraries.

Another area of interest to potential users is the ability to perform burnup calculations. Currently, analysts can evaluate total core burnup with DIF3D/REBUS, but this does not provide detailed information. If PROTEUS were coupled with a depletion code, users could examine how the detailed flux profile affects burnup (similar to Monte-Carlo coupled codes, such as MCODE [22]). According to the PROTEUS developers, ORNL has initiated work to create an API for ORIGEN within PROTEUS, but it was not evaluated here because the effort is very recent.

PROTEUS provides a text output, which contains the final eigenvalue, and a VisIt-compatible HDF5 file [8]. The HDF5 file contains the full solution, which allows users to evaluate detailed results such as the power distribution within a single fuel pin. This degree of detail is beneficial to users who are interested in in-depth analysis of the complex behavior of a reactor core. However, in many instances users frequently want to report integral parameters, such as power peaking factors, for simple comparison of reactor designs. There is not currently code structure in place to facilitate such integration of the solution data. PROTEUS does have a recently-developed function which allows users to average solution data over *regions*, i.e. a collection of finite elements with the same material assigned. Depending on how these regions are defined by the user, these averages may not be particularly useful. For instance, the user may assign segments of different fuel pins to the same region, in order to save effort generating the material assignment

file. In this case, it would be impossible to use the region-averaging feature to calculate the pin power distribution. Similarly, fuel assembly averages could not be produced directly. One possible solution would be to develop a utility that works with RGG to recognize common units like pins and assemblies. It may be very difficult, however, to develop a universal utility for determining this information for models that were not created with RGG. Because PROTEUS does provide a very detailed solution, it would be good if the user also had flexibility in the way they average the result data. For example in MCNP, a user can request results for a region as a whole, or they can apply any degree of radial and axial segmentation for more detailed information [23]. VisIt has some capability to integrate results over user-defined regions, but this capability is not currently fully functional with the PROTEUS HDF5 file (the data can only be integrated over complete PROTEUS regions, as mentioned above, but not over any subset of a region). If this functionality were added, users could manually evaluate parameters such as power peaking factors for any model. In conclusion, it is recommended that a tool for determining power peaking factors is developed for RGG-created models, and that the ability to integrate result quantities over any region is developed for all models (either via VisIt, or via some other utility).

3.2.2 Documentation

PROTEUS documentation includes both a User's Manual and a Methodology Manual [3, 4]. This robust documentation does a very good job of providing necessary information to users. However, it would be helpful if the documentation provided a little more guidance on practical code use. It is not possible to provide fixed rules, but examples would help users to make more informed decisions in their own simulations, which could save them a great deal of time. Two key areas where examples could be helpful are (1) the computing power necessary and (2) the solution accuracy achieved, for full core simulations of models with various degrees of refinement in the mesh, the solution cubature, and the cross-section library energy structure. This type of information is something that could naturally evolve out of the future code verification and validation work.

While stand-alone usage of PROTEUS is well documented, the documentation provides little information on PROTEUS in relation to the other SHARP codes. For example, to create a PROTEUS model with RGG, a particular flag must be turned on to avoid creating internal sidesets (i.e. boundaries) between adjacent assemblies. However, neither the RGG nor PROTEUS documentation explicitly states that this flag must be used to create a model for PROTEUS. In addition, there is little information on PROTEUS's role within SHARP coupled simulations. It would be very useful for the code documentation to discuss things such as, for example, the information that PROTEUS passes to the other SHARP physics codes. This could be accomplished by additions to the PROTEUS documentation, or by the creation of a new manual on SHARP coupling.

It would also be useful for the manual to provide more discussion on the other codes that most PROTEUS users will be using, such as MC^2-3, CUBIT and VisIt. These codes of course have their own manuals, but if there is any information that could help facilitate the intelligent use of these codes for PROTEUS in particular, this would be helpful for users. In particular, this would be beneficial to new users who have no experience with any of these codes.

3.2.3 General Usability

To perform a PROTEUS simulation, there are four input files required (with supplemental files needed for kinetics jobs, fixed source problems, and parallel runs). The main input files are driver (.inp) file, which controls the PROTEUS calculation and also points to the other input files, a mesh file (.nemesh or .ascii for serial execution), an assignment (.assignment) file which places materials in the geometric regions and provides material properties, and a cross-section (binary .ISOTXS or ASCII .anlxs file). These files are each developed individually, but the information contained in them must be compatible.

Reactor analysts want to spend their time focusing on the physics of a problem, rather than on building simulation inputs. Currently, it can be time-consuming to generate the full set of inputs necessary to evaluate a problem with PROTEUS. In addition, it is necessary to learn other codes in order to generate the mesh and cross-sections for a PROTEUS simulation (although the latter could change after the PROTEUS cross-section API is verified).

The SHARP toolkits RGG and NiCE are being developed to help address these issues. RGG offers a method in which full-core three-dimensional meshes can be generated using simple text-based inputs, while NiCE will provide other utilities to facilitate input file creation and result analysis. Both of these toolkits will serve to make PROTEUS more user-friendly, particularly for new users or users analyzing standard PWR and SFR geometries.

The quality of the documentation, as discussed above, also helps to make PROTEUS more user-friendly. Having detailed documentation makes it easier for analysts to use the code. PROTEUS also appears to have good error checks in place, which similarly is very helpful.

In general, the high computing demands of the code may limit the potential user base. For example, in the EBR-II analysis discussed above, it was necessary to obtain an account on one of the supercomputers in order to run the simulation. And this was for a mesh that was still too coarse to obtain a good solution; for improved accuracy the model would have to be run with a finer mesh (which would require even more processors). Many potential users do not have access to this kind of computing power.

Currently the biggest impediment to using PROTEUS is the fact that the code has not had sufficient testing and development to be considered a production level tool. Because of this, the PROTEUS manual includes a disclaimer which states that the reliability of the code is not guaranteed. Furthermore, PROTEUS's accuracy has not been evaluated for a wide range of reactor physics problems. Without question, validation and verification are the biggest priority needs for PROTEUS. Difficulties in code use can be managed, but code reliability is essential.

The manual disclaimer also states that "In many cases, performance and efficiency have been sacrificed in favor of providing a working capability which prevents one from making a realistic computational effort comparison to other available tools. In many cases, reliable algorithms have been chosen to compensate for memory shortages on high performance computing machines which degrade the wall clock performance of the existing code". If there are steps the code developers could take to improve the performance and efficiency of the code, without sacrificing any of the current capabilities, these efforts should be pursued.

The PROTEUS developers were very helpful throughout the code evaluation. They provided instruction and feedback, and also made efforts to assist whenever possible. To facilitate the EBR-II analysis, they installed PROTEUS on additional machines, and then assisted in getting an

ALCF account when it became clear that the ANL Nuclear Engineering Division machines would not be sufficient. The code developers also addressed issues very quickly. For example, when code errors were identified during the EBR-II analysis, they made the necessary corrections to the code the same day.

3.3 ART's Requirements for PROTEUS

PROTEUS offers numerous advantages, including the ability to model unconventional geometries and the multiphysics analysis potential provided by its role within SHARP. However, there are areas where additional work should be done to improve PROTEUS for users. These include:

- Perform additional verification & validation as needed to ensure the accuracy of PROTEUS.
- Where possible, make changes to the code structure to improve performance and efficiency.
- Provide additional examples to give users a better guide for intelligent code use.
    - Computing power needed for different refinements in space, angle, and energy
    - Solution accuracy with different refinements in space, angle, and energy, and with different solution options.
- Resolve issues with VisIt file so users can integrate quantities over any region (or develop alternate utilities to do this).
- Develop utilities to determine common parameters of interest like power peaking factors.
- Expand the PROTEUS documentation to include discussions on PROTEUS in the context of the other SHARP codes – requirements for creating a PROTEUS model with RGG, explanation of PROTEUS coupling with NEK and Diablo, etc.
- Expand the documentation to include discussions on the intelligent use of MC^2-3, CUBIT, VisIt, and any other supporting software, in the context of their role in PROTEUS simulations.
- Continue verification of the cross-section API (and potentially implement the ability to use cross-section libraries other than ENDF-VII).
- Implement photon transport.
- Improve PROTEUS's time-dependent simulation capabilities.
- Couple PROTEUS to a depletion code for burnup calculations.

4 SHARP Thermal-Hydraulics (Nek5000)

4.1 Overview of Ongoing Application of Nek5000 to Reactor Analysis

The Nek5000 code is actively being used to analyze hydraulics and conjugate heat transfer in SFRs for problems of interest to ART. Nek5000 Large Eddy Simulations (LES) of coolant flow in SFR bundles has been demonstrated to be scalable up to 65,000 processors [5], providing a source of validation for RANS modelling [24] and providing physical insight into complex flow phenomena such as the hot spots that may develop due to flow stagnation near the contact point of the wire-wrap and fuel pin [25].

Nek5000 is being used to analyze the JAEA sodium jet mixing experiment PLAJEST under an ART-supported collaboration with CEA and JAEA. The objective of this exercise is to validate Nek5000 for the prediction of thermal fluctuations in sodium that may result in thermal striping and fatigue in structural materials in SFRs. Preliminary results were presented in a prior report [26], and sample results are provided in Figure 4.1. The Large Eddy Simulations shown in Figure 4.1 were performed by a member of the Nek5000 development team, although future work is being planned that will include contributions from a nuclear engineer whose training was supported by this project.

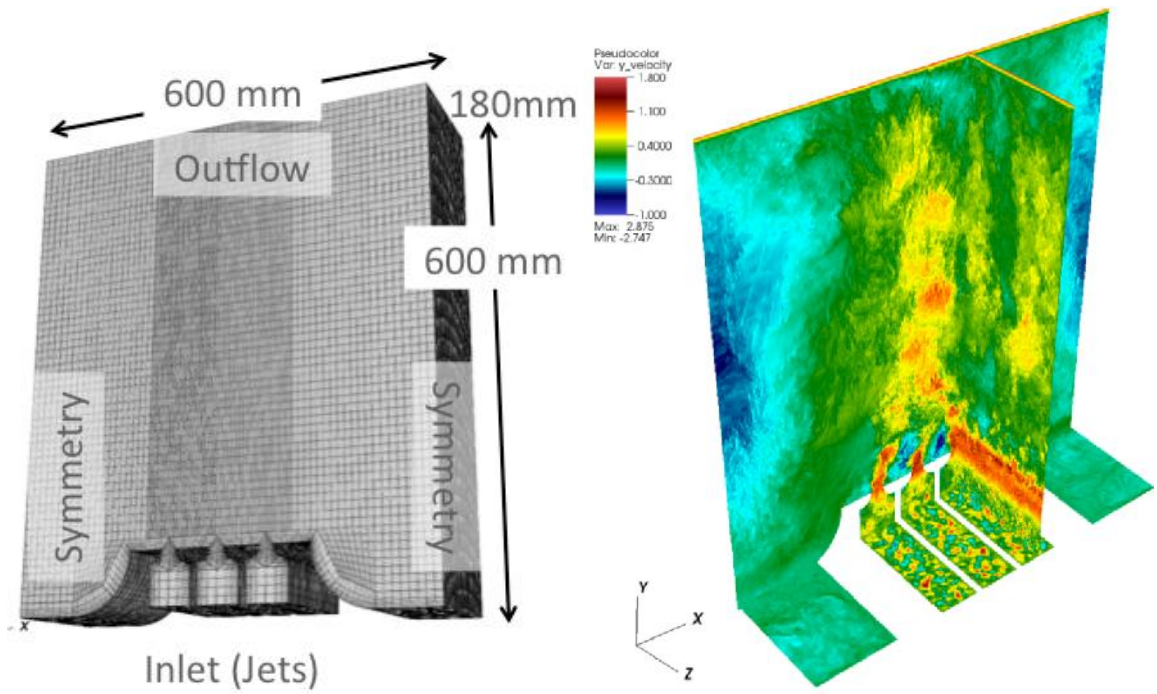


Figure 4-1 Model (left) and Velocity Predictions (right) for the PLAJEST Benchmark (Preliminary Results)

In addition, two-phase boiling models are being implemented in Nek5000 for analysis of LWR core thermal-hydraulics in an effort supported by another program. One of the co-authors of this report received training in Nek5000 and is working with the development team to implement two-phase flow models into Nek5000. Nek5000 was originally developed to model incompressible flows, and so a simple homogeneous flow solver is being implemented as a first

step to modeling boiling. The main assumption of this homogeneous model is relative velocity of the two phases, i.e, water and vapor, is negligible. It is well known that the assumption of homogeneous flow is insufficient to describe LWR core channel thermal-hydraulics. Thus, the initial exercise of implementing the homogeneous flow model is intended to evaluate methods and numerical development issues without the complexity of developing more sophisticated models. As an example of the results of the new capability in Nek5000, some simulations of a vertical heated pipe experiment [27] are presented Figure 4-2 with preliminary comparisons to measured wall temperatures in Figure 4-3. The model predicts dryout occurring at a significantly lower elevation in the pipe, and does not capture the rapid temperature rise correctly. These errors can be attributed to known deficiencies in the homogeneous flow model, and will be improved in future work by implementing more sophisticated drift flux and two-fluid models.

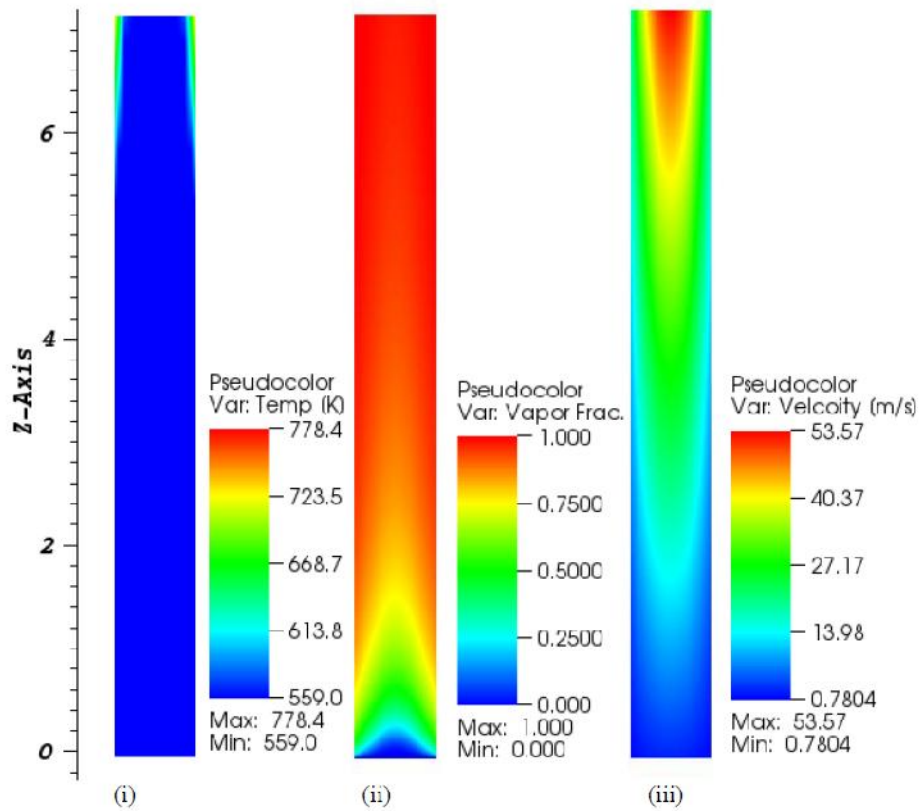


Figure 4-2 Contour Plot of Predicted (i) Temperature, (ii) Vapor volume fraction and (iii) Velocity Magnitude Profile of Heated Flow in a Vertical Pipe (Preliminary Results)

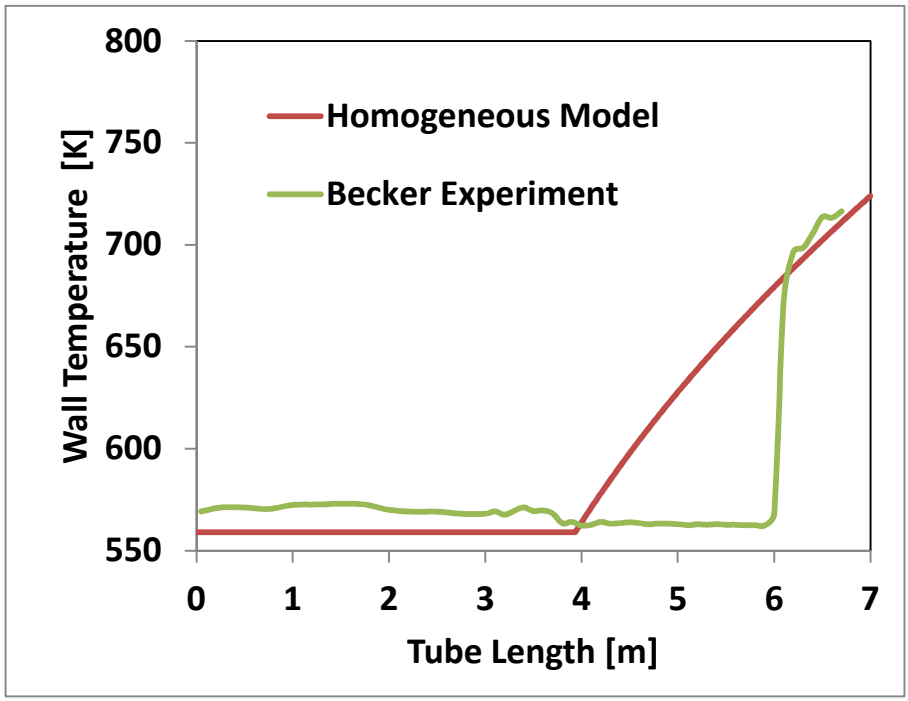


Figure 4-3 Comparison of Experimental Measurements to Nek5000 Predictions using the Homogeneous Flow Model (Preliminary Results)

4.2 Feedback on Utility of Nek5000 for Reactor Analysis Needs

For the important problem of structural deformation of an SFR core, Nek5000 provides a unique thermal-hydraulics modeling capability when coupled in the SHARP framework to PROTEUS and Diablo. In this regard, Nek5000 provides ART with a very useful resource for modeling and simulation.

For other thermal-hydraulic design and safety performance analysis applications, Nek5000 is one of several tools available to ART engineers. Other commonly used tools include the commercial CFD code STAR-CCM+ [28] for general flow and heat transfer applications, and Argonne's liquid-metal reactor safety systems code SAS4A/SASSYS-1 [18] for channel thermal-hydraulics and a variety of other physics modeling for SFR safety performance. SAS4A/SASSYS-1 includes a sub-channel model to compute steady-state and transient fuel, cladding and coolant temperatures in each fuel pin and coolant sub-channel of a subassembly, which was validated using the measured temperatures in the special instrumented subassembly XX09 in the Experimental Breeder Reactor-II during the Shutdown Heat Removal Test-17 [29]. STAR-CCM+, which is primarily used with its steady or unsteady Reynolds-Averaged Navier-Stokes (RANS) turbulence modelling options, has been used in a variety of SFR pin bundle analyses. Some of this analysis work was motivated by a study of a particular 217-pin bundle simulation where an interesting phenomenon was observed: RANS predicted a periodicity in the predicted coolant temperature profile induced by the wire-wrap, which cannot be predicted by the current SAS4A/SASSYS-1 sub-channel model. Such studies can now be used in to improve sub-channel models [30]. Other applications include a study comparing spacer grid and wire-wrap options [31].

Nek5000 provides the highest level of fidelity in the set of available tools, as it includes Direct Numerical Simulation and Large Eddy Simulation (LES) of turbulent flows coupled to conjugate heat transfer models. The LES capability has been shown to reproduce measured data with a high level of accuracy for several relevant applications, including heat transfer in electrically heated rod bundles [32] and turbulent jet mixing [33,34]. Because Nek5000 predictions have been demonstrated to be so reliable, ART engineers would be able to use it to reduce the number of experiments required to test designs (e.g. for pressure tests of subassembly inlets) and to perhaps to push designs into regions where predictions by other codes have high uncertainty, e.g. for situations with flow instabilities or flows that are not fully developed. However, one immediate use of Nek5000 would be to provide additional validation points (in addition to experimental datasets) that can provide validation to the lower-order methods which can then perform the routine design analysis. This approach keeps the computational expense reasonable, while still providing a high level of confidence in code predictions. Past experiences have shown, e.g. for the MATIS benchmark, that the commercial code STAR-CCM+ can be used effectively after selecting turbulence modeling options based on predictions from prior Nek5000 simulations.

Furthermore, the NEAMS team is currently working to provide a flexible modeling capability in the Nek5000 code, so that it can provide several levels of modeling fidelity itself. Development of an unsteady Reynolds-Averaged Navier Stokes (URANS) turbulence model is underway, which will significantly reduce the computational burden of high-resolution simulations by employing more approximate models for turbulence. For the multi-physics demonstration problem, a porous media modeling capability was developed, which is a much faster but lower-fidelity model that homogenizes geometric detail.

Because Nek5000 is an open-source code and code developers are available at Argonne, there are no barriers to updating the code as ART's modeling needs require. This advantage was demonstrated during the core structural deformation exercise: the demonstration required a faster-running model, and the porous media model was implemented in a matter of weeks.

Nek5000's success at performing well on extreme scale computers will enable it to perform large-scale simulations that are impractical if not impossible to perform with commercial tools like STAR-CCM+. The parallel algorithms used by such commercial tools impose an upper limit on the number of parallel processes that can be executed efficiently. In fact, based on experience with other programs, the commercial tools have other limits on the number of mesh that can be generated, even with more than sufficient physical memory on the computing workstation. For such problems, Nek5000 becomes the more practical option, with less time required for the simulation.

Although the exercise of implementing two-phase boiling models in Nek5000 is not directly relevant to SFR analysis needs in the near term, it does demonstrate that the Nek5000 code can be learned by new users who can make substantive code development in a relatively short period of time. Thus we anticipate that Nek5000 can continue to be used and further developed by engineers who are already supporting the ART program.

While the physical modeling capability and parallel scaling provide unique advantages for Nek5000 for ART's modeling and simulation needs, commercial tools will continue to be used for routine analyses for the foreseeable future. Commercial vendors provide documentation, mesh generation tools, post-processing tools, and dedicated technical support that can never be

practically matched by tools developed in the Lab system. These tools enhance user productivity, so for moderately-sized problems where commercial tools like STAR-CCM+ can be used with confidence, they will continue to be preferred.

4.3 ART's Requirements for Nek5000

There are several areas in which the NEAMS program could support further development to make Nek5000 a more appealing tool, including:

- The computational expense of Large Eddy Simulations is often significantly greater than the RANS techniques offered by commercial tools like STAR-CCM+. The continued development of the unsteady RANS capability in Nek5000 would enhance its appeal to ART's needs.
- Although Nek5000 has already performed several successful validation exercises with comparisons against measured data, the validation effort must continue to provide confidence in its capability to model important thermal-hydraulic phenomena in sodium-cooled fast reactors, such as buoyancy-driven flow, conjugate heat transfer, turbulent mixing, and heat transfer in liquid metals. Nek5000's primary advantage over commercial CFD tools is its potentially more accurate solution, so demonstrating and documenting this capability is essential to meet quality assurance requirements for nuclear reactor analysis.
- The documentation available for Nek5000 is actually based on a commercial product (Nekton) that precedes Nek5000, and some additional documents that describe additional features. However, the documentation is not a complete and coherent document, which imposes a barrier to its adoption.
- Nek5000 has stricter mesh requirements than most commercial CFD tools, i.e. all elements must be hexahedra, possibly with curved edges. Although such meshes may produce more accurate results, they are usually time-consuming for most analysts to generate for complex geometries. The NEAMS team should respond with a plan for reducing this barrier to entry for Nek5000, e.g. by allowing more flexible meshing options in Nek5000, by furthering the development of MeshKit and RGG, or perhaps by working with commercial vendors for mesh generation to recommend tools to new Nek5000 users.
- Boundary conditions are specified in a Fortran source file (userbc) routine and thus are subject to user coding error each time this routine is touched, which may make satisfying code validation requirements difficult in the future. It is true that only the code setups could be validated and, to be precise, only for a particular experiment setup or a class of problems. Perhaps additional parameters could be added to the user input file for cases with simple boundary conditions, in order to reduce the number of models that require the development of user code and thus the number of associated user coding errors.

5 SHARP Structural Mechanics (Diablo)

5.1 Application to Soil-Structure Interaction Analysis

5.1.1 Introduction

The test problem chosen is to calculate the vertical impedance function of a massless square footing on an elastic half-space in the time domain. The reason for choosing this test problem is that it is a straightforward problem for a finite element method (FEM) analysis and the analytical solution is available for comparison. The impedance function is the dynamic force-displacement relation between a massless foundation and the supporting medium. Calculating the impedance function is the first step in the two-step approach for Soil-Structure Interaction (SSI) analysis. The second step is to solve the model of the structure and its foundation with the impedance function placed at its bottom and mass of the foundation is included in the model. This two-step approach does not require intensive computation and large disk space for data storage; therefore, it has been the preferred approach for the nuclear industry. To compute the impedance function requires the solution of the equation of motion for elastic waves propagating in a 3-D half-space medium. This problem is usually solved analytically in the frequency domain due to the mathematical difficulty. The major obstacle for solving this problem using FEM in the time domain is the infinite boundary of the supporting soil. Waves are radiating outward from the vibrating structure toward infinity. In FEM calculations, only a finite region of the foundation medium is modeled and the waves are reflected back to influence the structural response; as a result, errors are introduced into the results. To cope with this difficulty, Lysmer and Kuhlemeyer developed an absorbing boundary in 1969 [35] that can absorb the incoming waves and is suitable for FEM analysis. These boundaries treat both dilatational and shear waves with acceptable accuracy in many applications; however, it is not perfect in terms of absorbing the incoming waves and cannot absorb the Rayleigh surface waves. As a result, the location of the boundary for the model still needs to be placed far away from the structure, which requires long and intensive CPU computation and large disk space for data storage. However, with parallel computing, SSI analyses in time domain become feasible.

Theoretically, SSI analyses by either the frequency domain approach or the time domain approach should result the same answers; however, in practice, this is not the case. The result obtained by the frequency domain approach using Fast Fourier Transform (FFT) algorithm is the steady state solution. In order to get the transient solution, the duration of the time history has to be increased by an approach known as zero packing. Due to the limitation of computer memory, usually the duration of zero packing is not long enough to get the transient solution; as a result of this, error is introduced. In addition, the frequency domain approach cannot be used to solve nonlinear problems, either the structure is nonlinear or the response is nonlinear. For SFR plants the system operates at high temperature but very low pressures, usually close to atmospheric pressure. As a result, these SFR plants are characterized by thin-walled components and piping which are not suitable for resisting seismic loads. To reduce capital cost, usually the base isolation system is used in the design to reduce the seismic demand. A base isolated SFR is a nonlinear structure that cannot be analyzed by the frequency domain approach. Therefore, for fast nuclear power plants, time domain approach should be the future direction for SSI analyses.

Diablo has implemented the Lysmer-Kuhlemeyer boundary condition. Therefore, it can be used to perform SSI analysis in time domain. In this report, calculating vertical impedance

function of a square footing on an elastic half-space using Diablo is presented and results are compared with those of the analytical solutions. Note that most commercial FEM codes do not implement the absorbing boundary condition; therefore, they cannot be used for SSI analysis.

An additional reason for choosing this sample problem is that it also serves as the first user's feedback: Diablo has a distinct feature; it can perform the dynamic SSI analysis in the time domain that most commercial codes cannot. Additional comments of using Diablo can be found at the end of this report.

5.1.2 System Considered

The system investigated is shown in Figure 5-1. It is a massless, rigid, square footing which is supported at the surface of a homogeneous, isotropic, linearly elastic half-space and the square footing is vibrating with a fixed amplitude sinusoidal displacement in the vertical direction. The origin of the x, y, and z Cartesian coordinate system is located at the center of the square footing, and the side length of the square footing is denoted by B. The supporting soil is characterized by its unit weight, ¥ , shear wave velocity, Vs, and Poisson's ratio, ¬ . The vertical sinusoidal displacement of the foundation, *w(t)*, may be expressed as

$$w(t) = A_0 \sin(2® \, ft) \qquad (1)$$

where *f* is the frequency of the vertical sinusoidal displacement in Hz (cycle per second) and A0 is the amplitude in ft. The response quantity of interest is the steady state response of reaction force exerted by the square footing. Let this quantity be denoted by $R_0 \sin(2® \, ft)$ and the impedance function be denoted by $K_0 \sin(2® \, ft)$; then, K_0 is related by R_0 by

$$K_0 = R_0 / A_0 \qquad (2)$$

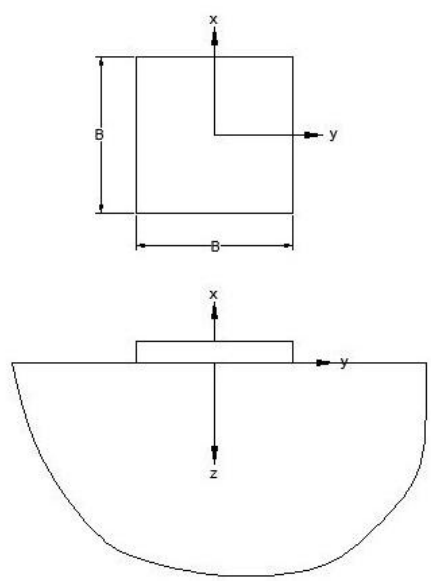


Figure 5-1 System Considered for Soil-Structure Interaction

5.1.3 Modeling and Approach

The FEM model of the system is shown in Figure 5-2, in which the elastic half-space is modelled by a square block with a side length of *L* and depth of *D*. The massless foundation is not modelled. However, the sinusoidal vertical displacement of the foundation is specified at the nodes representing the interface of the massless foundation and the supporting soil. Lysmer and Kuhlemeyer absorbing boundary condition is specified for all four-side and bottom surfaces. In the numerical data presented herein, the unit weight of soil is taken to be 120 lb/in^3, V_s = 1000 ft/sec, and ¬ =0.45. The soil element size is determined from Section 3.3.3.4 of ASCE 4-98 [36] that the vertical dimension of the soil element should be smaller or equal to one-fifth of the smallest wave-length of interest. The maximum frequency considered in the analysis is 33.3 Hz; therefore, the vertical dimension of the soil element, d, should be less than

$$d = \frac{V_s}{5 \times 33.3} = 6\,ft \qquad (3)$$

In this report, cubic soil element with size of 5 ft. is used. Nodal coordinates and element connectivity are generated by ANSYS, commercial computer software and then imported into the Diablo input files as the model description. Two sizes of foundation are considered, B= 10 ft. and B=20 ft. For B=10 ft., the values of *L* and *D* considered are (*L*, *D*) = (100, 100), (150, 125), (200, 150), (250, 175), (300, 200), and (400, 200) in unit of ft. For *B*=20 ft., values of (*L*, *D*) = (200, 150), (300, 200), and (400, 200) are considered. The values for frequency of the vertical sinusoidal displacement considered are *f* = 0.2, 0.5, 1, 2, 5, 10, 12.5, 16.67, 25 and 33.3 Hz. The time step for numerical integration, Δt , is set to be

$$\Delta t = \frac{1}{20 f} \qquad (4)$$

The initial conditions for the numerical integration are zero for the displacement and 2® fA_0 for the velocity. Note that for numerical stability, the displacement specified for the nodes at the interface of the footing and the supporting cannot be calculated directly from Equation 1 [37]; a time history generated from Equation 1 has to be filtered through the Newmark time integrator and the time step chosen defined by the following two equations:

$$a_{n+1} = \frac{1}{¥\Delta t}(v_{n+1} - v_n - (1-¥)\Delta t\, a_n) \qquad (5)$$

$$x_{n+1} = x_n + v_n \Delta t + (0.5 - \quad)\Delta t^2 a_n + \quad \Delta t^2 a_{n+1} \qquad (6)$$

where $¥ = 0.25$ and $= 0.5$; x, v, and a are the displacement, velocity, and acceleration, respectively; the subscripts n and n+1 are the two consecutive integers representing two consecutive time steps.

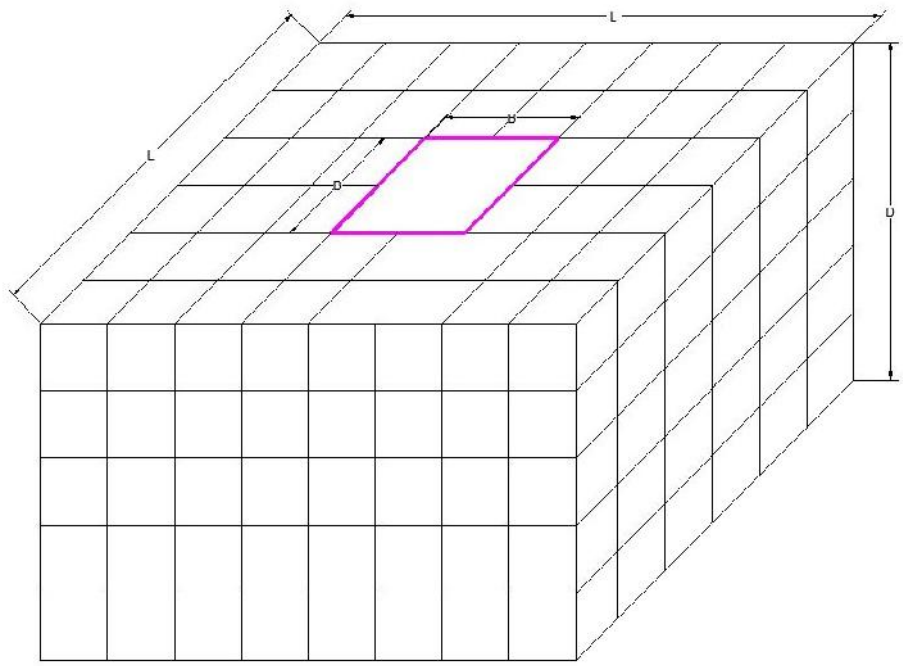


Figure 5-2 FEM Model for Soil-Structure Interaction Test Problem

5.1.4 Results

The results of calculations using Diablo for cases considered are presented in this section. The numerical data for K_0 are tabulated in Table 5-1 and Table 5-2. The analytical solutions taken from Reference 38 are also included in the same tables for comparisons. Theoretically, data obtained from Diablo should approach to these solutions. Examining data presented in Table 5-1 and Table 5-2, one can see that the Diablo results are in the same order of magnitude as those of the analytical results. However, one may notice that the accuracy of the Diablo results does not monotonically increase with the increase of the value of *L* as one would have expected. One may also notice that for some cases, the results seem to converge upon a different value than that of the analytical solution. However, one should not be concerned about these trends because coarse meshes are used for the footing (4 elements for *B*=10 ft. and 16 elements for *B*=20 ft.); therefore, the accuracy of the results should not be expected. Note that obtaining accurate results is not the main purpose of this study; therefore, convergence study was not carried out. However, to show that the solution does converge to the analytical results, additional runs were performed for the case of *B*=10 ft. and *L*=200 ft. with *d* = 2.5 ft., a half of element size used for other cases, and *d* = 1.25 ft. for *f* = 1 Hz. The results are listed in Table 5-3. For the purpose of comparison, the results for element size of 5 ft. are also listed and the percentages of error with respect to the analytical results are presented. One can see that the accuracy of the results does improve by using smaller element size. Note that the FEM model for these cases has more than 1 M degree-of-freedom. It consists of more than 400 thousand nodes and 384 thousand brick elements. For the case of *d* = 1.25 ft. and *f* = 1 Hz, the job took more than 14 hours on Eddy cluster using 160 CPU cores.

The filtered displacement calculated by Equation 5 and 6 for f = 1 Hz is presented graphically in Figure 5-3. The corresponding reaction time history is plotted in Figure 5-4. One can see that in Figure 5-4 the effect of initial conditions on the response dies out very quickly; then the response reaches steady state. To appreciate the Lysmer and Kuhlemeyer absorbing boundary condition, an identical FEM model is built with fixed boundary condition replacing the Lysmer and Kuhlemeyer absorbing boundary conditions at sides and bottom. The reaction time histories calculated with the fixed boundary condition and with the Lysmer and Kuhlemeyer absorbing boundary condition for B = 10 ft., L=200 ft. and f = 33.3 Hz are presented in Figure 5-5 and Figure 5-6 , respectively. The time history in Figure 5-5 clearly shows the influence of the reflected waves from the fixed boundary conditions. Comparing these two figures, one can see clearly the benefit of the Lysmer and Kuhlemeyer absorbing boundary condition. Note that the effect of fixed boundary condition depends on the frequency of the excitation. It is less pronounced for low frequency excitation. Instead of f = 33.3 Hz used for Figure 5-5 and Figure 5-6, f = 1 Hz is used to compute the response using the model with the fixed boundary condition. The reaction time history is plotted in Figure 5-7. Comparing Figure 5-7 with Figure 5-4, one can see that the difference is not as dramatic as that of between Figure 5-5 and Figure 5-6.

Table 5-1 Vertical Impedance Function for Square Footing with *B*=10ft

$f\,(Hz)$	$K_0\ (\times 10^8\,lb)$						
	L(ft)=100	150	200	250	300	400	Ref. 2
0.2	0.87	1.48	1.79	1.61	1.80	1.80	1.55
0.5	1.36	1.64	1.84	1.64	1.83	1.41	1.55
1	1.87	1.45	1.44	1.64	1.84	1.83	1.55
2	1.05	1.87	1.85	1.43	1.83	1.42	1.55
5	1.24	1.81	1.76	1.39	1.79	1.81	1.55
10	1.00	1.21	1.66	1.39	1.86	1.86	1.58
12.5	1.61	1.85	1.76	1.77	1.77	1.44	1.61
16.7	1.44	1.82	1.70	1.51	1.26	1.88	1.66
25	1.47	1.05	2.01	1.72	1.30	1.94	1.79
33.3	1.44	1.73	1.98	1.94	1.69	2.14	1.99

Table 5-2 Vertical Impedance Function for Square Footing with *B*=20ft

$f\,(Hz)$	$K_0\ (\times 10^8 lb)$			
	L(ft)=200	300	400	Ref. 2
0.2	2.93	3.36	3.31	3.10
0.5	3.46	3.43	3.37	3.10
1	3.03	3.46	2.83	3.09
2	3.34	3.44	3.42	3.09
5	3.35	3.26	3.49	3.17
10	3.44	3.76	3.55	3.4
12.5	3.36	3.63	3.72	3.59
16.7	4.16	4.43	4.29	3.98
25	5.31	5.03	5.93	5.15
33.3	7.44	7.88	7.91	6.85

Table 5-3 Comparison of Vertical Impedance Functions for Square Footings with B=10ft, L=200ft and Element Sizes of 5ft and 2.5 ft.

$f\,(Hz)$	$K_0\ (\times 10^8\,lb)$					
	$d = 5\,ft.$	% error	$d = 2.5\,ft.$	% error	$d = 1.25\,ft.$	% error
0.2	1.79	15.5	1.59	2.6		
0.5	1.84	18.7	1.55	0		
1	1.44	-7.1	1.72	11.0	1.66	7.1%
2	1.85	19.3	1.71	10.3		
5	1.76	13.5	1.64	5.8		
10	1.66	5.1	1.74	10.1		
12.5	1.76	9.3	1.38	-14.3		
16.7	1.70	2.4	1.82	9.6		
25	2.01	12.3	1.85	3.4		
33.3	1.98	-0.5	1.83	-8.0		

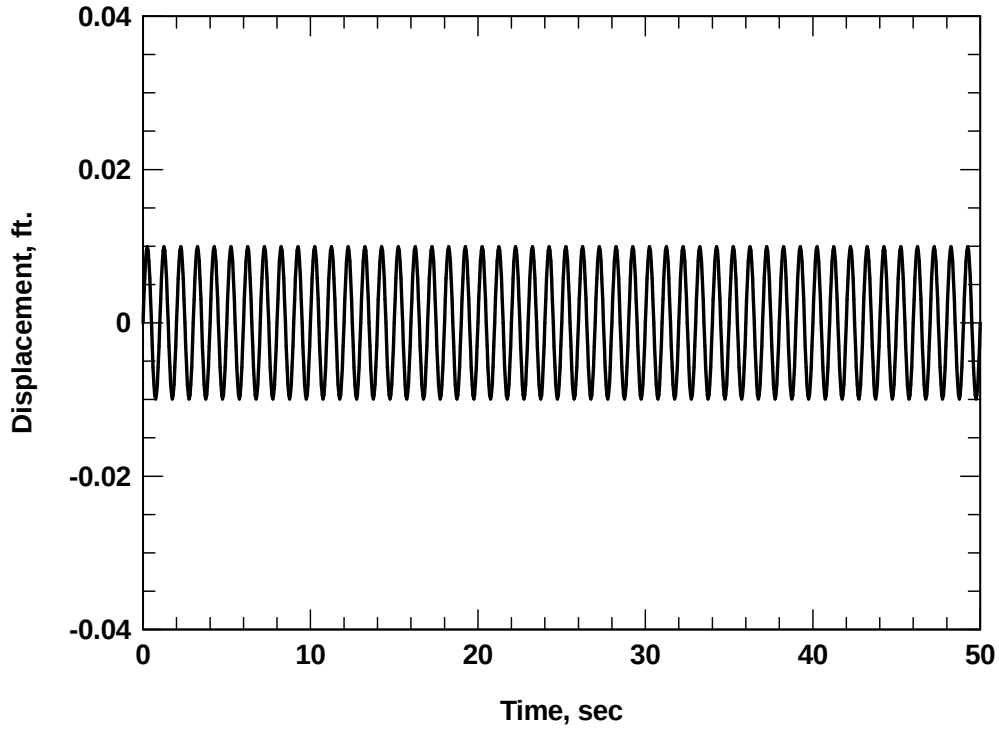


Figure 5-3 Filtered Time History of Input Displacement for *f*=1 Hz

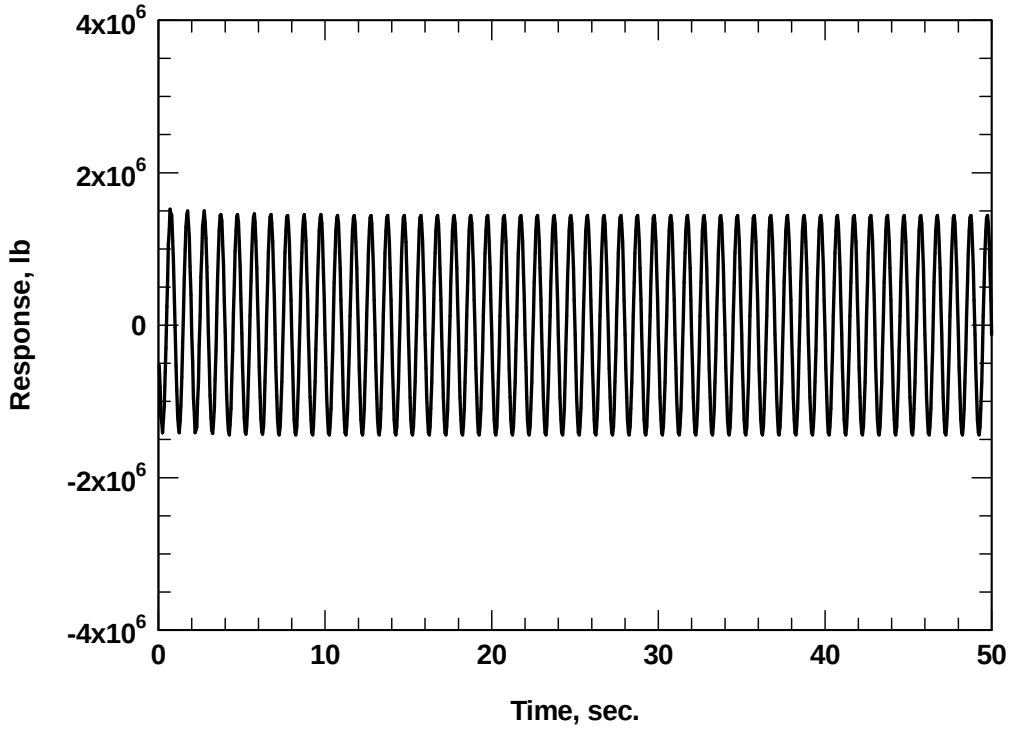


Figure 5-4 Time History of Reaction Force for Square Footing with *B*=10 ft, *f*=1 Hz (Lysmer Boundary Condition)

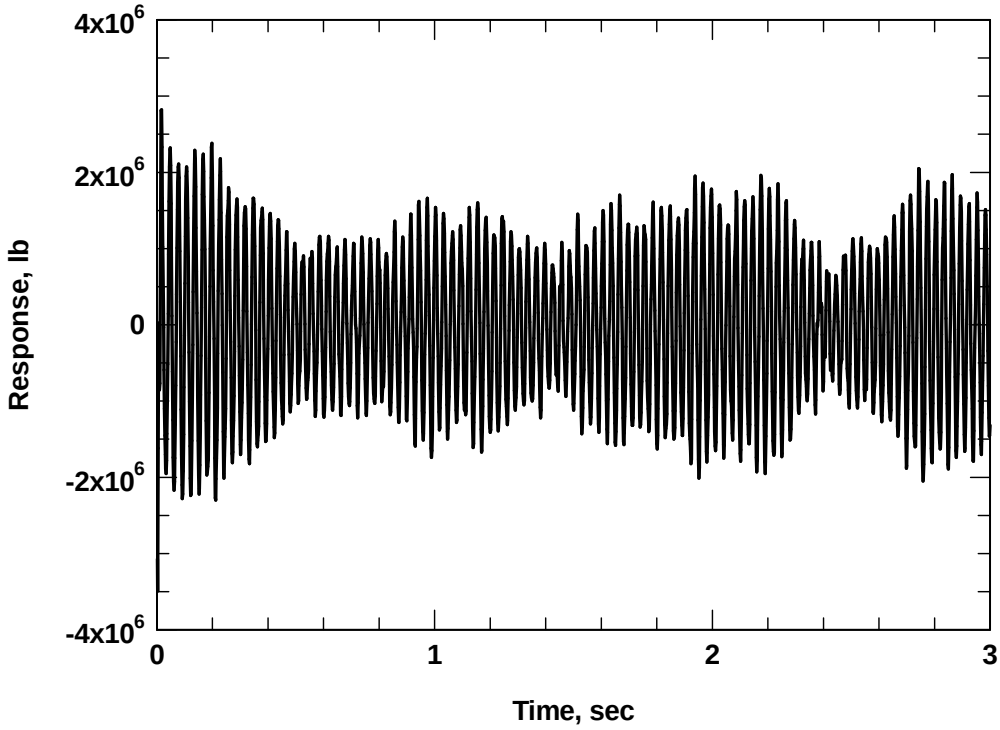


Figure 5-5 Time History of Reaction Force for Square Footing with B=10 ft, f=33 Hz (Fixed Boundary Condition)

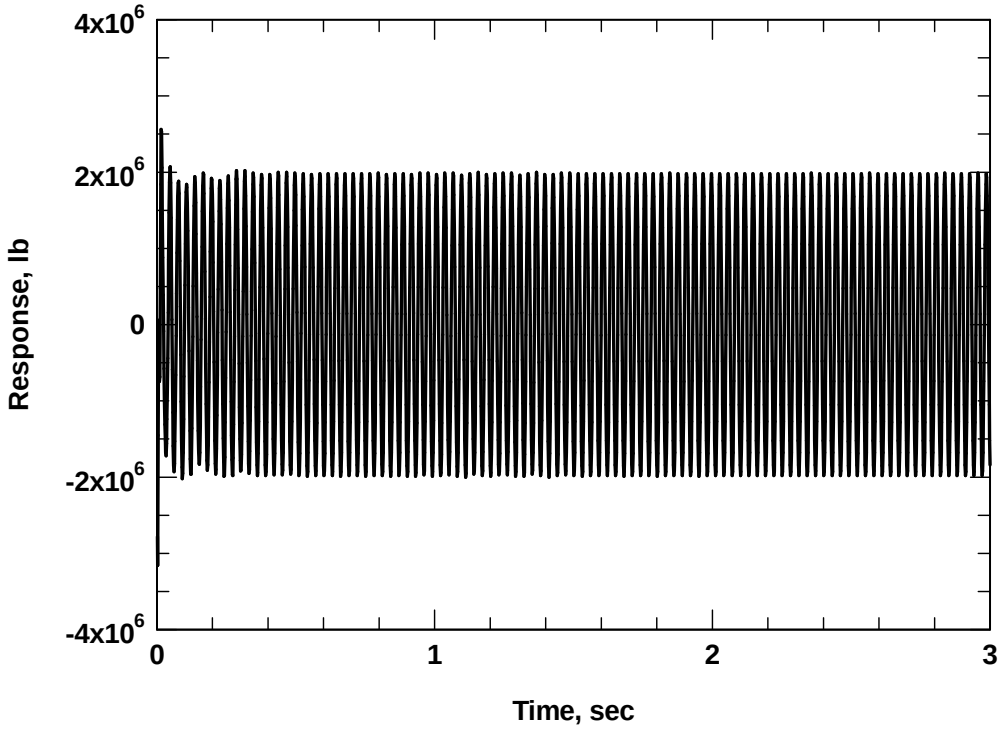


Figure 5-6 Time History of Reaction Force for Square Footing with B=10ft, f=33 Hz (Lysmer Boundary Condition)

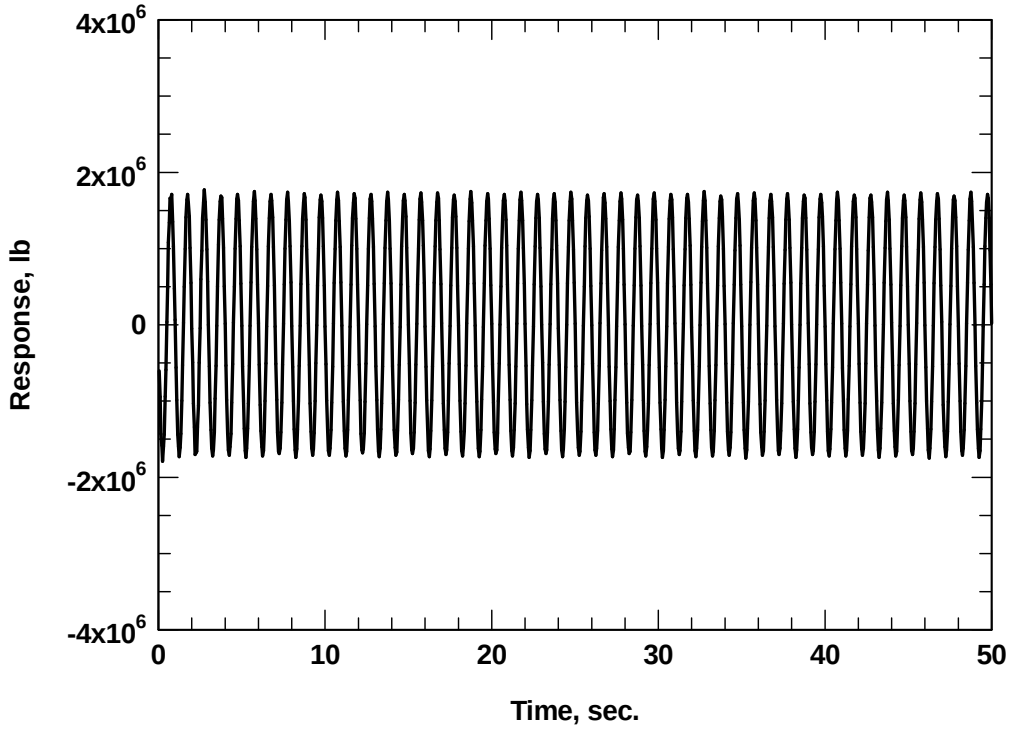


Figure 5-7 Time History of Reaction Force for Square Footing with B=10ft, f=33 Hz (Fixed Boundary Condition)

5.2 Feedback on Utility of Diablo for Reactor Analysis Needs

Because of its potential to couple to the neutronics and thermal-hydraulics modules in the SHARP framework, Diablo shows the potential to provide a unique modeling capability for core restraint design and safety performance analysis of SFRs. Other features make it appealing for other applications. The implementation of the absorbing boundary condition enables Diablo to perform soil-structure interaction analysis in the time domain, potentially providing an enhanced modeling capability for seismic analysis of SFRs featuring base isolation systems. The scalability of Diablo on high-performance computing platforms may make the time-domain analysis tractable. The Diablo code offers the state of the art Mortar method for contact analysis which is not available in the commercial software, and may be advantageous for particular applications.

However, it is anticipated that commercial FEM software will continue to be widely used for SFR structural analyses that support the ART program. Commercial vendors for codes like ANSYS and ABAQUS provide documentation, mesh generation tools, intuitive graphical user interfaces, user-input checking, dedicated technical support staff, and an adequate predictive capability for the majority of applications. Therefore, for typical applications, engineers will be much more productive with the commercial tools, being able to provide analysis results in a relatively short time frame. For certain niche applications, like those mentioned above, Diablo exhibits potential that should be pursued further.

The remainder of this section provides more specific feedback on the Diablo code, with contributions from two users who evaluated Diablo.

5.2.1 User Manual

The user manual, which is currently in a draft state, is considered to be the weakest point of the code package. The manual referenced herein is the user manual v3.0 issued on March, 2014. This manual requires serious editing because it contains many mistakes. Often, chapter, section number, or figure number is incorrectly referenced; for example, the first sentence of Section 6.1 states, "Figure 1.1 summarizes the keyword blocks that may be present in the model assembly file." However, the manual does not have Figure 1.1.

The title of chapters listed in the table of content are different from that in the main body. For example, in Table of content, the title of Chapter 15 is "Writing Restart States", but in the text, Chapter 15 is "Contact Models Material". Also, the page number in the table of content does not match with the actual page number. For example, in table of content, Page 238 is Chapter 20; however, in Page 238, it is Chapter 23.

Furthermore, the manual does not provide enough information for the user to prepare the input deck. For example, according to the manual, if HYPRE library is chosen for the linear solver, the choices for the preconditioner are NONE, BOOMER_AMG, PILUT, and PARASAILS (see Section 12.3.3 of Page 110). The default is NONE. However, there is no information given for when/why to choose or not to choose NONE. For the test problem, the developer suggested that PARASAILS be used, but the manual states that PARASAILS is for thermal problems and BOOMER_AMG is for solid mechanics problem. The test problem is a solid mechanics problem; so it is counterintuitive that PARASAILS is suggested rather than BOOMER_AMG. Starting from sample input files provided by the developers, the input files for this test problem were prepared and emailed to the developers for review. Without the developer's help, the test problem would not have worked. There are many situations like this. The user has to rely on the developer's help. It is suggested that the user manual be revised and a tutorial chapter with sample problems be provided.

5.2.2 Technical Support

For commercial code, users usually are not able to contact the developers for support. In Diablo's case, the developer also plays the technical support role. So, users can access the developers directly for help. This can be either a positive or negative experience for the user depending on the type of support question. The positive aspect is that the developer can provide the user the correct answer in a short amount of time because the developer knows how to debug the code. For example, at the early stage of running the test problem, the execution of the code was terminated due to segmentation fault 11. Segmentation fault 11 usually means that the code tries to access hardware protected memory. This does not provide enough information for debugging. However, by examining the output, the developer was able to pin point the source of the error and the problem was fixed quickly. The negative aspect is that the developer is usually very busy and may not be available to answer questions promptly. For example, Diablo has implemented the Reduced Domain method for the SSI analysis. However, it is an undocumented feature. Originally, this feature was considered to be included in the test problem, but due to lack of documentation, this feature is excluded from the test problem. In the future, it is suggested that funding be allocated for technical support.

5.2.3 Software

Diablo does not have graphical user interface (GUI). It requires users to use its own scripting language to prepare the input deck. Also, it does not have a meshing tool. The user has to use a third party meshing tool. For the test problem presented herein, ANSYS was used. The nodal coordinates and element connectivity generated by ANSYS are imported into the input file; as a result of this, the input file may become a large size file. For example, the maximum size file created for the test problem has 1.1M lines. This large size file is difficult to edit and debug. On top of this, certain blocks of the input file accept only fixed format data which makes editing and debugging even harder. Diablo does not have a post processing tool to visualize/plot the response quantities. For the test problem, MS Excel was used to retrieve the reaction forces from the output file and plot the data.

In Diablo, the input data for a model are divided into two files. One file is associated with the definition of the controls assigned to the model such as the type of the analysis (static, dynamic, linear, or nonlinear), the time step used in the integration, the convergence criterion, the solver selection, and the forcing functions. The other file is associated with the finite element definition of the model such as the nodes, elements, and boundary conditions. This separation is a very convenient feature for doing parametric studies because only the file associated with the model definition need to be modified.

5.3 ART's Requirements for Diablo

The end-user evaluation can be summarized by providing the following list of requirements to improve Diablo, making it more suitable for structural analysis needs in the ART program:

- While understandable, the main differences between Diablo and the commercial codes lies in the user interface and the user support. This is clearly the result of difference in resources devoted to this task between a developmental laboratory code and a commercial code. For example, not all user input errors are detected leading to the execution crashing due to invalid user input. Without the error detection, identifying user error is difficult. The LLNL development team has assisted in identifying these types of errors in user input. In comparison, commercial companies employ dedicated technical support engineers, who are able to quickly support customers without interfering with code development efforts.
- The manuals for Diablo are incomplete and have serveral errors, which hinders new users from learning how to use the code.
- The Diablo documentation should also include consistent guidance to the user on when it is appropriate to select the various options.
- The Diablo team must be able to provide technical support to their outside users. Learning to use the code is essentially impossible without close interaction with the developers.
- Like the other SHARP components, Diablo does not have its own graphical user interface, mesh generation tools, or post-processing tools. Mesh generation and visualizing results must be performed with third-party tools. Currently, SHARP does not have a computing environment for Diablo, as there is no NiCE interface. The SHARP mesh generation tools, like RGG, may be useful for reactor analysis (like the radial core expansion demonstration problem), but are not directly applicable to seismic analysis and other applications. The

absence of these productivity tools is one of the primary reasons that commercial tools (like ANSYS and ABAQUS) will continue to be used for routine design and analysis needs in the foreseeable future.

- Diablo has focused on non-linear structural problems with contact. For reasons related to this, the elements available are limited mostly low-order hexahedral and quadrilateral elements. In particular, a tetrahedaral element does not exist requiring brick elements for meshing. Meshing complex geometries with tetrahedral or hexahedral elements can be difficult. It is possible in the linear analysis to use degenerated elements which then assume prism or tetrahedral shapes.
- Irradiation-induced swelling and creep models are required in order to represent realistic structural deformation in nuclear reactor cores. ART engineers have implemented such models in the commercial code ANSYS by developing user-defined materials and user code that provides customized creep and swelling.
- Although commercial tools do not provide access to the source code, they do provide analysts with the means to add new capabilities (like irradiation-induced swelling and creep models) through user-developed code. If it is not possible for ART engineers to have full access to the Diablo source code, something like a user-code capability should be provided so that engineers can augment Diablo without burdening the Diablo development team.
- Currently, Diablo is only available on UNIX/LINUX platform. To attract more users, making Diablo available for PC platform should be considered because it is more economic and convenient for users to develop and test a small scaled model on local PCs to gain confidence before embarking the full scaled model on cluster computers.

The ART end-users hope that the NEAMS program can support these efforts, so that Diablo can be used to support ongoing analysis needs, such as core restraint design, safety performance, and soil-structure interaction.

6 Computing Environment for SHARP (NiCE)

The NEAMS Integrated Computational Environment (NiCE) is an open-source, Eclipse-based GUI designed to facilitate the use of modeling and simulation codes [14]. The goal of this utility is to support the full simulation process, including input file creation, job launching, and result analysis. Currently NiCE has some structure in place for Nek5000, PROTEUS, and SHARP as a whole. The NiCE team is also developing a utility called the SHARP Executable Release (SHARPER), which provides support for the setup and installation of SHARP codes.

6.1 NiCE for Nek5000

NiCE currently features a Nek5000 Model Builder, which generates the necessary input files, and a Nek5000 Launcher, which initiates a Nek5000 simulation using the files created by the Model Builder. NiCE developers have also recently created a Mesh Editor, which allows users to create new meshes for use with Nek5000.

One of the primary motivations behind NiCE's Nek5000 features is to provide support for university students taking courses in fluid dynamics and heat transfer. These students may be new to both Nek5000 and Linux systems, and they are working in the relatively short timeframe of a university semester. Therefore, having a tool which allows them to focus on the physics of the problem, without having to learn unnecessary details about code use, could be beneficial. This type of utility could also be useful in training sessions, where new users are again working within a very limited timeframe. The intended support for Nek5000 is a good idea, but NiCE still requires a great deal of development before it is fully functional and useful for Nek5000 users. The tool was not evaluated by ART end-users.

6.2 NiCE for PROTEUS

The current PROTEUS features within NiCE are the PROTEUS Model Builder and the PROTEUS Launcher. These tools are intended to create, and then run, a PROTEUS simulation.

6.2.1 PROTEUS Model Builder

The PROTEUS Model Builder is a builder for the PROTEUS driver file, the text file that contains the solution options and points to the other input files. There is currently no support for the mesh, assignment, and cross-section files. Unfortunately, there are also numerous issues with this driver file builder. The dialog window does not offer the full set of PROTEUS input variables, NiCE initially populates the input boxes with inappropriate values (probably from an example problem) that are not the default values and there is no distinction between required and optional variables. Additionally, all of the input spaces are text boxes where the user must type the input, rather than drop-down menus or radio buttons, so using NiCE is no easier than just editing the text file by hand. In fact, for simple text input files like the PROTEUS driver file, most users are more likely to modify an existing input file than to create a new one from scratch in a GUI, since this is generally faster to do.

The dialog window assumes that there will always be exactly three external boundary surfaces (called sidesets) for which the user needs to specify boundary conditions. In reality, the user could need to specify any number of sidesets, including zero, depending on the geometry of the problem. One of the advantages of PROTEUS over competing deterministic neutronics tools is

that it accepts general geometries, so the PROTEUS Model Builder should not impose additional restrictions.

Overall, it appears that the developers may have examined a single example case and assumed that its content applies to all cases when creating the PROTEUS Model Builder. Furthermore, when the actual content is written to the text file by the Model Builder, it is not written very cleanly (example shown in Figure 6-1). There is no formatting, and many extraneous comments (probably copied from an example problem) are written to the text file. While it is true that it is not necessary for users to interact with the text file to run PROTEUS from NiCE, users may want to utilize the NiCE-created file in other simulations. It is very common to build a text input with a GUI, and then to manually make small changes to the file as needed. Therefore, NiCE developers should write the input file cleanly, with the assumption that some users may want to look at it.

```
bench05.inp
SN_TYPE                 LEG-TCHEBY          ! Cubature type
Theta_Resolution        3                   ! L order
Phi_Resolution          9                   ! T order
SEGMENT_ANGLE           0                   ! Parallelization in angle

DEBUG_PRINT_LEVEL        0
DEBUG_PRINT_SETUP        0
DEBUG_PRINT_FORMATION    0
DEBUG_PRINT_OUTER        0
THERMAL_POWER            1.0

USE_PRECONDITIONER       SSOR                ! SSOR,SOR,ILU,ICC
USE_TCHEBYCHEV_ACCEL     YES

Scattering_Order         0                   ! 0,1,2,3,4,5,...
EIGENVALUE_GUESS         1.17                ! The guess for the initial eigenvalue
Tolerance_eigenvalue     1.0e-7              ! The maximum relative error to allow on the eigenvalue
Tolerance_Fission        1.0e-8              ! The maximum relative error to allow on the outer iterations for the flux
Tolerance_Flux           1.0e-7              ! The maximum relative error to allow on the inner iterations for the flux
ITERATIVE_IMPROVEMENT    0.1

Iterations_Fission       100                 ! The maximum number of outer iterations
Iterations_UpScatter     5                   ! The maximum number of upscatter iterations
Iterations_scattering    5
Iterations_SA_CG         100
Iterations_PETSc         1000

Solve_Type both

SOURCEFILE_MESH          bench.nemesh        ! Path to mesh file
```

(a)

```
PROTEUS Model Builder    proteus_neutronics_1.inp
METHOD  SN2ND   !
SN_TYPE LEG-TCHEBY  !Specifies the type of SN cubature to use (16 char name)
THETA_RESOLUTION    3   !Specifies the resolution of the SN cubature. Valid numbers for a hex reflected geometry are 1,3,5,7,9,...
PHI_RESOLUTION  5   !Only needed for product cubatures to select the resolution to apply in the radial plane. Valid numbers for a he
SEGMENT_ANGLE   1   !The number of segments to attempt in the angular approximation
DEBUG_PRINT_LEVEL   0   !Specify level of debug printing for entire routine (must be integer)
DEBUG_PRINT_SETUP   0   !Specify level of debug printing of setup related operations (must be integer)
DEBUG_PRINT_FORMATION   0   !This controls the debug printing during the tracking/matrix formation
DEBUG_PRINT_OUTER   0   !This controls the debug printing during the outer iterations
THERMAL_POWER   60000.0 !Specifies the normalization power in Watts
SCATTERING_ORDER    0   !Indicates the Legendre expansion order of the scattering kernel (must be integer)
EIGENVALUE_GUESS    1.0 !The guess for the initial eigenvalue
USE_TCHEBYCHEV_ACCEL    YES !Indicates whether Tchebychev acceleration is used
TOLERANCE_EIGENVALUE    1.0e-7  !The maximum relative error to allow on the eigenvalue
TOLERANCE_FISSION   1.0e-6  !The maximum relative error to allow on the outer iterations for the flux
TOLERANCE_FLUX  1.0e-6  !The maximum relative error to allow on the inner iterations for the flux
ITERATIONS_FISSION  8   !The maximum number of outer iterations
ITERATIONS_UPSCATTER    0   !The maximum number of Gauss-Siedel 'upscatter' iterations
ITERATIONS_SCATTER  100 !The maximum number of inner iterations per outer iteration
ITERATIONS_SYNTHETIC    100 !The maximum number of Krylov iterations to allow for the synthetic diffusion solver
ITERATIONS_KRYLOV   1000    !The maximum number of Krylov iterations to allow in the iterative flux solver
SOURCEFILE_MESH sahex1_unic.h5m !Specifies the UNIX file path to a spatial geometry mesh file (128 char pathname)
SOURCEFILE_XS   sahex1_unic_4g.ISOTXS   !Specifies the Unix file path to a cross section data file (128 char pathname)
SOURCEFILE_MATERIAL sahex1_unic.assignment  !Specifies the Unix file path to a material mapping file (128 char pathname)
EXPORT_FLUX YES !Indicates whether the scalar flux is to be exported in addition to primary reaction rates
EXPORT_FILE sahex1_4g_L1T2.hdf5 !Specifies the flux file name designation of a HDF5 or pmo file to export the solution (128 char pat
SN2ND_COMPUTES_DENSITY  YES !
BASIC_BWO   NO  !
P_MULTIGRID_ORDER   0   !
BC_ALIAS    side_set_0000001 VOID   !Used to change a boundary surface name to "VOID" or "REFLECTIVE" at runtime
BC_ALIAS    side_set_0000002 VOID   !Used to change a boundary surface name to "VOID" or "REFLECTIVE" at runtime
BC_ALIAS    side_set_0000003 REFLECTIVE !Used to change a boundary surface name to "VOID" or "REFLECTIVE" at runtime
```

(b)

Figure 6-1 PROTEUS Input Structure: (a) Example (Human-Created) Input File Distributed with PROTEUS, compared to (b) Example Input File Written by NiCE

NiCE does not allow the user to specify the filename when a new input is created. All inputs are by default named proteus_neutronics_*num*.inp, where the value of *num* is based on NiCE's internal tracking of model numbers. Changing this approach is not a top priority, but ultimately, users should be able to choose their own filenames. From a user perspective, it is much easier to keep track of different simulations when the inputs are given logical names.

There are many areas where new utilities could assist in using PROTEUS. For example, within PROTEUS, the mesh, assignment, and cross-section inputs are developed separately, but must be consistent with one another. Any job building or launching utility should check that every region in the mesh has a material (and material properties) in the assignment file, and that each of those materials has data in the cross-section file.

In instances where users create their PROTEUS mesh in RGG, NiCE could serve as an information bridge between the two codes. In particular, NiCE could utilize the material information in the RGG inputs as a starting point for the PROTEUS assignment file. NiCE could also keep track of pin and assembly lattice information, so that pin-by-pin power distribution information can be easily determined for meshes in which repeated structures have the same region number.

NiCE could also help to integrate the usage of PROTEUS with the usage of the existing ARC analysis tools, like DIF3D/REBUS. As described above, PROTEUS is intended to supplement, but not replace, these codes. These codes have very cumbersome input that can be very difficult for new users to grasp quickly. They also have fixed-format text based input, which makes it easier for users to make mistakes. It would be very useful to develop utilities which can convert PROTEUS input into DIF3D-compatible data, and utilities which can take a common set of input information and translate it into the input format needed by each code. Other, more extreme ideas include a tool that could convert a PROTEUS problem with detailed heterogenous assemblies into a DIF3D input with radially-homogenized assemblies. In general, users may often want to evaluate detailed power distribution information in PROTEUS, and then evaluate more global parameters such as total burnup in DIF3D/REBUS. From a user perspective, it is very helpful to not have to construct a model from scratch to evaluate the same design with a different code.

6.2.2 PROTEUS Launcher

The PROTEUS Launcher is a utility for running PROTEUS simulations from within NiCE. It is currently only capable of launching jobs locally or on select ORNL machines. Because all PROTEUS executables used in this evaluation were located on remote ANL machines, it was not possible to test this utility.

The PROTEUS Launcher dialog box allows the user to input two pieces of information, the path to the input driver file and the path to the PROTEUS executable. There is no structure for launching simulations in parallel. However, given the intended uses of PROTEUS, it is unlikely that users will be running serial jobs. Therefore, the Launcher will only be useful if it is capable of submitting parallel simulations. This may not be easily achievable. Many of the machines where PROTEUS may be run have unique queuing systems or job submission formats that make it unlikely that a universal job launching utility would work for them.

In general, creating a launcher for PROTEUS is probably unnecessary. PROTEUS itself is fairly simple to launch, and given the complexity of the code, it is unlikely that there will be code users who do not know how to use a Linux terminal.

6.3 NiCE for SHARP

NiCE currently offers two utilities related to running SHARP: a SHARP Model Builder, for generating SHARP input files, and a SHARP Launcher, for initiating the SHARP codes to run on a local or remote system. NiCE also includes an output analysis tool for SHARP neutronics called the Reactor Analyzer. In addition, developers are working to implement a NiCE wrapper for VisIt.

6.3.1 SHARP Model Builder and Launcher

At this time, the SHARP Model Builder dialog window is identical to the PROTEUS Model Builder discussed above. It is just a builder for the simple driver text file used by PROTEUS. There is not actually structure in place for creating a SHARP multiphysics problem. The SHARP Launcher similarly appears to be an exact duplicate of the PROTEUS Launcher.

Overall, coupled SHARP simulations are the area where NiCE could perhaps eventually offer the most benefit to the code users, since these simulations require a complex management and transfer of information from different codes. There are many features that could be added to NiCE to help support SHARP simulations. For example, it would be helpful to have checks for consistencies in some of the model parameters between PROTEUS, Nek5000, and Diablo (e.g., confirm that the coolant material is the same in all models). An even better utility would be one that allows a user to input common information, such as the material in a geometric region, and then translates that information into the different formats needed by each of the SHARP codes. Ideally, this utility could also be integrated with other codes necessary for SHARP simulations (such as MC^2-3, which is used for cross-section generation). Anything that reduces redundancies in the model creation process will have benefit to the user.

Ultimately, it would be useful for NiCE to also be integrated with RGG, MOAB, and any other utilities SHARP code users may employ in creating their input files. The integration of MOAB could be useful, for example, if a PROTEUS user constructs a .h5m mesh with RGG, and then wants to convert it first to an exodus mesh (via MOAB's *mbconvert*), and then a PROTEUS-compatible .nemesh file. If MOAB were integrated into NiCE, the user could perform all of these steps from within the GUI.

NiCE developers should also keep in mind that the probable user base for each of the SHARP codes is not the same, and therefore the needs of the users may not be the same. Nek5000 is an older code that already has a large set of users, and it will probably continue to expand its user base since CFD is utilized across multiple engineering disciplines. In addition, there has already been interest in developing tools that facilitate the use of Nek5000 in a classroom setting. Therefore, having NiCE to simplify the use of Nek5000 could be very beneficial. PROTEUS, on the other hand, is very specifically intended for analysis of nuclear reactors. PROTEUS also has very high computing power demands, and intelligent use of the code requires a good understanding of neutron transport solution methods. Therefore, PROTEUS may not have as many eventual users as Nek5000, and these users may not have the same "beginner" needs.

6.3.2 Reactor Analyzer and VisIt Interface

NiCE also includes a tool for evaluation of neutronics simulation results called the Reactor Analyzer. This tool supports analysis on a core, assembly, or pin basis, for both light water reactors (LWRs) and sodium-cooled fast reactors (SFRs).

NiCE does not currently have code structure in place for getting the information needed by the Reactor Analyzer from a PROTEUS simulation. The HDF5 file used by the Reactor Analyzer is not the same HDF5 file produced by PROTEUS. The NiCE HDF5 file requires several types of solution data integrated over axial segments in each fuel pin and fuel assembly, but PROTEUS does not currently provide such integrated data. In general, PROTEUS does not have the concept of a "pin" or an "assembly", so producing such integrated data would require a significant code development effort. The NiCE developers state in the documentation: "While we were very successful in making the infrastructure to view data from SHARP, we have not yet finished the work required to automatically generate NiCE's HDF5 file." This statement is somewhat misleading, since there is still significant effort required to calculate this integrated data and generate the NiCE HDF5 file.

While engineers very frequently require integral core characteristics to be derived from their simulations, an output analysis tool should not limit itself to providing views of integrated data. One of the biggest advantages of PROTEUS over other neutronics codes is its ability to model unconventional geometries, and to provide very detailed solutions. Therefore, any output analysis tool should also facilitate visualizing and extracting detailed local solution data from PROTEUS predictions. In response to this, NiCE developers are also working on adding a VisIt plugin for NiCE. VisIt has the capability to efficiently visualize the detailed solution data, but is not particularly user-friendly. However, this utility requires VisIt2.8, which is still in beta and has not been released, so it was not possible to do any testing of it. If this plugin will offer functions beyond those currently available in standalone VisIt, or if this new interface will be user-friendly than the existing VisIt interface, its development could provide benefit to SHARP code users. Users may often prefer to use a tool like this, which first presents the full solution, and then allows them to evaluate averaged results or summary parameters from this data, rather than a tool which is strictly focused on select region-averaged values. The more flexible the output analysis tools are, the more users they can benefit. Furthermore, PROTEUS users are not limited to analyzing standard LWR and SFR geometries. Therefore, it is useful to have output analysis tools that can accommodate any geometry.

However, the Reactor Analyzer tool could provide benefits to analyzing results from conventional reactor physics codes, like DIF3D, which employ assembly homogenization and are quite restrictive in the types of core geometry that can be represented.

As discussed in Section 3, there are indeed some instances where users are interested in examining region-averaged values or summary values. In reactor design analysis, it is sometimes useful to have a few representative parameters which allow a core to be assessed quickly. In particular, there has already been user interest in developing a utility for the determination of power peaking factors from PROTEUS simulations. Because of the wide variety of reactor designs PROTEUS users may be simulating, it would be very difficult to develop a universal utility for power-peaking factor calculations. However, a method could be implemented for determining pin-specific data for runs where the mesh is generated with RGG, using the pin,

assembly, and core specifications in the RGG input files. If NiCE is well-integrated with RGG, it could facilitate this.

It should also be noted that one of the primary purposes of PROTEUS is use in coupled multi-physics SHARP simulations. Therefore, an output analysis tool for SHARP should not be solely focused on PROTEUS neutronics results. A much better utility would be something that lets the user evaluate the relationships between the neutronics, thermal-hydraulics, and structural results. It would also be very useful to be able to examine how these relationships evolve over multiple timesteps of a given simulation.

The NiCE developers state that the Reactor Analyzer-compatible file would automatically be created during a SHARP run launched by NiCE. While this would be fine to do, any output analysis tools should also be able to extract the necessary data from output files from jobs that were not run from within NiCE. Users often want to evaluate data from runs on other machines, or from runs done by colleagues.

6.4 Documentation and Code Developer Support

Documentation on NiCE is currently provided in the form of a wiki site [14]. This site gives an overview of the current features available in NiCE, including instructions on installation and usage.

The wiki documentation is very helpful. However, there are ways in which it could be improved. Overall, the discussions of the NiCE features should be modified to better clarify exactly what each feature does. For example, on the page “Using SHARP with NiCE”, the terms “SHARP” and “PROTEUS” are used somewhat interchangeably. There is discussion of a SHARP Launcher, which “can launch SHARP jobs in NiCE”. The wiki page claims “for now the SHARP Launcher only works locally and with a remote machine at ORNL”, but it is very unclear what this utility is actually capable of launching. It does not appear that there is any structure in place within NiCE for running coupled SHARP simulations, so this SHARP Launcher is probably in fact a launcher for PROTEUS (and is therefore identical to the PROTEUS Launcher). If this is the case, the documentation should be modified to state that the SHARP Launcher currently only launches PROTEUS jobs, rather than implying it launches SHARP jobs. Inconsistencies such as this make it very difficult for users to develop a clear picture of exactly what NiCE’s current capabilities are.

The NiCE development team was helpful throughout the evaluation process. They were quick to respond to emails and questions, and provided clarification when needed. They have been receptive to hearing feedback, and have expressed an interest in addressing the concerns that have been outlined above.

6.5 SHARPER

The SHARP codes have numerous third-party software dependencies, each of which must be installed in a particular way for SHARP installation to be successful. To help handle the complexity of the SHARP installation process, a new utility SHARPER is being developed. SHARPER [39] is currently a collection of Bash scripts which manage the filesystem, and configure and install the necessary codes. According to the README file distributed with the SHARPER source code, SHARPER currently supports the SHARP codes Nek5000, MOAB, and PROTEUS, as well as the supporting software packages MPICH, HDF5, NetCDF, METIS,

PETSc, and MOAB. The optional packages ParMETIS, VTK, and NiCE are also supported. SHARPER could be very beneficial to SHARP code users. In general, many analysts may not have a great deal of experience in code development and installation. SHARPER will allow them to install codes that they otherwise may not have been able to install on their own. Even analysts with some experience in code installation could benefit from SHARPER because of the complexities involved in installing these interdependent codes.

In the future, it would be very helpful for SHARPER to be expanded to also include installation of the SHARP mesh-generation toolkit MeshKit (and all additional supporting software). This would require the introduction of several additional options to the installation script, *install.sh*, because there is variety in the supporting codes users will want to install, depending on their planned usage of MeshKit.

This capability has not been sufficiently evaluated by the ART end-users.

6.6 ART's Requirements for NiCE

Further development of NiCE should be approached with a more clear vision of the intended final product, and all of its utilities and features. Currently, a great deal of work has been done on NiCE, but NiCE still does not provide benefit to SHARP code users because of "holes" in the developed features. Before moving forward with NiCE development, there should be an overall plan regarding the features that need to be present in the final end product, as well as the path necessary to achieve those features. NiCE developers should focus on first creating the features that could provide the most benefit to SHARP code users.

NiCE's current support for SHARP (and the accompanying documentation) also gives the impression that the NiCE developers have a somewhat limited understanding of the SHARP codes. While it's not necessary that they be experts, some degree of comprehension is essential to efficiently develop supporting utilities that are useful. Efforts should be made to better understand the SHARP codes, with a focus on the aspects of these codes that make them unique from many other codes available for nuclear reactor analysis. Other general recommendations for the NiCE developers include:

- Communicate more with the developers of the other SHARP codes regarding the utilities needed.
- Be as specific as possible in the NiCE documentation when describing the individual NiCE utilities.
- Keep in mind that the users of the individual SHARP physics codes may not have the same experience level or needs, and therefore the support needed for these codes may not be the same.
- Develop the NiCE utilities based off of the SHARP code manuals, rather than example problems, and keep in mind that there is a great deal of flexibility and variety in the information users may need to provide in the input files.
- Ensure that any output analysis tools are not dependent upon the simulation being run from within NiCE.

Overall, having a supporting utility such as NiCE could be very helpful in SHARP code usage. In general, users benefit from tools that allow them to easily build inputs, tools that can

reduce redundancies in the analysis process (such as creating several different input models for the same core design), and tools which help them easily transfer information from one code to another. Specific capabilities which could be developed in NiCE in support of the SHARP codes include:

- Integration of RGG so that RGG information (material assignments, pin layout within a lattice, etc.) are communicated to PROTEUS, and the other SHARP physics codes.
- Integration of other supporting utilities such as MOAB's *mbconvert*.
- Support for SHARP coupled simulations by performing input consistency checks, developing utilities to reduce the redundant input that must be provided to each of the physics codes, and developing output analysis tools which allow the user to evaluate the relationships between the results of the neutronics, thermal-hydraulic, and structural simulations (including the way these results change with time during a transient).
- Support for evaluating the same design with both PROTEUS and the conventional reactor physics code DIF3D/REBUS.
- Integration of MC^2-3, which is used for generation of cross-sections for PROTEUS and DIF3D.
- Development of a result analysis toolkit which allows for evaluation of the high-resolution solution information, rather than just the specific data considered in the current Reactor Analyzer.
- Development of utilities to translate solution data from the format output by the SHARP codes to the format needed by NiCE. These utilities should be capable of getting the necessary information from the code output files, so that the output analysis tools are not limited to jobs that were run from within NiCE.

7 Mesh Generation Tools (RGG)

The Reactor Geometry Generator (RGG) is an open source toolkit for creating nuclear reactor assembly and core geometries and meshes. RGG is one tool within the mesh generation library MeshKit [40]. MeshKit in turn is one of several interfaces and tools developed under SIGMA (Scalable Interfaces for Geometry and Mesh-based Applications) at ANL [41]. RGG consists of two tools: AssyGen, a utility for creating individual assembly mesh files, and CoreGen, a utility for compiling the component assemblies into a full reactor core mesh [42]. RGG utilizes CUBIT, a software toolkit for generation of two- and three-dimensional finite element meshes [20]. A GUI for RGG is also currently under development by the company Kitware, and an initial version of the GUI was released recently [43].

Overall, it is very helpful for SHARP to offer tools for mesh generation. Users greatly appreciate utilities that allow them to generate inputs more quickly. In addition, many new SHARP users may not have any previous experience with mesh generation. RGG, which is much simpler to learn than complex finite element meshing toolkits such as CUBIT, allows these inexperienced users to begin performing simulations more quickly. It would be useful for SHARP development to continue to include the development of meshing tools, with a focus on specific functionalities that could benefit users of the different SHARP multi-physics codes.

Throughout the evaluation process, both the RGG documentation and the RGG code itself underwent frequent updates and revisions. Every effort has been made to assess the latest versions, but it is possible that some features have changed since they were evaluated for this discussion.

7.1 RGG and MeshKit

7.1.1 Documentation and Code Developer Support

RGG suffers from a lack of documentation (on both code installation and usage), which makes it difficult for analysts to use this toolkit. RGG has numerous code dependencies on third-party software, and these other codes must be installed prior to installation of RGG (which is installed within MeshKit). When the evaluation process began, the RGG installation instructions provided by the RGG developers were very out of date, and they did not clarify key information such as incompatibilities among versions of the supporting codes. Many days were therefore wasted trying to install MeshKit and the supporting software, without success.

Additional documentation on RGG and MeshKit has been recently developed, but installation is still very poorly described. For example, the SIGMA website's instructions for installing MeshKit and its dependencies includes installation of several codes (Triangle, Zoltan, Tetgen, etc.) that are not listed as dependencies in the README file distributed with RGG. There is no clarification of why the user may need these codes. This page also makes no mention of CUBIT or installing CGM with a CUBIT dependency. Therefore, it does not appear that these are instructions for installing MeshKit for RGG use. This is never stated however, and furthermore the website does not appear to provide separate instructions for installing MeshKit for RGG. There are alternate installation instructions provided in the RGG README file, but these appear to be out of date. In general, the documentation on MeshKit installation should explicitly state the installation process, including the specific version number of each supporting code, and should clarify how the process differs for different MeshKit purposes. It would also be helpful if the

documentation mentions issues frequently encountered in the installation process, and how to resolve them. Beyond the development of improved documentation, it would be even more helpful if a utility was developed to perform the installation process (including the installation of all dependencies) for users.

It would also be useful for the MeshKit documentation to briefly explain what the code dependencies are, and why the user may need them. Of course, each of these codes has its own documentation. However, when there are ten or more dependencies, some of which are not always necessary, it is not reasonable to expect a new MeshKit user to pursue finding documentation on each of these codes, or to expect them to know which codes are needed in a particular case. The MeshKit documentation should provide summary information on these codes, in the context of their application within MeshKit. This will help users make more informed decisions regarding whether they need each of these codes for their analysis purposes.

Overall, the MeshKit documentation appears to assume that users are already fairly familiar with other meshing tools (including all of the code dependencies). Many terms are also used without explanation. For example, on the RGG webpage, the AssyGen is description states "The AssyGen tool is the first step of the three-step core mesh creation process, it reads an input file describing a reactor assembly lattice and generates an ACIS or OCC -based geometry file". However, there is no additional discussion of what ACIS and OCC mean, or what the differences are between these geometry file types. From a user perspective, it would be very helpful to have brief discussions of background information like this within the code manual, so that the user doesn't have to search for external references. Some SHARP users may not have any previous experience with mesh generation, and they will find additional clarification very helpful.

The RGG documentation also does not provide enough information on code usage. There is a brief definition of each input variable, but little information on how these variables are actually applied. In some instances, there is not even sufficient information on how to specify the variable in the input file. For example, for the variable NeumannSet, the RGG documentation states "It can take values 'top', 'bot', or 'side' ". However, to use the keyword "side", the user must also input the equation for the line that forms the particular side (by providing the a, b, c, and d values of the line equation ax+by+cz=d). This is not indicated anywhere in the RGG documentation, but was learned through emails with the RGG developers.

As with PROTEUS, it also would be helpful if the RGG documentation gave some basic information on CUBIT. In particular, the documentation should briefly discuss the CUBIT files (*filename*.jou, *filename*.template.jou, and *filename*.sat) created by AssyGen.

It would also be useful to have more documentation connecting RGG with the SHARP physics codes. For example, to create a mesh for PROTEUS, the individual assemblies must be created with the flag "CreateSideSet no". However, the only information in the RGG manual on this variable is "Option to disable sideset creation for CUBIT journal file. Takes values 'yes' or 'no'". This provides users with no information on the purpose of this variable, and it is never stated that this variable must be set to "no" for a PROTEUS mesh. In general, if RGG is meant to be a utility within the SHARP package, there should be documentation (either in the RGG manual or the individual SHARP physics code manuals) on how to correctly use RGG for each of the codes. Many analysts who are new to using RGG will be new to using the other SHARP codes as well, and they would greatly benefit from this type of information.

The RGG developers were helpful during the evaluation process. They were willing to meet whenever necessary, both to provide an introduction to the code and later to resolve issues. They responded quickly to emails, and provided clarification when asked. However, they did not do a good job of providing detailed installation instructions, which delayed the evaluation process. They also did not install RGG (and the necessary supporting software) on any machines within ANL's Nuclear Engineering Division. Unfortunately, this meant that the only usable version of RGG was an installation in one of the code developer's directories on the MCS machines. This caused significant problems because the code developers were constantly modifying this installation. RGG is of course still under development, and it is understandable that the code developers are primarily focused on continuing to improve their toolkit. However, it would be good for the project managers to encourage more of a dedicated effort towards user support, in addition to code development. This would go very far in developing a larger user base for RGG.

7.1.2 Usability and Ability to Meet User Needs

Overall, RGG provides a great deal of benefit to users who are evaluating standard LWR and SFR geometries. For example, in the EBR-II analysis discussed in Section 3.1, it would have taken much longer to develop the mesh files without RGG, and the process would have required more familiarity with CUBIT. The well-defined structure of the RGG input files also made it easy to develop a script for creating the PROTEUS assignment file.

As mentioned in the discussion of NiCE, the analysis process could be further enhanced by the development of tools that can utilize the information in the RGG inputs for the SHARP physics codes. Examples include using the material information in the RGG input files as a starting point for developing the PROTEUS assignment file, and tracking the pin and assembly layout information for use in evaluating power peaking factors in meshes where repeated geometries all have the same region number.

There are both advantages and disadvantages to RGG's use of CUBIT. CUBIT is a robust, stable toolkit. It allows code users to view both individual assembly and full-core models in a flexible, three-dimensional view that is much more sophisticated than tools like VisEd for MCNP or the VisIt mesh viewing capabilities available with DIF3D. However, one drawback to CUBIT is that it may be difficult for some RGG users to obtain this software. According to the CUBIT website, there is no charge for U.S. government-use licenses of CUBIT, but academic and commercial users of the code must obtain licenses from the private company Computational Simulation Software LLC (csimsoft).

One of the biggest issues with the RGG code itself is a lack of detailed error messages. There are many simple error checks that code developers could implement that would be very helpful for users. For example, if coregen cannot find one of the assembly models, it currently prints "Couldn't read mesh file. Failed to load meshes." This gives no indication of which assembly mesh file it could not load. It also does not clarify whether there was a problem with the file or it couldn't find the file. In general, it would be good to establish a series of checks for RGG to perform before it constructs a full core model, and to indicate in any error messages exactly which assembly file failed the checks. RGG developers could also work with the developers of the SHARP physics codes to establish other error or warning messages for mesh features that would be an issue in simulations. Error messages provide tremendous benefit to code users because they can prevent users from having to spend hours (or even days) trying to pinpoint why something went wrong.

As another example, coregen sometimes prints "Warning: acis file has sequence numbers!". This message may print multiple times, implying that it is being issued for multiple assembly files. However, there is no way to determine which of the assembly files it refers to. Furthermore, this message is not discussed at all in the RGG documentation, making it impossible for the RGG user to tell if this is actually a problem. Similarly, whenever CUBIT is run with one of the RGG-generated journal files to create an assembly model, it will terminate with "ERROR: Errors found during the CUBIT session" whenever the assembly model has sidesets. RGG developers have confirmed via email that this is a known issue that can be ignored, but this is not mentioned anywhere in the RGG documentation, which could easily lead users to waste a great deal of time trying to determine whether there is a problem.

It also seems like the code developers may not have good in-team checks in place for identifying errors as they modify the code. As mentioned above, the EBR-II analysis was limited because the RGG utilities used to build the model were constantly being changed by the developers, which occasionally introduced issues. It would be good for the RGG developers to establish a set of verification models that they rebuild and examine each time changes are made to the code. These checks should confirm that regions and sidesets are correctly labeled, that the mesh has the intended degree of refinement, and so on.

7.2 RGG GUI

The initial release of the RGG GUI occurred very recently, and only limited testing has been performed with the GUI. The intended features of the GUI include creating AssyGen and Coregen inputs from scratch, running AssyGen and CoreGen, loading existing AssyGen and CoreGen input files, and visualizing the final RGG-created mesh file.

Overall, the GUI could enhance the RGG user's experience. Currently, the user can only visualize an RGG-created full-core model by first running RGG and then converting the .h5m mesh to an exodus file, which can then be viewed in CUBIT. The new RGG GUI now allows users to visualize the same core model without having to first run RGG. This is a much faster way to confirm the core model will be built as anticipated.

One significant issue is that there is not good consistency between the documentation provided by the RGG GUI developers, and the SIGMA team, the developers of RGG and MeshKit. Most importantly, the two are not consistent in their definition of RGG. The GUI documentation states that RGG "is an open-source graphical user interface (GUI) tool" and that AssyGen and CoreGen are tools utilized by RGG. The SIGMA team's website, on the other hand, states that AssyGen and CoreGen are utilities within RGG, and that the GUI is a supporting utility for RGG. This inconsistency is present throughout the documentation. For example, both the Kitware documentation and the SIGMA team website provide information that they each refer to as "RGG installation instructions". However, Kitware's instructions are for installing the GUI, while the SIGMA team's instructions are for installing the MeshKit package. Overall, this is a minor detail, but referring to two separate entities (a code and the GUI that supports the code) both by the name "RGG" could easily lead to confusion. There should be a consistent definition of "RGG".

The GUI development team was helpful during the evaluation process. They responded quickly to emails, and were very persistent in resolving a set of (never before seen) error messages that occurred when the GUI was installed on an NE laptop.

7.3 ART's Requirements for RGG

RGG offers a very useful method for easily creating models of standard SFR and LWR geometries. However, the code's usability is hampered by the lack of documentation on installation and usage, and by a lack of error checks in the code structure. The following provides a summary of the topics that should be added to RGG documentation:

- Installation
  - Brief overview of each of the code dependencies
  - Clarification of when/why particular dependencies are needed
  - Explicit instructions for installing MeshKit (including the differences in installing MeshKit for RGG vs. MeshKit for other purposes)
  - Clarification of the required specific versions of each of the supporting codes
  - Explanation of terms used in the current documentation that are not commonly known, such as the difference between ACIS and OCC
- Usage
  - More detailed explanation of the input variables and how they are used by RGG
  - Instructions on using RGG to create meshes for the individual SHARP physics codes
  - Explanation of RGG error messages
  - Discussion of CUBIT in the context of RGG

In addition to improved documentation, the following items should also be pursued:

- Improved verification checks within the RGG development team to ensure that errors are not introduced with code changes
- Improved error checks in the AssyGen and CoreGen code structures, for both
  - Issues that will cause a problem at some point in the mesh creation process
  - Issues that will not prevent mesh creation, but will be a problem if the mesh is used with a SHARP physics code
- Increased detail in current error messages (e.g., if the error occurs with one of several assembly files being used in a core model, the error message should clarify which assembly file)
- Development of utilities which communicate RGG information (material assignments, pin layout within a lattice, etc.) to PROTEUS, or the other SHARP physics codes as needed

8 Summary of ART's Requirements for SHARP

ART engineers have been given the opportunity to use the SHARP advanced modeling and simulation tool kit for nuclear reactors, and have provided a detailed evaluation of their utility for the analysis needs of the program. This section provides a summary of those items deemed most important.

The strengths of SHARP are the accuracy of its physics models, the capability to model arbitrary geometries, emphasis on validation, and scalability on high performance computers. The NEAMS organization has taken the right approach by selecting and further developing codes with advanced physics models; without such improved models there would be little incentive for engineers to invest in learning new tools. While most of the engineers performing this evaluation are accustomed to performing simulations on desktop workstations, SHARP's performance on parallel machines will certainly be advantageous in the future as computing power grows exponentially. NEAMS has also invested in validation of their models, primarily by analyzing experiments that focus on a single phenomenon, and provided results to the community. Further progress in each of these are certainly possible, but the overall philosophy seems to be correct.

Although the ART engineers agree that the primary code development effort must be built on a foundation of accurate models and efficient algorithms, the tools will not be adopted by engineers unless there is a relatively straightforward process to build models and extract meaningful results. The ART end-users encountered several challenges while trying to learn how to use the codes, and often found the workflow tedious. The end-users would like to make the following recommendations regarding code use:

- Significant updates are required to the documentation of both the SHARP physics codes and the supporting elements. There is also a gap between the documentation of the supporting tools (e.g. RGG, VisIt) required for model development and post-processing, and the physics codes themselves, where neither set of manuals provide user guidance on code-specific issues. More importantly, there is no documentation to describe the procedure for performing multi-physics simulations with SHARP, which would be essential for its adoption.
- The complex installation process for SHARP and supporting elements, especially MOAB and RGG, are providing a substantive hindrance to new users interested in using SHARP. The recent version of MOAB and RGG has yet to be installed on workstations in Argonne's Nuclear Engineering Division.
- The SHARP suite of tools provides highly resolved information about the reactor, but analysts are frequently interested in larger scale core characteristics—such as power peaking factors, average channel or assembly outlet temperature, maximum cladding temperature in each fuel pin. The required information is available from the SHARP solution, but the analyst would need to integrate the solution data. One possible solution would be to develop a utility that works with RGG to recognize common units like pins and assemblies, so that data can be integrated over pins and assemblies later. More advanced tools for integrating over arbitrary regions would be beneficial for other applications.
- Both Nek5000 and Diablo impose strict requirements for the types of mesh that can be used. Although such meshes may produce more accurate results, they are usually time-

consuming for most analysts to generate for complex geometries. The NEAMS team should respond with a plan for reducing this barrier to entry for Nek5000, e.g. by allowing more flexible meshing, or by furthering the development of MeshKit and RGG, or perhaps by working with commercial vendors for mesh generation to recommend tools to new users.

- Each of the SHARP physics modules need additional error checks to insure consistent user input. This will facilitate the process for model development, and ensure that analysts are using the codes properly.
- The SHARP team must be able to provide technical support to their outside users. Learning to use the code is essentially impossible without close interaction with the developers.

The ART program is making significant investments to understand and quantify phenomena related to structural core deformation in SFRs. If advanced modeling and simulation tools can provide a higher degree of confidence in the prediction of the reactor's response to these structural deformations, we have the potential to reduce capital costs for new SFR concepts, while maintaining safety performance. Furthermore, we may have a stronger position when advocating for the inclusion of radial core expansion feedback—and corresponding changes to core restraint design—with our international partners. By performing integrated neutronics, thermal-hydraulics, and structural mechanics simulation on relevant length- and time-scales to explicitly treat the important phenomena, the SHARP tools are demonstrating the potential to fulfill this modeling need. In particular, PROTEUS is unique in its capability to perform large-scale deterministic multi-group neutron transport simulations on arbitrary reactor geometries—including SFRs with deformed fuel subassemblies. Before SHARP can be relied upon for design and licensing needs, IT will need to be thoroughly validated against experimental data, such as the FFTF data currently being recovered by ART. Additional modeling capabilities and code features would also be required. The list of requirements for the particular problem of radial core expansion includes:

- For safety analysis, quasi-static simulations are insufficient. A time-dependent capability requires a significant development effort in PROTEUS, which includes only an adiabatic kinetics feature, and improved quasi-static is likely necessary.
- The calculation procedure for performing multi-physics SHARP simulations, especially with the structural mechanics code Diablo, must be automated so that they can be performed in a practical manner.
- Because the fuel, cladding, and coolant are homogenized inside the ducts of the fuel assembly, isotope densities within the homogenized region must be adjusted to conserve mass as the deformations cause volume changes. Currently, these adjustments do not account for thermal expansion of the fuel, which should be considered in future work.
- PROTEUS lacks the capability to predict photon transport, and a gamma heating model would be necessary for accurate simulations. Photon transport would be useful for other applications, too, including as estimating the flux at a detector location.
- The spatial mesh for all three physics modules was relatively coarse, as the objective was for demonstration rather than validation. Further work should be performed to provide simulations for a spatially-converged mesh, which would also provide insight into the

computational expense of running these simulations and test the RGG mesh generation tool. Similar studies are required to ensure that the multigroup cross section library is sufficient.

The effort's scope was expanded to include an evaluation of PROTEUS, Nek5000, and Diablo as single physics codes for other applications. Additional recommendations for the physics modules include:

- Each of the SHARP physics codes (PROTEUS, Nek5000, and Diablo) require further validation before they can be relied upon for nuclear reactor analysis. In particular, the PROTEUS code requires significant verification and validation to ensure its predictions are reliable for a range of reactor physics problems. In addition, multi-physics SHARP simulations require verification (i.e. to ensure the correct data is being transmitted between code modules) and validation (i.e. to ensure that integral reactor behavior is predicted correctly).
- Coupling PROTEUS to a depletion code would enable analysts to predict how the detailed flux profile affects burnup, and would provide depleted compositions to PROTEUS without a multi-stage simulation.
- The computational expense of Large Eddy Simulations is often significantly greater than the RANS techniques offered by commercial tools like STAR-CCM+. The continued development of the unsteady RANS capability in Nek5000 would enhance its appeal to ART's needs.
- Irradiation-induced swelling and creep models for Diablo would be required to accurately represent structural deformation in nuclear reactor cores.
- The ART engineers were able to access the full source code for PROTEUS and Nek5000, but were prohibited from accessing the Diablo source code. The commercial structural mechanics codes that ART engineers use do not provide access to their proprietary source code, either. However, they do provide analysts with limited means to add new capabilities (like irradiation-induced swelling and creep models) through user-developed code. If it is not possible for ART engineers to have full access to the Diablo source code, something like a user-code capability should be provided so that engineers can make minor augmentations without burdening the Diablo developers.

The ART end-users hope that this feedback will be constructive, and help guide the NEAMS program towards development efforts that will be useful to engineers performing SFR analysis. This report should mark the beginning of a collaborative effort, not just the end of an exercise.

9 References

1. A. Siegel, T. Tautges, A. Caceres, D. Kaushik, P. Fischer, G. Palmiotti, M.A. Smith, J. Ragusa, "Software Design of SHARP," in Proceedings of the Joint International Topical Meeting on Mathematics and Computations and Supercomputing in Nuclear Applications (M&C + SNA), American Nuclear Society, April 2007.
2. D. Parsons, J.M. Solberg, R.M. Ferencz, M.A. Havstad, N.E. Hodge, and A.P. Wemhoff, "Diablo User Manual", Lawrence Livermore National Laboratory report UCRL-SM-234927, Sept. 2007.
3. E.R. Shemon, M.A. Smith, C. Lee, A. Marin-Lafleche, "PROTEUS-SN User Manual: Revision 1.1", ANL/NE-14/6, August 2014.
4. E.R. Shemon, M.A. Smith, C. Lee, "PROTEUS-SN Methodology Manual", ANL/NE-14/5, June 2014. .
5. P.F. Fischer, J.W. Lottes, S.G. Kerkemier, Nek5000 Web Page, http://Nek5000.mcs.anl.gov, 2008.
6. T.J. Tautges, R. Meyers, K. Merkley, C. Stimpson, C. Ernst, "MOAB: A Mesh-Oriented Database", Sandia National Laboratories report SAND2004-1592, April 2004.
7. R. Jain, T.J. Tautges, "RGG: Reactor Geometry (and Mesh) Generator", presented at International Congress on the Advances in Nuclear Power Plants, Chicago, 2012.
8. E.R. Shemon, C. Lee, "PROTEUS Training Summary", ANL-ARC-286, April 2014.
9. A. Obabko, "NEAMS Nek5000 Training", ANL-ARC-287, August 2014.
10. Y.I. Chang, P.J. Finck, C. Grandy, Advanced Burner Test Reactor Preconceptual Design Report" Argonne National Laboratory report ANL-ABR-1 (ANL-AFCI-173), Sept. 2006.
11. E.R. Shemon, J.J. Grudzinski, C.H. Lee, J.W. Thomas, "Specification of the Advanced Burner Test Reactor Coupling Demonstration Problem: ARC Multi-Physics Coupling Demonstration", ANL-ARC-284, January 2014.
12. E. Merzari, E.R. Shemon, J.W. Thomas, A. Obabko, R. Jain, V. Mahadevan, T. Tautges, J. Solberg, R. Ferencz, R. Whitesides, "Multi-Physics Demonstration Problem with the SHARP Reactor Simulation Toolkit", ANL-ARC-283, January 2014.
13. E. Merzari, E.R. Shemon, Y. Yu, J.W. Thomas, A. Obabko, R. Jain, V. Mahadevan, J. Solberg, R. Ferencz, R. Whitesides, "Full Core Multi-Physics Simulation with Offline Mesh Deformation", April 2014.
14. J.J. Billings, et al, "NiCE User Manual", https://sourceforge.net/projects/niceproject/, 2014.
15. H. Childs et al, "VisIt: An End-User Tool for Visualizing and Analyzing Very Large Data", High-Performance Visualization – Enabling Extreme-Scale Scientific Insight, p. 357-372, Oct. 2012.
16. J.J. Billings, J. Deyton, A. Wojtowicz, "Demonstration of NiCE User Environment Customized for ARC Users (revised)", April 2014.

17. J. J. Grudzinski, C. Grandy, "Fuel Assembly Bowing and Core Restraint Design in Fast Reactors", Proceedings of ASME 2014 International Mechanical & Engineering Congress & Exposition (IMECE 2014), Montreal, Canada, November 2014.
18. T. H. Fanning, ed., "The SAS4A/SASSYS-1 Safety Analysis Code System", ANL/NE-12/4, Nuclear Engineering Division, Argonne National Laboratory, January 31, 2012.
19. T. Fei, A. Mohamed, T.K. Kim, "Neutronics Benchmark Specifications for EBR-II Shutdown Heat Removal Test SHRT-45R-Revision 1", ANL-ARC-228, January 2013.
20. Sandia National Laboratory, CUBIT Website, https://cubit.sandia.gov/ , 2010.
21. C.H. Lee and W. S. Yang, "MC^2-3: Multigroup Cross Section Generation Code for Fast Reactor Analysis", ANL/NE-11-41, September 2011
22. T.P. Gerrity III, "MCODE-3: Time-Dependent Depletion Isotopics with MCNP-5 and SCALE-6.1", Massachusetts Institute of Technology, 2012.
23. X-5 Monte Carlo Team, Los Alamos National Laboratory, "MCNP – A General Monte Carlo N-Particle Transport Code, Version 5; Volume II: User's Guide", LA-CP-03-0245, February 2008. .
24. W.D. Pointer, et al, "Simulations of Turbulent Diffusion in Wire-Wrapped Sodium Fast Reactor Fuel Assemblies", 2009 International Conference on Fast Reactors and Related Fuel Cycles (FR09), Kyoto, Japan, December 7-11, 2009.
25. E. Merzari, W. D. Pointer, J.G. Smith, A. Tentner, P. Fischer, "Numerical Simulation of the Flow in Wire-Wrapped Pin Bundles: Effect of Pin-Wire Contact Modeling", *Nuclear Engineering and Design*, **253**, p. 374-386, 2012.
26. Y. Yu, S. Aithal, A. Obabko, J. Walker, J.W. Thomas, "CFD Validation with the Sodium Triple-Jet Mixing Experiment PLAJEST: STC-A7 CEA/JAEA/DOE Collaboration on SFR Thermal-Hydraulics", ANL-ARC-289, July 2014.
27. K.M.Becker, C.H.Ling, S.Hedberg, G.Strand, "An Experimental Investigation of Post Dryout Heat Transfer," Department of Nuclear Reactor Engineering, Royal Institute of Technology, KTH-NEL-33, Sweden, 1983.
28. CD-adapco, Ltd., "STAR-CCM+ User's Manual", 2014.
29. F.E. Dunn, Cahalan, J.E., "Whole Core Sub-Channel Analysis Verification with the EBR-II SHRT-17 Test", ICAPP'06, Reno, NV USA, June 4-8, 2006.
30. T.H Fanning, et al, "Multi-Resolution Modeling of Subassembly Pin Bundles for Advanced Fast Reactor Safety Simulations", M&C 2009, Saratoga Springs, New York, May 3-7, 2009
31. E. Merzari, et al, "Hydrodynamic Analysis of Different Spacing Strategies for a Tight-Lattice SFR Bundle", *Transactions of the American Nuclear Society*, **103**, November 2011.
32. A. V. Obabko, A. E. Aksenova, V. V. Chudanov, P. F. Fischer, V.M. Goloviznin, A.A. Kanaev, S. Karabasov, E. Merzari, S. Parker, V. A. Pervichko, W. D. Pointer, T. Taugets, M. A. Zaytsev, "Cross-verification and validation simulations of Matis benchmark," ANL/TM-341, 2014 (in preparation)

33. S. M. Aithal, P. F. Fischer, O. Marin, E. Merzari, A. V. Obabko, Y. Yu, and W. D. Pointer, "LES and RANS simulations of the ANL MAX experiment for various inlet boundary conditions," ANL/MCS-TM-337, September 30, 2013.
34. S. Lomperski, E. Merzari, and D. Pointer, P. Fischer and A. Obabko, "MAX Startup Tests," ANL-NE 10/28, September 2010.
35. Lysmer, J., and Kuhlemeyer, R. L., "Finite Dynamic Model for Infinite Media," Journal of the Engineering Mechanics Division, ASCE, pp. 859-977, August, 1969.
36. ASCE 4-98, "Seismic Analysis of Safety-Related Nuclear Structures and Commentary", ASCE Standard, American Society of Civil Engineers.
37. Tang, Y and Solberg, J., 2014, private communication.
38. Wong, H.L., and Luco, J.E., "Tables of Impedance Functions and Input Motions for Rectangular Foundations", Report No. CE 78-15, University of Southern California, Los Angeles, CA, 1978.
39. A. R. Bennett and J. Billings, SHARPER wiki, http://sourceforge.net/p/sharper/wiki/Home/ .
40. SIGMA Development Team, MeshKit Website, http://sigma.mcs.anl.gov/meshkit-library/, 2014.
41. SIGMA Development Team, SIGMA Website, http://sigma.mcs.anl.gov/ , 2014.
42. SIGMA Development Team, RGG Website, http://press3.mcs.anl.gov/sigma/meshkit-library/rgg/ , 2014.
43. Kitware, Inc.,CMB RGG Website, http://www.computationalmodelbuilder.org/rgg/ , 2014.

Nuclear Engineering Division
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 208
Argonne, IL 60439

www.anl.gov

Argonne National Laboratory is a U.S. Department of Energy
laboratory managed by UChicago Argonne, LLC