ORNL/LTR-2014/382

Enhancements to the SHARP Build System and NEK5000 Coupling

Alexander J. McCaskey
Andrew Bennett
Jay Jay Billings

September 2014

Approved for public release;
distribution is unlimited.

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

ORNL/LTR-2014/382

Computer Science and Mathematics Division

ENHANCEMENTS TO THE SHARP BUILD SYSTEM AND NEK5000 COUPLING

Alexander J. McCaskey
Andrew Bennett
Jay Jay Billings

Date Published: September 2014

Prepared by
OAK RIDGE NATIONAL LABORATORY
P.O. Box 2008
Oak Ridge, Tennessee 37831-6285
managed by
UT-Battelle, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

The SHARP project for the Department of Energy's Nuclear Energy Advanced Modeling and Simulation (NEAMS) program provides a multiphysics framework for coupled simulations of advanced nuclear reactor designs. It provides an overall coupling environment that utilizes custom interfaces to couple existing physics codes through a common spatial decomposition and unique solution transfer component. As of this writing, SHARP couples neutronics, thermal hydraulics, and structural mechanics using PROTEUS [13], Nek5000 [7], and Diablo [6] respectively. This report details two primary SHARP improvements regarding the Nek5000 and Diablo individual physics codes: (1) an improved Nek5000 coupling interface that lets SHARP achieve a vast increase in overall solution accuracy by manipulating the structure of the internal Nek5000 spatial mesh, and (2) the capability to seamlessly couple structural mechanics calculations into the framework through improvements to the SHARP build system.

The Nek5000 coupling interface now uses a barycentric Lagrange interpolation method that takes the vertex-based power computed from the PROTEUS neutronics solver and maps it to the user-specified, general-order Nek5000 spectral element mesh. Before this work, SHARP handled this vertex-based solution transfer in an averaging-based manner. SHARP users can now achieve higher levels of accuracy by specifying any arbitrary Nek5000 spectral mesh order. This improvement takes the average percentage error between the PROTEUS power solution and the Nek5000 interpolated result down drastically from over 23% to just above 2%, and maintains the correct power profile.

We have integrated Diablo [6] into the SHARP build system to facilitate the future coupling of structural mechanics calculations into SHARP. Previously, simulations involving Diablo were done in an iterative manner, requiring a large amount manual work, and left only as a task for advanced users. This report will detail a new Diablo build system that was implemented using GNU Autotools [5], mirroring much of the current SHARP build system [3], and easing the use of structural mechanics calculations for end-users of the SHARP multiphysics framework. It lets users easily build and use Diablo as a stand-alone simulation, as well as fully couple with the other SHARP physics modules. The top-level SHARP build system was modified to allow Diablo to hook in directly. New dependency handlers were implemented to let SHARP users easily build the framework with these new simulation capabilities.

The remainder of this report will describe this work in full, with a detailed discussion of the overall design philosophy of SHARP, the new solution interpolation method introduced, and the Diablo integration work. We will conclude with a discussion of possible future SHARP improvements that will serve to increase solution accuracy and framework capability.

1. OVERVIEW

The SHARP project of the Department of Energy's (DOE) Nuclear Energy Advanced Modeling and Simulation (NEAMS) program represents an effort aimed at building a comprehensive nuclear reactor simulation framework that incorporates multiple physics models across a wide array of length and time scales. SHARP has been designed with a "bottom-up" philosophy that places a strong focus on coupling existing validated and verified physics codes, in stark contrast to other well known multiphysics frameworks such as MOOSE [8] or COMSOL [1]. These other, "top-down" frameworks are designed in a way that effectively forces a rewrite of existing physics functionality to adhere to the framework's overall extensibility and coupling scheme.

To achieve its overall framework design, SHARP relies on spatial domain coupling to achieve solution transfer and data exchange from one physics module to the next. This spatial coupling is a challenge in and of itself, as each physics module may represent its underlying mesh solution data in its own internal way. To overcome this challenge and map solution data between disparate meshes, SHARP uses a tool known as MOAB (Mesh-Oriented datABase) [14], a software library from Argonne National Laboratory (ANL) that lets users create, query, and modify mesh data, as well as assign field data to the various elements of that mesh.

As shown in Fig. 1, MOAB provides the backbone for the entire SHARP framework. It connects the various physics modules available with a common mesh format and custom coupling package called MBCoupler, which provides the MOAB-based solution transfer mechanism. This lets SHARP reuse existing physics codes in an extensible manner, with the bulk of the work going toward the development of coupling adapters that map each physics module's internal mesh representation to and from the MOAB mesh format. To enable this mesh data transfer, SHARP also provides a coupling library, COUPE [11], that drives the execution and solution transfer of each underlying physics component by interacting with the custom adapters and MBCoupler.

By default, SHARP couples neutronics and thermal hydraulics in an operator split fashion, i.e., a neutronics solve is executed for a given time step and the result is transferred to the thermal hydraulics code to be used in its current solve. The result of that calculation is then passed back to the neutronics physics module at the beginning of the next time step. This sequential workflow enables loose coupling between the two physics modules. However, recent work done by ANL SHARP team members has shown SHARP can handle implicit coupling schemes as well [10], thus promoting even tighter coupling between the single physics codes and enabling higher solution accuracy. For the neutronics calculation, SHARP uses PROTEUS (formerly UNIC), which uses an even-parity discrete ordinates approximation (SN2ND) to solve the steady-state neutron transport equation [13]. To compute the effects of fluid dynamics on a reactor core, SHARP uses the well-known Nek5000 computational fluid dynamics code. Nek5000 employs a spectral element–based methodology that simulates incompressible fluid flow with thermal and passive scalar transport [7].

The primary focus of this report is to detail improvements to one of these coupling interfaces, specifically the interface used by COUPE to map the Nek5000 mesh to and from the single intermediary MOAB mesh. We will show improvements to the overall solution transfer mechanism used by this interface that lets SHARP users take advantage of higher Nek5000 spectral orders, and thus, improve solution accuracy. We will also detail work done on the SHARP build system to facilitate integration with the Diablo structural mechanics code. This work will make it easer for SHARP users to take into account mesh deformations into a given multiphysics reactor simulation.

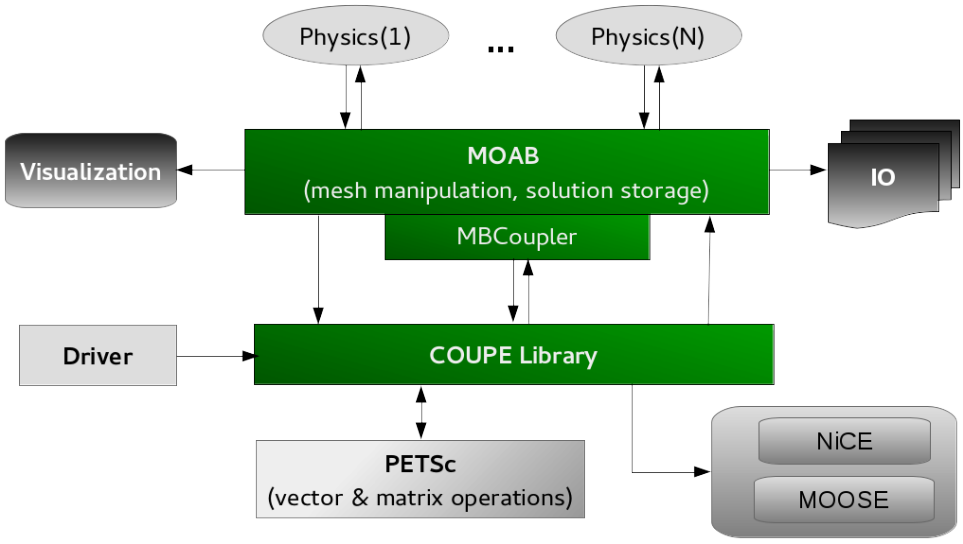

Fig. 1. Overall SHARP design, showing a strong dependency on the MOAB mesh database to promote coupling between disparate physics modules.

2. VERTEX-BASED COUPLING WITH ARBITRARY NEK5000 SPECTRAL ORDER

Even though each physics module (PROTEUS and Nek5000) computes solutions on each vertex of its internal mesh representations, the default behavior for SHARP is to transfer solutions between modules in an element-based fashion. The values of all vertex solutions on a given element are averaged and set as the solution for that element, then that information is transferred to the next physics module. This averaging causes a significant amount of data loss and leads to decreased solution accuracy. Clearly, the optimal way to transfer solution data is vertex-based, but this is challenging because each physics module represents the discretized spatial domain differently. There is not a one-to-one mapping between the vertices of each mesh. Providing a way for these vertex-based solutions to interact can drastically improve the fidelity of the simulations.

Being able to perform a vertex-based solution transfer is especially important in transferring data to Nek5000. Nek5000 relies on the spectral element method and provides users with a parameter that can manipulate the spectral order of the calculation, i.e., how many node points exist for a given element. This parameter can be set by users at runtime and provides a mechanism for increasing overall solution accuracy for a stand-alone Nek5000 calculation. SHARP must be able to map vertex data from PROTEUS to an arbitrarily large Nek5000 spectral order to let users tailor the desired accuracy of a given calculation.

Before this work, a vertex-based transfer implementation was used to map the neutronics power solution from PROTEUS into Nek5000. By default, SHARP transfers PROTEUS vertex data through an intermediary quadratic MOAB mesh (elements with 27 node points in 3 dimensions). To map that quadratic mesh data to a higher-order spectral Nek5000 mesh, a hybrid approach was implemented whereby solutions on vertices that shared locations between the two meshes were transferred vertex-wise; all other vertices were set to the element average. This method, in theory, is better than simply setting all vertices to the element averages; but in practice, it achieves only small improvements in accuracy compared with the element-based averaging (see Fig. 2). To improve the vertex-based solution transfer, we

have implemented a new projection routine that drastically increases the accuracy of the power profile interpolated by Nek5000 and can be controlled at runtime by the user-defined spectral order.

Typically in finite element analysis, this vertex-based solution transfer is accomplished through L_2 projection or an interpolation routine, with the former representing optimized average behavior and the latter producing an exact function representation only at the interpolation nodes. In this work we have chosen to transfer vertex-based solutions from PROTEUS to Nek5000 using a L_2 projection, and we will show how this simplifies to basic interpolation for Nek5000's spectral LGL basis.

2.1 L_2 Projection from a source to target mesh

Suppose we have the solution $u \in L_2(\Omega)$ computed by a source physics module over the space $\Omega \subset \mathbb{R}^d$ and wish to compute its projection onto a target mesh, $P_h u \equiv u_h \in V_h$, where P_h is a projection operator and V_h is the set of basis functions for the target mesh. This can be accomplished by minimizing the L_2 norm of $u_h - u$, which leads to the following expression:

$$\int_\Omega (u - u_h)\phi_i d\Omega = 0, \tag{1}$$

where ϕ_i are elements of V_h. This form can be discretized by noting that our desired solution $u_h \in V_h$ can be written as a linear combination of these basis functions, $u_h = \sum_{i=1}^{N} \alpha_i \phi_j$, with N the dimensionality of V_h. After injecting this linear combination and rearranging, we have that

$$\int_\Omega u\phi_i d\Omega = \int_\Omega \Big(\sum_{j=0}^{N} \alpha_j \phi_j \Big) \phi_i d\Omega, \tag{2}$$

which can be expressed in matrix-vector form $\mathbf{M}\boldsymbol{\alpha} = \mathbf{b}$, with

$$M_{ij} = \int_\Omega \phi_i \phi_j d\Omega, \tag{3}$$

and

$$b_i = \int_\Omega u\phi_i d\Omega. \tag{4}$$

Finding our projected solution u_h amounts to solving this system of equations for $\boldsymbol{\alpha}$.

2.2 Nek5000 Spectral Mesh

We must know our basis functions ϕ_i to compute the coefficient vector $\boldsymbol{\alpha}$, and thus our projected solution. For SHARP, we are taking a vertex-based solution function from PROTEUS and projecting to the Nek5000 spectral element mesh, so our V_h must be defined by the spectral basis functions used by Nek5000. The Nek5000 mesh uses the Legendre-Gauss-Lobatto (LGL) basis, which is given (in one dimension for simplicity) by

$$\phi_i(x) = \frac{-1}{n(n+1)L_n(x_i)} \frac{(x^2 - 1)L_n'(x)}{x - x_i}. \tag{5}$$

where x_i are the LGL nodal points and $L_n(x)$ is the n^{th} order Legendre polynomial. Notice that this basis set has the unique property that $\phi_i(x_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta. From this simple property, and

by employing Gaussian quadrature to compute the integrals in Eqs. 3, 4, our L_2 projection simplifies greatly to

$$b_i = \sum_j w_j u(x_j)\phi_i(x_j) = w_i u(x_i), \tag{6}$$

and

$$M_{ij} = \sum_k w(x_k)\phi_i(x_k)\phi_j(x_k) = w_i \delta_{ij}, \tag{7}$$

where w_i are the corresponding Gaussian quadrature point weights. And since $\boldsymbol{\alpha} = \mathbf{M}^{-1}\mathbf{b}$, we are led to conclude that our L_2 projection can be computed from the vector of source mesh function evaluations at the target mesh quadrature points

$$\alpha_i = u(x_i). \tag{8}$$

Since the data coming to Nek5000 are associated with the nodes of a quadratic mesh, and our Nek5000 quadrature points are at an arbitrarily higher order, these α_i are in general not available. If the user specified a quadratic Nek5000 mesh, then our solution transfer task would be a simple one-to-one mapping. But for all other spectral orders, we must compute the values of the source mesh solution at the new, general order target mesh quadrature points. We need to provide an interpolation routine to compute the α_i coefficients; thus, our projected solution vector u_h.

To calculate this interpolation we use barycentric Lagrange interpolation [4]. Suppose we are given some point $\tilde{\mathbf{x}}$ that falls inside of an element of volume V and whose vertices are defined by a set of points denoted $\mathbf{x}_i$ such that each $\mathbf{x}_i$ is associated with some input data value u_i. That is to say, in the continuous picture $u_i = u(\mathbf{x}_i)$, we can interpolate the value $\tilde{u}$ by breaking up our element into sub-volumes:

$$V_i(\tilde{\mathbf{x}}) = \prod_j |x_i^{(j)} - \tilde{x}^{(j)}| \tag{9}$$

and then construct the interpolation by a weighted sum of the sub-volumes

$$\tilde{u}(\mathbf{x}) \approx \frac{\sum V_i(\mathbf{x})\, u(\mathbf{x}_i)}{\sum V_i(\mathbf{x})}. \tag{10}$$

2.3 Results for Test Sodium-Cooled Fast Reactor: SAHEX1

We tested our new implementation using a small test problem, SAHEX1, that models a test sodium-cooled fast reactor. We found significant improvements using our updated vertex-based projection over both the element- and vertex-based averaging methods previously implemented. Even at low Nek5000 spectral orders, we found that these gains were substantial. Our tests exhibit large gains in accuracy and follow the parabolic power profile coming from PROTEUS nearly exactly. Figures 2 and 3 show this increase in accuracy very well; the previous implementations model the PROTEUS power profile as simple step functions, leading to extremely large errors relative to the incoming power solution. Figure 4 shows a cross-cut of the nuclear reactor fuel pins in three dimensions for the previous averaging implementation (left) and our new implementation (right). From this figure, and the results in Figs. 2 and 3, it is clear that the accuracy increase gained from the new projection implementation is substantial.

The new method of projection allows us to specify arbitrary spectral order for the Nek5000 simulations, allowing users to execute SHARP coupled simulations with a much finer granularity. Comparing the solutions from a third order simulation and a fifth order simulation, we do see gains in the accuracy of the projected power. All results using the new method are significantly more accurate than those using the previous method.

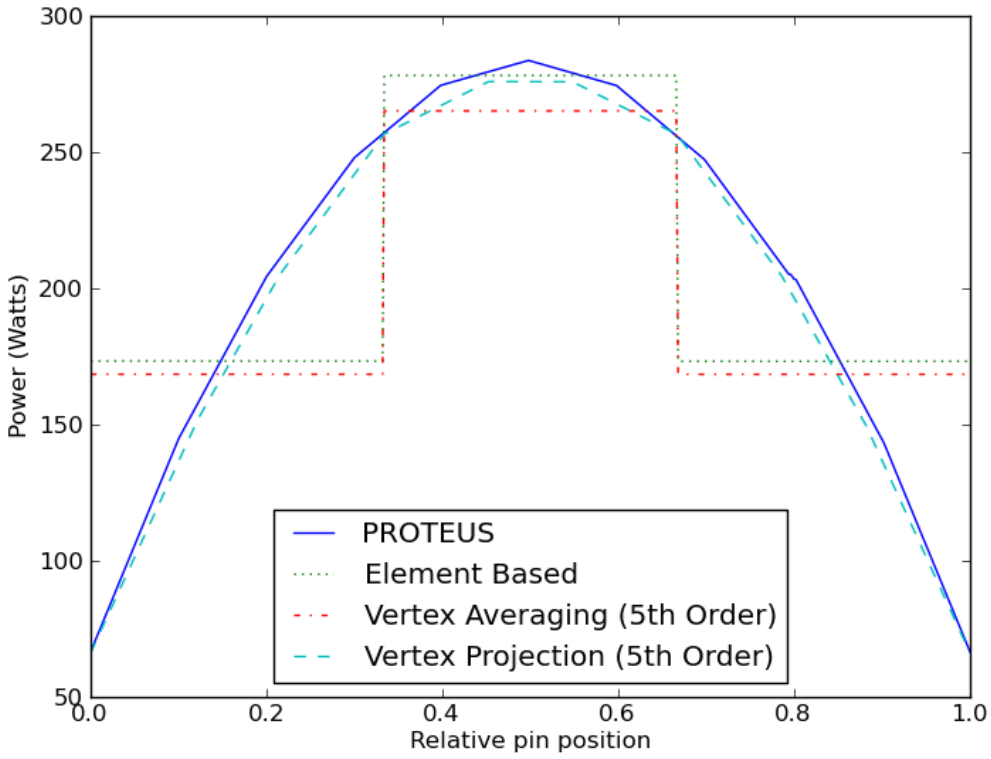

Fig. 2. Fuel pin power profile for various solution transfer implementations.

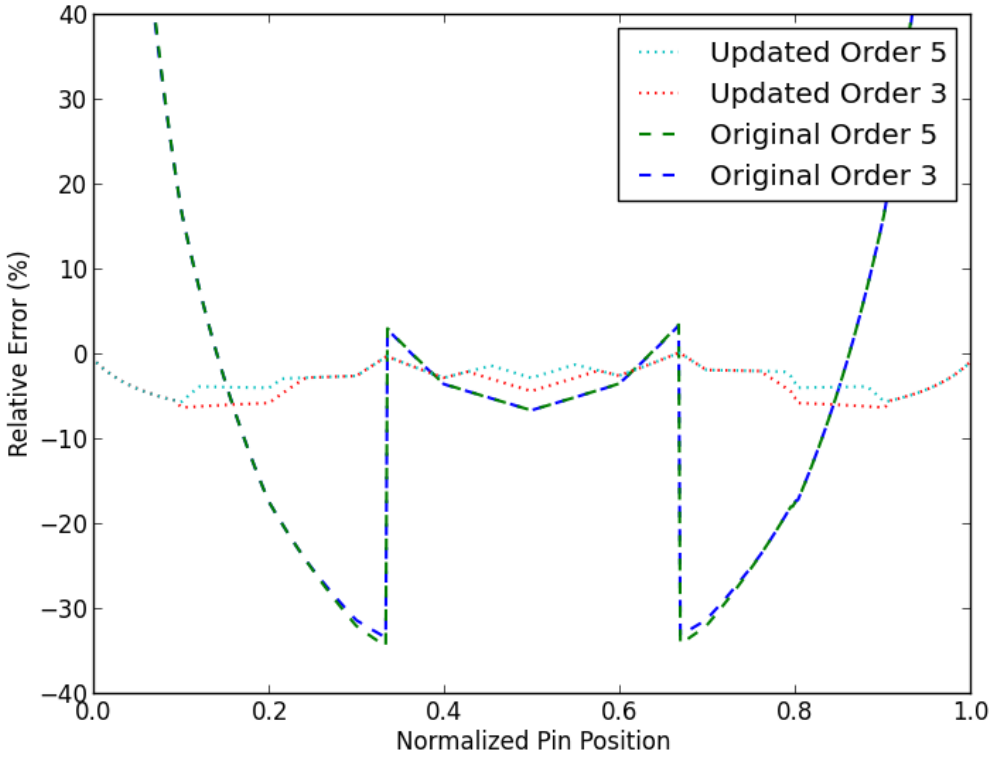

Fig. 3. Error percentage relative to incoming PROTEUS solution for various spectral orders and transfer implementations.

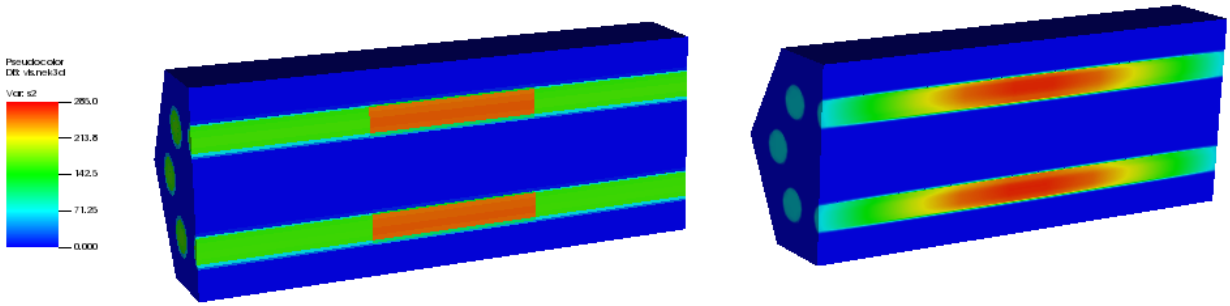

Fig. 4. Vertex-based averaging (left) versus vertex-based projection (right) implementation. The projection method provides greater accuracy and resolution.

3. INTEGRATION OF THE DIABLO STRUCTURAL MECHANICS SOLVER

The Diablo code developed at Lawrence Livermore National Laboratory (LLNL) uses implicit finite element methods to simulate nonlinear structural mechanics and heat transfer phenomena [6] and is used by the SHARP framework to calculate deformations in the reactor mesh by coupling to the temperature calculated by Nek5000. Results from preliminary simulations of the full Advanced Burner Test Reactor (ABTR) indicate that Diablo is fully capable of coupling to PROTEUS and Nek5000 via the SHARP framework [12].

The integration of Diablo into SHARP's build system required careful consideration of the overall Diablo architecture. Notably, overlapping and/or conflicting dependencies had to be addressed to maintain a smooth user experience. We have implemented a stand-alone build system for Diablo that is fully compatible with the top-level SHARP system by leveraging existing dependencies from the SHARP build system; new features have also been added to SHARP's build system to incorporate the new Diablo capabilities.

3.1 Architecture of Diablo

New implementations of the dependency handlers have been developed in the Diablo build, leading to an increase in both the robustness and the stability of the code as a stand-alone simulation tool. Previous iterations of the Diablo build system did not feature the dependency verification or user-specified options that allow configuration of Diablo to be much more flexible. Our updates also standardize the way these processes are handled, reducing the barrier to entry significantly.

Dependencies such as SILO, MILI, MUMPS, ExodusII, ScaLAPACK, PARPACK, and PWSMP were introduced to SHARP by coupling Diablo with the other physics modules. Furthermore, the existing HYPRE, NetCDF, HDF5, and ParMETIS handlers had to be modified to allow the introduction of Diablo. During the Diablo configuration, any non-standard options or code modifications are taken into account on the fly, allowing the users to focus on the actual research problems they are trying to solve, and not worrying about underlying complicated build details.

The end result of the Diablo build process is an executable and library that can be used to couple into SHARP, similar to the process by which PROTEUS is built. By default, Diablo is built out of source and uses a unique identifier to build directly into the Diablo build directory. This change from other SHARP physics modules will need to be taken into account in designing coupled drivers for simulations.

3.2 Updating the Diablo Build

Similar to the SHARP and PROTEUS build systems, the Diablo build system was implemented on top of GNU Autotools. The new system has a number of advantages over the previous iterations. Switching from the previous SCons-based system made the Diablo integration with the SHARP build system easier and more efficient. Built for LLNL machines, the SCons system featured several instances of hard-wiring paths and variable names. Rather than reworking this system, we decided that for this work we would simply re-purpose the older Autotools system, which greatly simplifies the overall maintenance of the SHARP software environment. The SCons-based system has been left in the Diablo code to ensure backward compatibility, as well as suffice those Diablo developers who find that system adequate for their needs.

Like the SCons system, the Autotools system previously in place for Diablo was not immediately compatible with the overarching SHARP build system. Several core functionalities had to be updated to directly couple all of the pieces. First, the existing Autotools system had not been updated to handle new functionality within Diablo. To bring it up to speed, we added the updated sources and integrated an up-to-date source dependency tracker to avoid adding manual updates in the future.

The second improvement we introduced was to update the general control flow of the build system. As with the SCons system, there were several references to LLNL machines, which had to be reworked to be completely general. Users not operating LLNL machines can now build Diablo without editing files directly, saving time and confusion. The most labor-intensive aspect of the integration efforts was updating the dependency requirements.

Most of the required dependencies for Diablo are included within the distribution as compressed archives. However, a structured set of instructions about how to correctly configure them to be compatible with one another was not provided to end users. To address this gap in knowledge, a definitive guide to the manual building of these dependencies was written. This guide can help to alleviate build difficulties for future users and developers (a copy of it can be found in both the code repository's main README file and on the SHARP wiki pages). Additionally, the configuration step of Diablo will now check each of the required dependencies for operability and compatibility, complete with detailed error recording. By the end of the first step of the build, a user should know whether the rest of the process will work.

Completion of these updates lets the Diablo build system seamlessly couple to the rest of SHARP's physics modules. For a consistent experience, we updated the SHARP build system to accommodate these new updates as well.

3.3 Integration into the SHARP Build

Many of the updates that were implemented in the Diablo build system needed to have SHARP-level counterparts implemented as well. As a first step, functionality to configure and build Diablo via the newly implemented system was incorporated into SHARP. Thanks to careful design considerations discussed in [3], this portion of the integration was trivial. Most of the work required to integrate Diablo into the SHARP build was in dependency handling.

Dependency handling macros were incorporated for each piece newly introduced by Diablo. The complexity of a build system grows substantially with the introduction of each new dependency. Further increasing the complexity of the build is the fact that the dependencies are not always completely independent from one another. To provide a minimally frustrating build experience, dependency handling should be hidden from users. This was accomplished by an iterative approach in which each new dependency was implemented singly, according to the level of additional complexity that it added. This

approach allowed the SHARP build system to stay relatively stable throughout the development of the Diablo integration.

The SHARP build system has been made to be dynamic, with the addition of control flow logic that builds the correct dependency stack based on the directions that the user gives during configuration. Diablo's dependencies are installed and checked only if the user wishes to use (and has access to) the Diablo source code. This flexibility keeps the build light in cases when only minimal functionality is desired, but it provides the option for robustness.

4. CONCLUSIONS AND FUTURE WORK

The new features developed for SHARP bring the project a step closer to providing high-resolution simulations of nuclear reactor cores. Improvements to the coupling help to bring higher accuracy to existing portions of the code, while the addition of Diablo to the SHARP build system will facilitate the inclusion of structural deformations into multiphysics reactor calculations. Most importantly, these improvements come with little increase in complexity from a user's perspective, meaning that SHARP remains a useful tool for expert and non-expert users alike.

Improving the SHARP-Nek5000 solution transfer mechanism increases the accuracy of the simulations and lets end-users verify the data with a higher level of confidence. The methods used to increase the accuracy of the solution transfer come with the bonus that they do not significantly increase the running time of simulations. Thanks to this improvement, users can now effectively specify the granularity with which to use Nek5000 in a coupled simulation, leading to greater accuracy and user control.

The advanced reactors that will be built in the coming decades will increasingly rely on natural phenomena, such as gravity or thermal expansion of structural materials, to ensure the safe and efficient operation of the facility. As a consequence, the optimization of the performance of these designs requires a better understanding of the interactions between structural components and the reactor core. The integration of Diablo into SHARP provides a toolset for detailed, accurate assessments of these interactions in various designs.

The improvements detailed in this report serve to increase overall SHARP usability and solution accuracy, but there is still much work that can be done to improve the SHARP multiphysics software suite. Future work should include build support for MeshKit, and specifically the Reactor Geometry and Mesh Generator (RGG) [2], to let users of SHARP easily and efficiently construct reactor meshes and utilize them in a given SHARP multiphysics execution [15]. Other future additions to SHARP will be based on the improvements detailed in this report. To take full advantage of the integration of Diablo into the build system, new coupled drivers and problem definitions will need to be developed to programmatically couple Diablo to Nek5000 and PROTEUS. The tradeoff between the accuracy of the coupling and the increase in runtime when increasing the Nek5000 spectral order will need to be explored to ensure that users get the most out of simulations.

REFERENCES

[1] COMSOL: Multiphysics User's Guide. *Version: September, http://www.comsol.com/shared/downloads/IntroductionToCOMSOLMultiphysics.pdf*, 2005.

[2] Sigma Development Team, MeshKit Website. *September, http://sigma.mcs.anl.gov/meshkit-library*, 2014.

[3] Andrew Bennett, Vijay Mahadevan, and Jay Jay Billings. Integrated Build System of the SHARP Software Suite. In *NEAMS Deliverable Report*, Oak Ridge, TN, USA, March 2014.

[4] Jean-Paul Berrut and Lloyd N Trefethen. Barycentric lagrange interpolation. *Siam Review*, 46(3):501–517, 2004.

[5] John Calcote. *Autotools: A Practitioner's Guide to GNU Autoconf, Automake, and Libtool*. No Starch Press, 2010.

[6] R. M. Ferencz. Technical Spotlight: NEAMS Structural Mechanics with Diablo. Technical report, Lawrence Livermore National Laboratory (LLNL), Livermore, CA, 2013.

[7] P. Fischer, J. Kruse, J. Mullen, H. Tufo, J. Lottes, and S. Kerkemeier. Nek5000 - Open Source Spectral Element CFD Solver. *Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, IL, see https://nek5000. mcs. anl. gov/index. php/MainPage*, 2008.

[8] Derek Gaston, Chris Newman, Glen Hansen, and Damien Lebrun-Grandié. MOOSE: A Parallel Computational Framework for Coupled Systems of Nonlinear Equations. *Nuclear Engineering and Design*, 239(10):1768–1778, 2009.

[9] V. S. Mahadevan, E. Merzari, T. Tautges, R. Jain, A. Obabko, M. Smith, and P. Fischer. High-Resolution Coupled Physics Solvers for Analysing Fine-Scale Nuclear Reactor Design Problems. *Phil. Trans. R. Soc. A*, 372(20130381), June 2014.

[10] Vijay S Mahadevan, Elia Merzari, Timothy Tautges, Rajeev Jain, Aleksandr Obabko, Michael Smith, and Paul Fischer. High-Resolution Coupled Physics Solvers for Analysing Fine-Scale Nuclear Reactor Design Problems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372(2021):20130381, 2014.

[11] E. Merzari, E.R. Shemon, Y. Yu, J.W. Thomas, A. Obabko, R. Jain, V. Mahadevan, J. Solberg, R. Ferencz, and R. Whitesides. Full Core Multi-Physics Simulation with Offline Mesh Deformation. April 2014.

[12] E. R. Shemon, M. A. Smith, C. H. Lee, and A. Marin-Lafleche. *PROTEUS-SN User Manual*. Aug 2014.

[13] T. J. Tautges, R. Meyers, K. Merkley, and C. Stimpson. MOAB: A Mesh-Oriented Database. *Sandia National Laboratory Report*, (SAND2004-1592), 2004.

[14] Justin Thomas. Advanced Reactor Technologies Work Package Manager, Argonne National Laboratory. Personal communication, September 2014.