ANL/NE-16/1

Argonne NATIONAL LABORATORY

SHARP Multiphysics Tutorials

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
Alexandria, 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.

ANL-NE-16/1

SHARP Multiphysics Tutorials

prepared by
Y. Q. Yu, E. Shemon
Nuclear Engineering Division, Argonne National Laboratory

V. Mahadevan, R. Rahaman
Mathematics and Computer Science, Argonne National Laboratory

February 29, 2016

ABSTRACT

SHARP, developed under the NEAMS Reactor Product Line, is an advanced modeling and simulation toolkit for the analysis of advanced nuclear reactors. SHARP is comprised of three physics modules currently including neutronics, thermal hydraulics, and structural mechanics. SHARP empowers designers to produce accurate results for modeling physical phenomena that have been identified as important for nuclear reactor analysis. SHARP can use existing physics codes and take advantage of existing infrastructure capabilities in the MOAB framework and the coupling driver/solver library, the Coupled Physics Environment (CouPE), which utilizes the widely used, scalable PETSc library.

This report aims at identifying the coupled-physics simulation capability of SHARP by introducing the demonstration example called sahex in advance of the SHARP release expected by Mar 2016. sahex consists of 6 fuel pins with cladding, 1 control rod, sodium coolant and an outer duct wall that encloses all the other components. This example is carefully chosen to demonstrate the proof of concept for solving more complex demonstration examples such as EBR II assembly and ABTR full core. The workflow of preparing the input files, running the case and analyzing the results is demonstrated in this report. Moreover, an extension of the sahex model called sahex_core which adds six homogenized neighboring assemblies to the full heterogeneous sahex model is presented to test homogenization capabilities in both Nek5000 and PROTEUS.

Some primary information on the configuration and build aspects for the SHARP toolkit, which includes capability to auto-download dependencies and configure/install with optimal flags in an architecture-aware fashion is also covered by this report. A step-by-step instruction is provided to help user to create their cases. Details on these processes will be provided in the SHARP user manual that will accompany the first release.

Table of Contents

Abstract ii
Table of Contents iii
List of Figures iv
List of Tables iv
1 Introduction of SHARP 1
2 SHARP Configuration and Installation 7
2.1 Package Dependencies 7
2.2 Source Management 7
2.3 Configuration Options 8
2.3.1 Enabling/disabling compile-time features or packages 8
2.3.2 Linking to existing libraries 8
2.3.3 Downloading third-party libraries 8
2.4 Installation 9
3 Demonstration Examples 11
3.1 sahex 11
3.1.1 Problem Description 11
3.1.2 Input files.. 12
3.1.3 Running…. 16
3.1.4 Analyzing.. 18
3.2 sahex_core 24
3.2.1 Problem Description 24
3.2.2 Input files.. 25
3.2.3 Running…. 26
3.2.4 Analyzing.. 27
4 Create Own Test Case 30
5 Reference 32

LIST OF FIGURES

Figure 1.1. a) Depiction of the CouPE solvers interacting with MOAB data backplane driving the standalone or coupled physics calculations. b) Sketch of global iteration. 3
Figure 1.2 Coupling and iteration process. 6
Figure 3.1 sahex problem geometry and mesh 11
Figure 3.2 NEK5000 mesh and Block number (sahex_nek.h5m) 14
Figure 3.3 DIABLO and PROTEUS mesh Block number (sahex_proteus.h5m) 15
Figure 3.4 Power profile and temperature distribution for steady state solution 18
Figure 3.5 Load .hdf5 and .h5m file in Visit 20
Figure 3.6 Temperature distributions on different elevations 21
Figure 3.7 k_{eff} transient profile as a function of feedback and temporal resolution 22
Figure 3.8 Transient evolution of coupled field profiles at the beginning, during and at the end of the perturbation a) power distribution (W) b) temperature (K) 23
Figure 3.9 sahex_core problem mesh 25
Figure 3.10 Velocity distribution for sahex_core 27
Figure 3.11 Power distributions for heterogeneous and homogenized assembly 28
Figure 3.12 Temperature distributions on difference elevations 29

LIST OF TABLES

Table 2.1. List of SHARP toolkit dependencies 7
Table 3.1 sahex detailed geometry information. 12
Table 3.2 sahex property and flow condition. 12
Table 3.3 sahex input files 13
Table 3.4 Mesh convergence study on k_{eff} 20

1 Introduction of 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 [2] framework. Physics modules currently include the PROTEUS [3] neutronics code, the Nek5000 [4] thermal-hydraulics code, and the Diablo [5] structural mechanics code. 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. 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.

In order to produce a fully coupled-physics simulation capability, two obvious approaches can be pursued. In one approach, existing single-physics codes/components can be assembled into an overall coupled simulation code with appropriate interfaces to communicate between the components to capture the nonlinear feedback effects. This is generally referred to as a "small-f" or "bottom-up" framework approach [1, 6]. The other approach is to use an integrated, coupled-physics modeling framework, with new code pieces for each relevant physics area developed inside that framework from scratch. This is sometimes referred to as a "large-F" or "top-down" approach [7, 8]. The primary advantage of the former approach is that it preserves several man-years invested in existing verified and validated individual physics modeling codes, but at the cost of some intrusive modifications to enable the software interfaces. The large-F approach avoids intrusive interfacing by providing a unified platform to enable coupling, but at the cost of re-writing all the necessary physics codes and verifying the components individually and as a whole. The overall approach being pursued in the RPL effort is to develop and demonstrate a small-f framework for performing coupled multi-physics

analysis of reactor core systems. This system takes advantage of many single-physics codes also sponsored by the overall NEAMS program over past several years.

In the SHARP framework, MOAB interfaces are implemented for 3 different physics components that are relevant to fast reactor physics analysis. The addition of a new physics component to the framework requires integration and ability to read the mesh and possibly associated data from iMesh/MOAB formats, along with implementation to propagate solution variables back onto the mesh after their computation via tags defined either on discrete vertices or elements. Because of the various storage formats used in physics models, and the parallel domain-decomposed environment in which these calculations are usually run, this integration process can be somewhat involved.

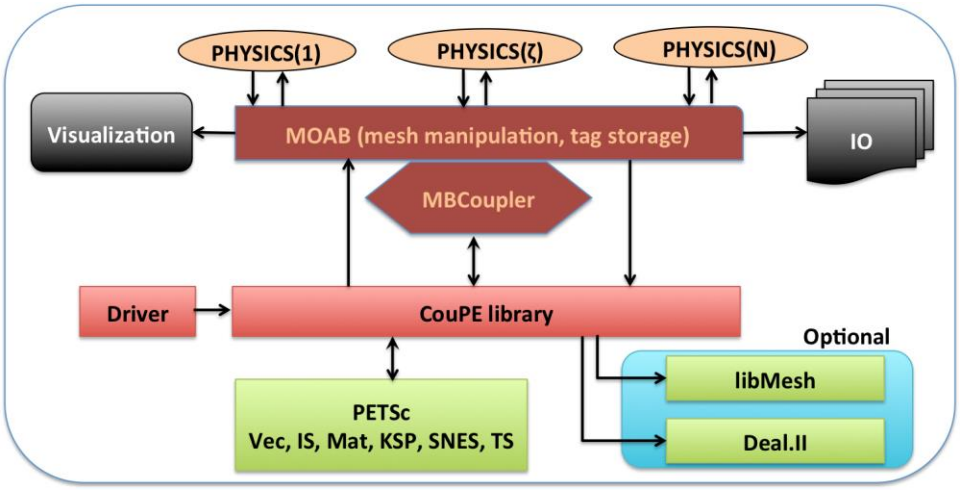
A multi-physics reactor core modeling code can be constructed in many ways, and numerous past efforts have provided stepping-stones for future efforts [8]. What distinguishes the SHARP effort from others is the goal of flexibility in the physics, discretization types, and software options supported by the framework. This section describes the SHARP modeling approach in detail and illustrates how various existing physics codes have been connected to this framework.

As stated above, SHARP employs a "bottom-up" approach, so it can use existing physics codes and take advantage of existing infrastructure capabilities in the MOAB framework and the coupling driver/solver library, the Coupled Physics Environment (CouPE), which utilizes the widely used, scalable PETSc library [8].

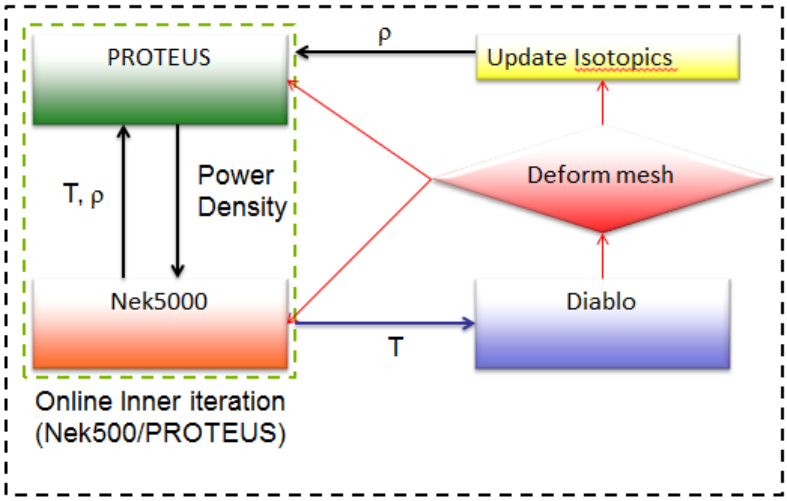
Using an existing physics code in this system (Figure 1.1) requires that the system support the mesh type used by the individual physics models. The physics models can retain their own native representation of the mesh, which gets transferred to and from MOAB's representation through a mesh adaptor; or it can use MOAB's representation directly. Language interoperability through the C/Fortran-based iMesh interfaces also allows flexibility in the implementations that are tuned to individual physics requirements without overhead.

In practice, this means that the coupled system may be solved on multiple meshes, each of which models part or the entire physical domain of the problem. To perform efficient coupled calculations, the results must be transferred from the mesh on which they are generated (source mesh), to the mesh for which they provide initial or boundary conditions (target mesh) due to

nonlinearity introduced because of coupling between physics models. "Multi-way" transfer is required in cases where the physics depend on each other's solution fields, for example in reactor analysis where neutronics computes heat generation based on temperature properties computed by thermal-hydraulics, which in turn depends on the heat-generation source term computed by neutronics.



a



b

Figure 1.1. a) Depiction of the CouPE solvers interacting with MOAB data backplane driving the standalone or coupled physics calculations. b) Sketch of global iteration.

Since relevant physics components solving a nuclear engineering problem have widely varying backgrounds in terms of code architectures, dependency requirements, and specialized solver

data-structures, a flexible approach to the coupling methodology was necessary to obtain accurate solutions. This motivation led to the development of the MOAB-based spatial projection tools and the CouPE drivers based on PETSc library to orchestrate the global nonlinear solver. Details regarding these tools are given in the following sections.

In the future, Diablo, PROTEUS, and Nek5000 will all run simultaneously underneath the CouPE framework and commproteusate quantities through MBCoupler in MOAB. As an interim step, the coupling has been accomplished through file-based transfer. This is a 2-step process, with the second step consisting of 8 substeps.

1. Individual Nek5000, PROTEUS, and Diablo meshes are generated in the undeformed configuration. Nek5000 and PROTEUS use MOAB mesh files natively. Currently Diablo uses an EXODUS input file and writes the equivalent MOAB (".h5m") file as part of the initialization process. Thus, four mesh files are prepared:
    a. NEK.in.h5m
    b. PROTEUS.in.h5m
    c. DIABLO.in.exo
    d. DIABLO.in.h5m (created by Diablo when it initializes)
2. SHARP iterates the problem until convergence:
    a. Coupled 2-mechanics runs (PROTEUS and Nek5000) are made using the updated mesh
    b. Temperature data from Nek5000 is written to its native "FLD" file format, NEK.temps.FLD
    c. The VisIt utility converts the "FLD" file format to a MOAB ("h5m") file, NEK.temps.h5m
    d. A standalone version of MBcoupler maps the Nek5000 data in .h5m format to the Diablo .h5m file, DIABLO.temps.h5m
    e. Diablo uses the temperature data and coupled solid mechanics to produce deformations, which are written as scalar quantities UX, UY, and UZ to an undeformed MOAB database, DIABLO.disp.h5m
    f. The standalone version of MBCoupler maps the UX,UY,UZ data to the Nek5000 and PROTEUS meshes,
        i. NEK.disp.h5m

ii. PROTEUS.disp.h5m

g. The standalone utility DEFORM moves the vertex coordinates on the PROTEUS and Nek5000 meshes according to the mapped values of UX, UY, and UZ,
   i. NEK.deformed.h5m
   ii. PROTEUS.deformed.h5m

h. PROTEUS densities and isotope volume fractions are updated based on the mesh deformation if so desired.

i. The deformed meshes are used as inputs to repeat step 2a above and continue the iterations, as depicted in Figure 1.2

To better understand the level of fidelity that can be achieved by the SHARP framework, some key aspects of the 3 physics components are given in the following chapters.

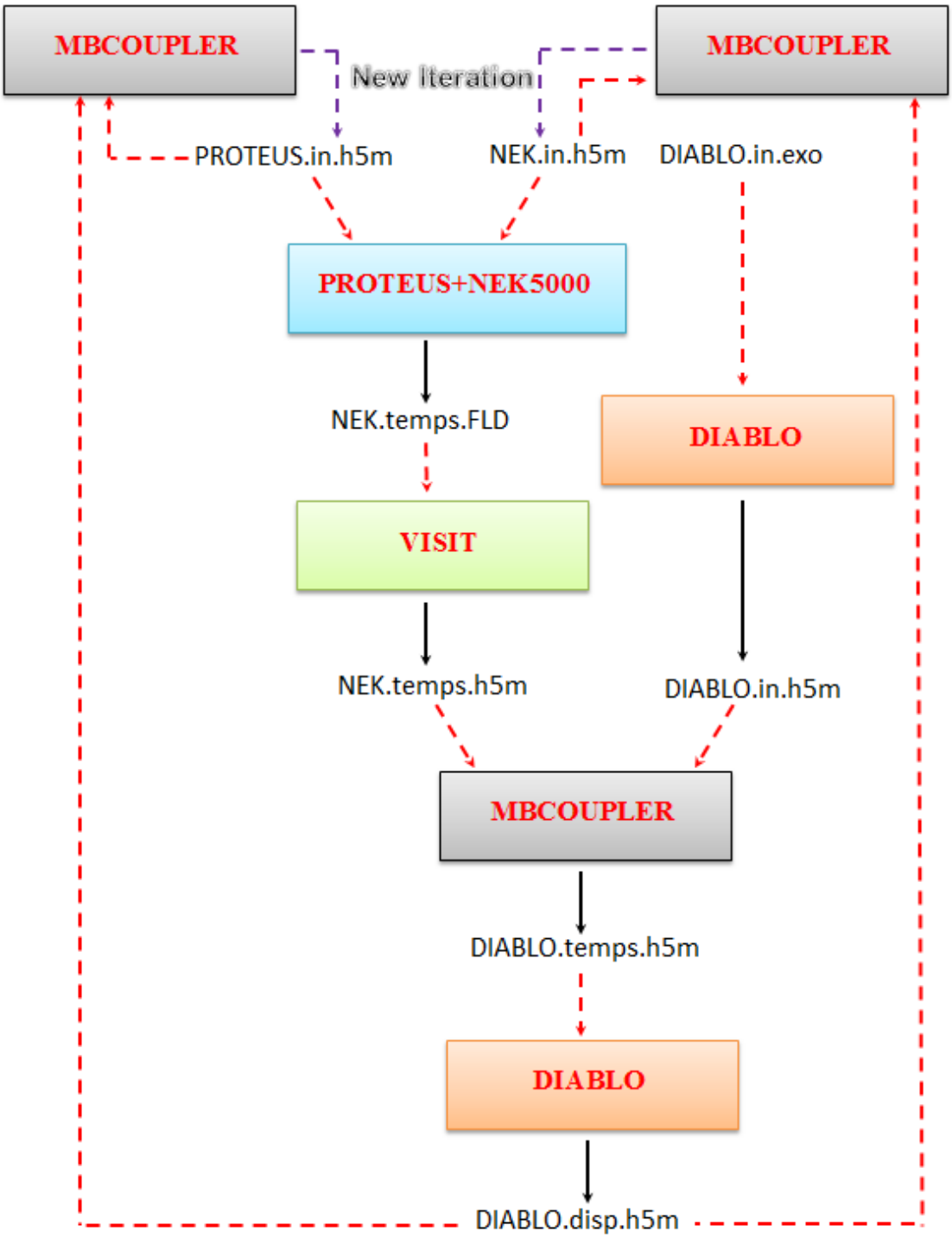


Figure 1.2 Coupling and iteration process.

2 SHARP Configuration and Installation

2.1 Package Dependencies

In Table 2.1, the dependencies of SHARP toolkit and each module are provided. "√" indicates a required package and "(O)" indicates an optional package. A number of packages are optional for single-physics modules but required by the coupled problems in SHARP.

Table 2.1. List of SHARP toolkit dependencies

Package	Version	DIABLO	PROTEUS	Nek5000	SHARP
ARPACK		√			
EXODUS II	6.06	(O)			
FEMSTER		√			
HDF5	1.8	(O)	(O)	(O)	√
HYPRE	2.9	√			
ITAPS		(O)			√
Metis	4.0	(O)	√	(O)	√
MILI	13.1	√			
MOAB	4.7	(O)	(O)	(O)	√
MPI		√	√	√	
MUMPS	4.1	√			
NASA		√			
NetCDF	4.3	(O)	(O)	(O)	√
ParMetis	4	(O)			√
PETSc	3.1 or 3.4		√		
PWSSMP		√			
SILO	4.1	√			
ZOLTAN	3.8	(O)	(O)	(O)	√

2.2 Source Management

The SHARP toolkit utilizes SVN version control system to manage the sources for different components. The repository is hosted at Argonne and is available at the following link, as long as users have been pre-approved to gain access to it.

```
Link: https://svn.mcs.anl.gov/repos/SHARP
```

For instructions on how to obtain access to the SHARP repository, contact sharp-dev@mcs.anl.gov. Once you have been granted access, the user could checkout the sources by invoking the following Subversion command:

```
svn co https://svn.mcs.anl.gov/repos/SHARP/trunk SHARP
```

This command will create the SHARP directory containing all the configuration scripts, physics code module sources and a particular version of the coupled system driver along with several examples and test problems.

2.3 Configuration Options

There are several families of configuration options available when running the configure script, that can be used to control whether certain features are enabled/disabled, or if certain dependencies (pre-installed) need to be used in the current build or if the user wants the package manager to auto-download and configure some of the dependencies.

2.3.1 Enabling/disabling compile-time features or packages

Compile-time features are enabled by options of the form:

```
--enable-<feature>[=yes|no]
--enable-<package>[=yes|no]
```

Values other than yes/no will return an error. --enable-<feature> is synonymous with --enable-<feature>=yes and --disable-<feature> is synonymous with --enable-<feature>=no.

2.3.2 Linking to existing libraries

The user may to link pre-installed libraries with options of the form:

```
--with-<PACKAGE>=PATH
```

where the PATH points to the installation directory typically containing the library and its headers. The PATH argument is mandatory and invalid paths return an error during configuration checks. If a valid library or dependency has been found, then the configuration for the dependency is processed to see if the required headers are available and if a test program that utilizes the library calls can be successfully compiled and linked in order to accumulate the overall LDFLAGS and LIBS to compile SHARP successfully.

2.3.3 Downloading third-party libraries

SHARP allows the user to download and compile third-party libraries through the configure script. Third-party libraries may be downloaded and compiled with options of the form:

```
--download-<PACKAGE>[=yes|no|url]
```

Specifying --download-library[=yes] will download a tarball (.tar.gz file) of the library's source code from a default URL and utilize default, verified workflows to configure, build and

install the dependency onto a dependency installation directory. Running "configure –help" provides a list of the default URLs associated with each library. Optionally, the user can also specify --download--library=URL in order to download the library from the given URL and build/install using the same process. Note that the user-provided URL should point to a valid tarball and may not be guaranteed to be compatible with other SHARP libraries and modules.

2.4 Installation

Step 1:

The user could checkout the sources by invoking the following Subversion command:

```
$svn co https://svn.mcs.anl.gov/repos/SHARP/trunk SHARP
```

This command will create a SHARP directory. Go to SHARP directory:

```
$cd SHARP
```

Step 2: To configure SHARP, the user first needs to run a top-level configuration generator script, aptly named "bootstrap". This script will verify whether the system contains the necessary and supported version of autotools before proceeding further. After verification, the autotools toolchain (aclocal, autoheader, autoconf, automake) can be used to generate the configuration script. An example output from successfully running the bootstrap script is shown below.

```
$./bootstrap
----------------------------------------------------------------
Bootstrap for SHARP build system.
Beginning to run bootstrap in /Users/mahadevan/source/sharp/trunk.
----------------------------------------------------------------
Scanning dependencies...
Checking for autoconf........ [ found version 2.69 ]
Checking for autoheader........ [ found version 2.69 ]
Checking for aclocal........ [ found version 1.14.1 ]
Checking for automake........ [ found version 1.14.1 ]
Checking for libtoolize........ [ found version 2.4.4 ]
Checking for autoreconf........ [ found version 2.69 ]
----------------------------------------------------------------
Found all necessary dependencies. Proceeding with setup...
----------------------------------------------------------------
Running the autotools...
Running autoreconf..... [ done ]
```

```
----------------------------------------------------------------------
Done bootstrapping your system. You may now run ./configure.
To see options use ./bootstrap --help or view the README file.
----------------------------------------------------------------------
```

Step 3: Create new "build" directory and set up the build process with desired set of features. For example, the user may quickly set up and running on Blues with the following commands:

```
$mkdir build
$cd build
$export MPI_DIR=/soft/mvapich2/2.0-intel-13.1
../configure --enable-diablo=yes --with-mpi=$MPI_DIR --download-
essential CC=$MPI_DIR/bin/mpicc CXX=$MPI_DIR/bin/mpicxx
F77=$MPI_DIR/bin/mpif77 FC=$MPI_DIR/bin/mpif90
$make all -j12 #build with 12 processors in parallel
or
$make all #build in serial
```

Step 4: (Optional): Visit is open source and necessary for results visualization. User is able to find the source from the following link:

https://wci.llnl.gov/simulation/computer-codes/visit/source

Copy the script form VisIt build_visit script from the above link.

Run the script with the following command in any directory that user wants to install Visit:

```
$ <script name> --itaps --netcdf --hdf5 --szip --console --silo --
python
```

After about 3 hours downloading and compiling the necessary packages, the user may try to run Visit by cd'ing into the visit2.10.0/src/bin directory and invoking "visit".

3 Demonstration Examples

A number of test problems are provided with SHARP. Users can perform both single physics and coupled physics with different drivers. The frequently-used drivers are ex2, proteusnek, proteusnek_pseudo_ss and proteusnekdiablo. The detailed usage and function of these drivers are described in the following chapter. In this section, two of these tests are described to illustrate the workflow of SHARP toolkit. Reference outputs are provided such that the user can run the various example executables described in the previous section and determines whether the same output is obtained.

3.1 sahex

3.1.1 Problem Description

This model consists of 6 fuel pins with cladding, 1 control rod, sodium coolant and an outer duct wall that encloses all the other components. This model is carefully chosen to demonstrate the proof of concept for solving the xx09 assembly used in EBR II SHRT-17. EBR-II is a heterogeneous, sodium-cooled fast breeder reactor which operated from 1964 to 1994.The geometry of the problem is shown in Figure 3.1. The detailed geometry information and flow condition are listed in Table 3.1 and

Table 3.2 respectively.

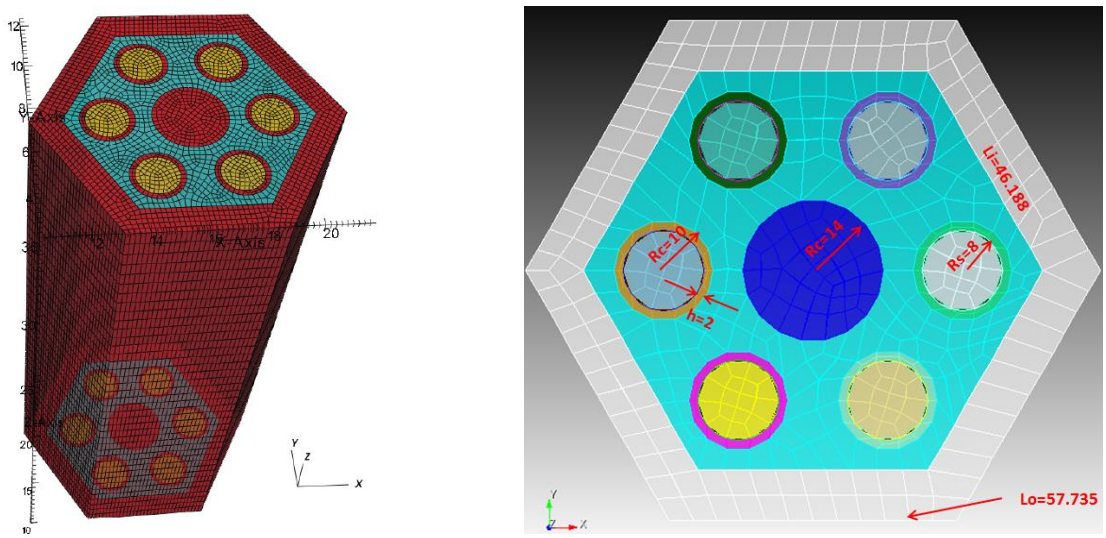


Figure 3.1 sahex problem geometry and mesh

The data transfer mechanism for optimal accuracy and consistency is tested for sahex as the source and target meshes are refined. The convergence criterion of the global nonlinear iteration between the physics is evaluated by measuring the error in the coupled solutions in each physics component. If both the absolute and the relative tolerances are below user-specified values, the iteration is set to have converged.

The problem was tested in a two-stage process:

1) Compute coupled initial steady state solutions at rated conditions.

2) Perform a quasi-static transient where the power, temperature and density evolve based on the change in total reactivity.

Table 3.1 sahex detailed geometry information.

Parameter	Unit (mm)	Parameter	Unit (mm)
Rc	14	*Li*	46.188
Rs	8	*Lo*	57.735
Ra	10	*H*	250
h	2		

Table 3.2 sahex property and flow condition.

Parameter	Value	Remark
ρ	0.85	850 kg/m^3 p1 in *sahex_nek.rea file*
μ	0.26	2.6e-4 pas p2 in *sahex_nek.rea file*
ρC_p	1.08110	C_p=1271.5 J/kgK P7 in *sahex_nek.rea file*
k	0.7	Na @ 700K (W/cm·K) P8 in *sahex_nek.rea file*
Re	11700	-
Δt	1e-3	s
V_{in}	65.3896	cm/s

3.1.2 Input files

The inputs files for sahex problem are listed in Table 3.3. These files include input files for different modules and other control files. The input data for the neutronics solver were generated a prior using the MC2 library (associated with PROTEUS solver) to obtain parameterized 9-group cross sections as a function of temperature and density. Since this is an isolated assembly model, a vacuum boundary (non-reentrant) condition is applied on the top and the bottom and reflective boundary condition on all other surfaces of the neutronics model (Dirichlet for the fluid, Neumann for the solid) and outlet boundary conditions are applied at the top surface.

Table 3.3 sahex input files

PROTEUS	
File Name	***Description***
sahex_proteus.inp	Driver input file, a plain text (ASCII) file, which drives the PROTEUS calculation by specifying solver tolerances, the angular discretization, parallelization options, and other input options. Additionally, the UNIX file paths to the other input files (cross sections, mesh, and material assignment file) are specified in it
sahex_proteus.h5m	h5m format mesh file where the block number and sideset number are specified
sahex_proteus.ISOTXS *sahex_proteus_4g.ISOTXS* *sahex_proteus_9g.ISOTXS*	The cross section file that consists of the multigroup cross sections for all isotopes and/or compositions in the problem. The ISOTXS format is the preferred file format for cross section data used by PROTEUS-SN. The MC2-3 code (Argonne National Laboratory) can be used to process multigroup cross sections in this format. Additionally, the DRAGON code (Ecole Polytechnique de Montreal) [11] has a capability to generate ISOTXS file.
sahex_proteus.anlxs	The cross section file with anlxs file format is a simple ASCII interpretation of the data provided in ISOTXS. Any anlxs file format that is provided is converted into the appropriate ISOTXS file at runtime.
sahex_proteus.assignment	Material assignment file which performs three main functions: (1) define materials or mixtures based on the isotopes in the cross section file, (2) assign these materials to blocks in the mesh, and (3) assign properties (e.g. density) to blocks in the mesh.
NEK5000	
sahex_nek.rea	Input file consists of several sections: ○ Parameters such as viscosity, conductivity, number of steps, time step size, order of the time stepping, frequency of output, iteration tolerances, flow rate, filter strength, etc. ○ passive scalar and logical switches ○ Mesh and boundary condition information ○ Output info like restart conditions
sahex_nek.h5m	h5m format mesh file which can only work under MOAB framework
sahex_nek.usr	The user file where users may specify spatially varying properties (e.g., viscosity), volumetric heating sources, body forces, and so forth. One can also specify arbitrary initial and boundary conditions through the routines useric() and userbc(). The routine userchk() allows the user to interrogate the solution at the end of each time step for diagnostic purposes.

SIZE	The SIZE file that defines the problem size, i.e. spatial points at which the solution is to be evaluated within each element, number of elements per processor etc. The SIZE file governs the memory allocation for most of the arrays in Nek5000, with the exception of those required by the C utilities.
Diablo	
sahex_Diablo.assembly	Input file consists of solver information and material data
sahex_Diablo.subassembly	Input file consists of block ids and boundary conditions
sahex_Diablo.exo	The EXODUS format mesh file and writes the equivalent MOAB (".h5m") file as part of the initialization process
sahex_Diablo.h5m	h5m format mesh file created by Diablo when it initializes
OTHER	
README	Simple description of the sahex problem
Makefile.am	Make file with Driver information
sahex_nek.jou	mesh script for RGG program in MeshKit

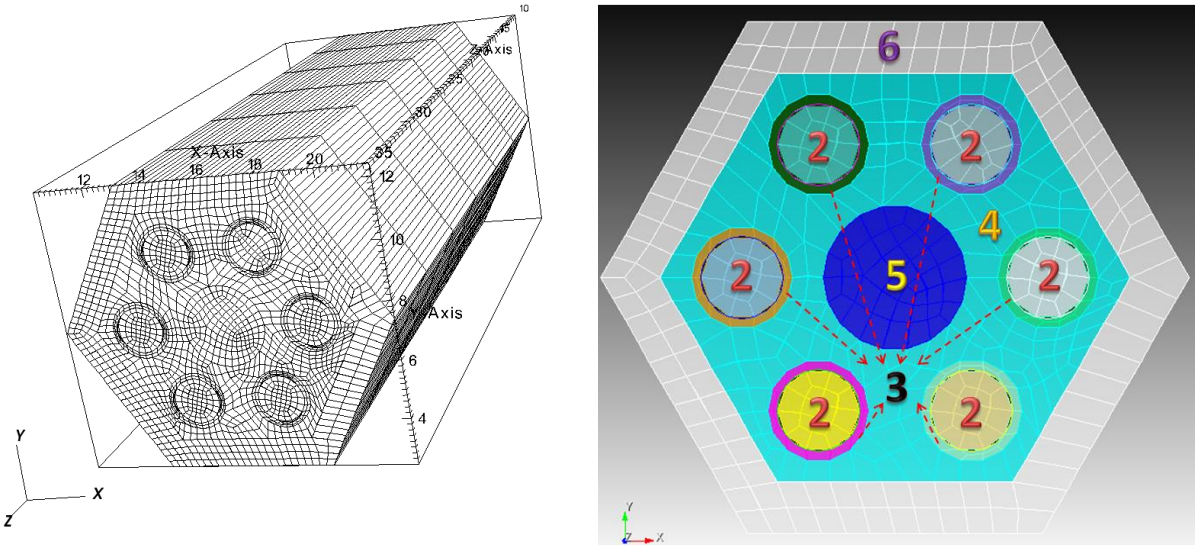


Figure 3.2 NEK5000 mesh and Block number (sahex_nek.h5m)

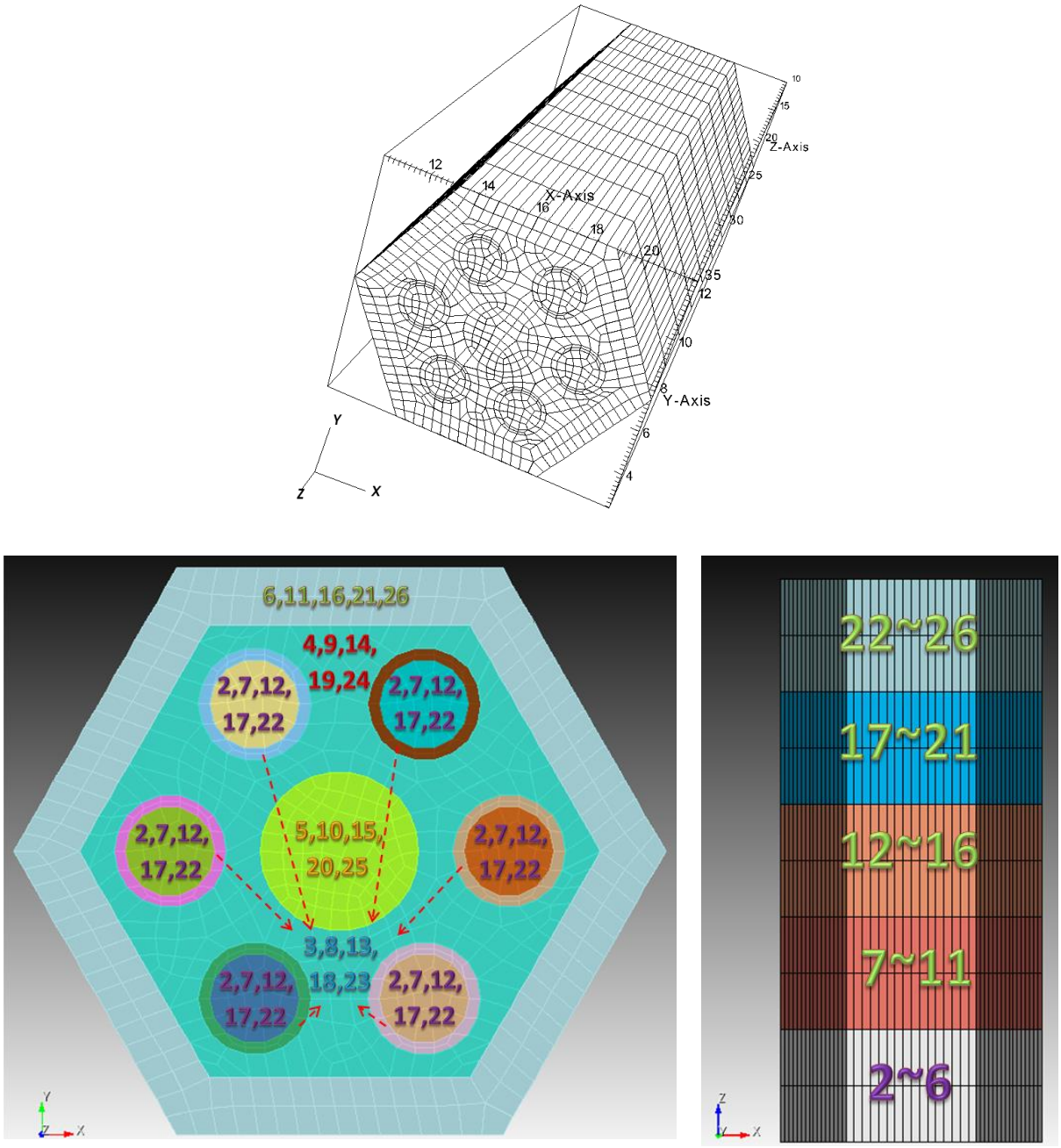


Figure 3.3 Diablo and PROTEUS mesh Block number (sahex_proteus.h5m)

The mesh and block number for Nek5000 and PROTEUS are shown in Figure 3.2 and Figure 3.3. For NEK5000, there are 5 blocks for fuel pins; cladding; sodium coolant; control rod and duct wall. For PROTEUS and DIABLO, there are 25 blocks which are divided by 5 parts in stream wise direction for different axial height materials. In term of the mesh resolution, the

hydraulics solver uses a comparatively coarser mesh than neutronics, consistent with the use of quadratic elements and nature of spectral discretization.

3.1.3 Running

3.1.3.1 Steady State

Step1: Prepare input files

Make sure all the input files are available in the working directory: ~/sharp/build/tests/sahex

Step2: Build driver

```
cd ~/sharp/build/tests/sahex
make all #build all the drivers
make single_physics_sahex # build single physics driver
make single_physics_with_mesh_sahex #build single physics driver to
read a mesh file and create an iMeshInstance
make proteusnek_sahex # build PROTEUS-NEK coupled physics driver
make proteusnek_pseudo_ss_sahex # build PROTEUS-NEK pseudo-transient
coupled physics driver
make proteusnekdiablo_sahex # build PROTEUS-NEK-Diablo coupled
physics driver
```

The above command lines create executable files single_physics_sahex, single_physics_with_mesh_sahex, proteusnek_sahex, proteusnek_pseudo_ss_sahex

Step3: Run the case

Serial:

```
./single_physics_sahex -physics_type proteus -session sahex # Run
PROTEUS single physics
./ single_physics_sahex -physics_type nek -session sahex # Run
NEK5000 single physics
./single_physics_sahex -physics_type proteus sahex_proteus.h5m # Run
PROTEUS single physics with iMeshInstance option
./ single_physics_sahex -physics_type nek sahex_nek.h5m # Run NEK5000
single physics iMeshInstance option
./proteusnek_sahex -session sahex # Run PROTEUS-Nek5000 coupled
calculation
./proteusnek_pseudo_ss_sahex -session sahex # Run PROTEUS-Nek5000
pseudo-transient coupled calculation
```

User can use the following option to control the calculation such as maximum integrations and minimum tolerance.

-coupe_rtol <1e-06>: Stop if decrease in function norm less than (CoupeSetTolerances)

-coupe_max_it <50>: Maximum iterations (CoupeSetTolerances)

-coupe_ploc_tol <5e-06>: Specify the minimum tolerance for the point location from a source to target mesh (CoupeSpatialProjection)

-coupe_type <picard>: Coupe type (one of) picard nk (CoupeSetType)

Parallel:

```
mpiexec -n <executable file>
```

Example on blues:

Create a script called sahex.pbs with the following content:

```
#!/bin/sh
#PBS -N sahex
#PBS -l nodes=4:ppn=16
#PBS -l walltime=10:00:00
#PBS -j oe
cd $PBS_O_WORKDIR
mpiexec
/blues/gpfs/home/yyu/sharp2/build/modules/sharp/src/tests/sahex/prote
usnek_sahex -session sahex | tee proteusnek_sahex.log
# Submit to batch queue with   qsub hello.pbs
# Submit to shared queue with  qsub -q shared hello.pbs (if <4
nodes,<60 min)
```

Use command:

```
qsub sahex.pbs
```

to submit jobs on blues.

Step4: check the output files

The following output files are created after calculation:

PROTEUS: sahex_4g_L1T2.hdf5.xx

NEK5000: sahex_nek.fldxx

MOAB: target_proteus_nek.h5m

3.1.3.2 Pseudo-transient

The type of transient examined in the paper is a simplified loss-of-heat-sink, where the temperature of the fluid at the inlet boundary is specified as a function of time and the evolution in the coupled fields is computed. This simulates an accident scenario when the heat exchangers fail to remove excess heat from the coolant, thereby increasing the inlet temperatures steadily, causing feedback effects from different sources to interact nonlinearly between the physics.

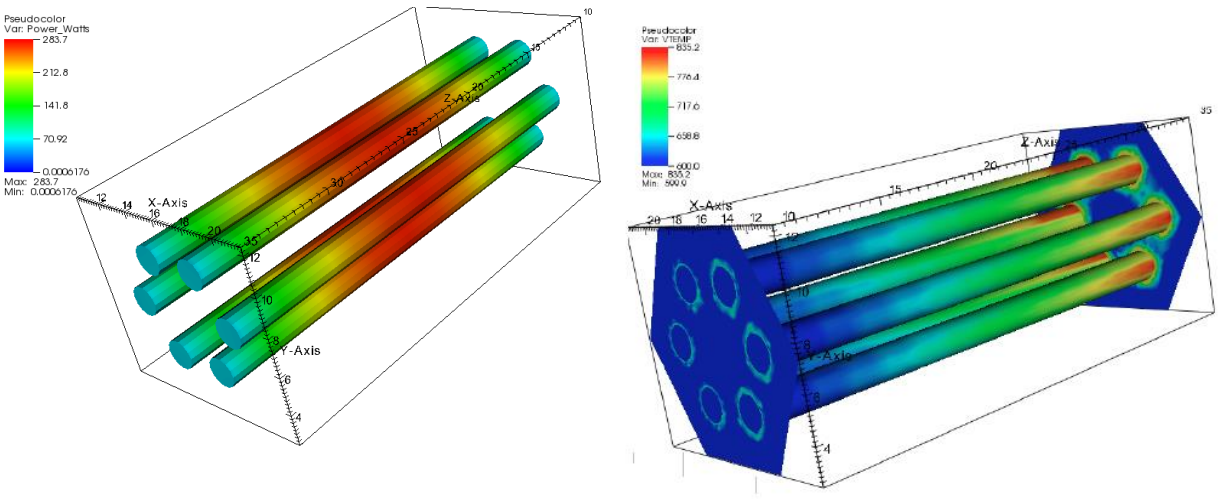
$$T(t) = \begin{cases} T_0, & t \leq t_0 \\ T_0 + \alpha T_0 tahh\left(\frac{t - t_0}{\Delta t}\right) & t \geq t_0 \end{cases}$$

where T_0 is the transient initiation time, T_0 is the initial converged temperature solution, Δt is the duration of the transient at the inlet and α is the damping parameter to control the magnitude of the perturbation (typically 0.2). User can Run the case with the following command:

```
./proteusnek_pseudo_ss_sahex -session sahex # Run PROTEUS-Nek5000
pseudo transient coupled calculation
```

3.1.4 Analyzing

3.1.4.1 Steady State



a) Power b) Temperature

Figure 3.4 Power profile and temperature distribution for steady state solution

The profile of the integral power based on the angular flux computed from solving the Boltzmann neutron transport equation and the temperature profile from thermal-hydraulic solver as shown in Figure 3.4 . The power distribution shifts towards the inlet of the core due to lower material density at the top of the assembly, while the peak temperatures are observable near the outlet since the coolant temperature is monotonically increasing. The convergence criteria for the neutronics and hydraulics solver were specified to 10^{-7} and 10^{-4}.

For Nek5000, use command

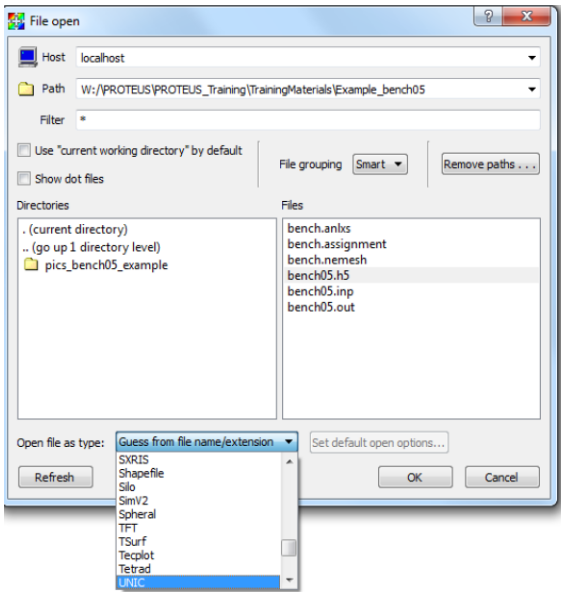
```
Visnek sahex_nek
```

to crease sahex_nek.5000 file which can be loaded into Visit to visualize the results or use vis.nek3d script

Script of vis.nek3d:

```
NEK5000
version: 1.0
filetemplate:  sahex_nek.fld%02d
firsttimestep: 1
numtimesteps: #
Remember to edit numtimesteps: # (the last time step)
```

For PROTEUS, choose UNIC for open file as type as shown in Figure 3.5 to load .hdf5xx file. Alternatively, we can load h5m file to visualize the result by using ITAPS_MOAB type.



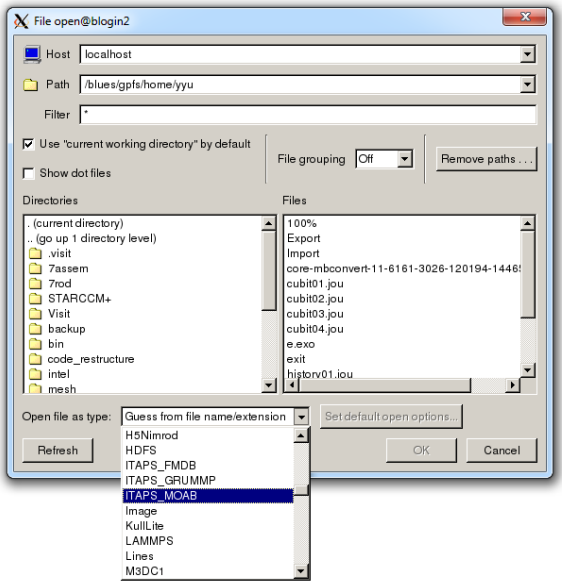


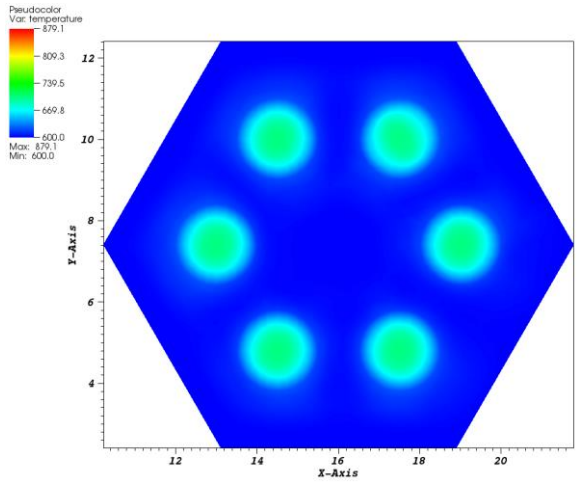
Figure 3.5 Load .hdf5 and .h5m file in Visit

The power distribution generated by adding a pseudocolor plot of the Power variable from sahex_4g_L1T2.hdf5.xx is shown in Figure 3.4 and the temperature distribution on different elevations from sahex_nek.fldxx is shown in Figure 3.6. Several successive refinements of the neutronics and hydraulics meshes are performed. The convergence result of k_{eff} based on a reference mesh solution is listed in Table 3.4. User can find the refined mesh in ./refinement directory. Users should be able to perform more elaborated mesh convergence study by generating their own meshes without doubt.

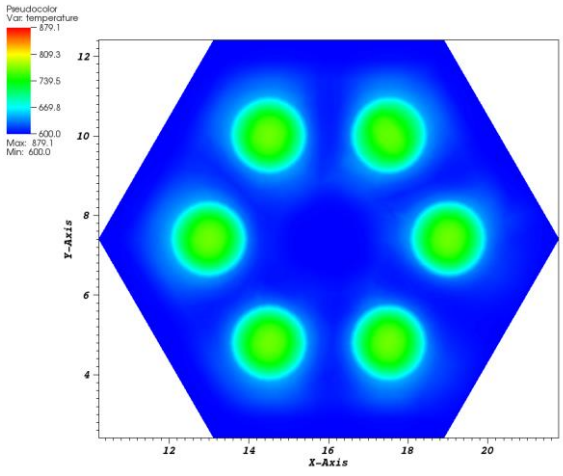
Table 3.4 Mesh convergence study on k_{eff}

Element #	Element size	time	keff	Error(%)
7590	0.7144	44.64	0.67387473	0.42
54740	0.3573	63.39	0.67671801	0.114
437920	0.1791	84.27	0.67751902	0.0315
3503380	0.0895	162.93	0.67775429	reference

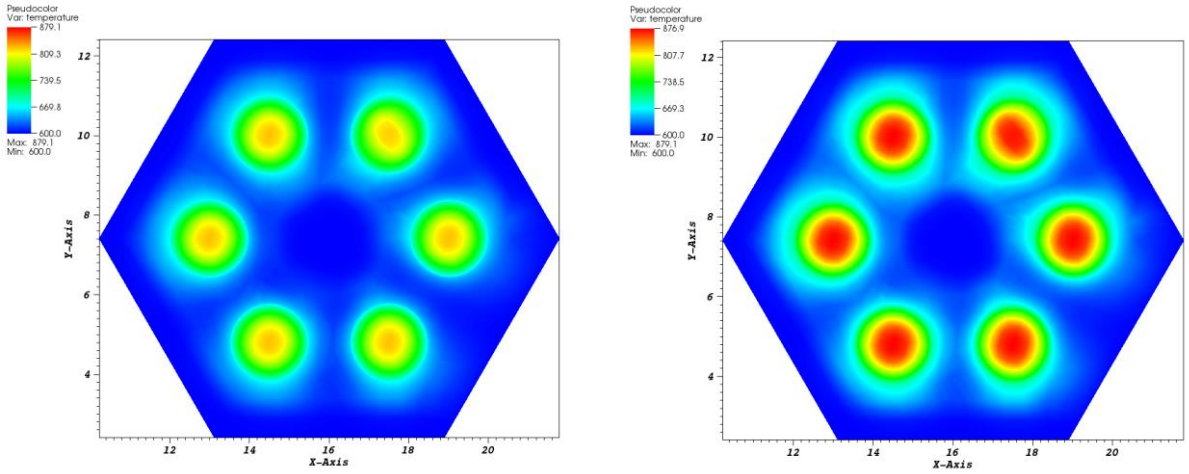
It is also imperative to note that the parallel performance of the solvers (on 32 processors) measured using the computational cost per Picard iteration increases nearly linearly with the number of degrees-of-freedom.



1/4 Span



1/2 Span



3/4 Span 1 Span

Figure 3.6 Temperature distributions on different elevations

3.1.4.2 Pseudo-transient

Once the Initial Conditions are converged, the loss-of-heat-sink simulation is initiated at $t=t_0$, by updating the inlet boundary conditions to increase inlet boundary temperature from 600 to 720K during the transient. The total power in the assembly is specified as user input and power distributions are normalized accordingly.

```
#include "../src/physics/impls/proteus/proteusimpl.h"
PetscErrorCode PerformPerturbation_PROTEUS(Physics proteus)
{
  PetscReal perturbation_factor = 2.0;
  // scale the total power by 2.0; should see
  // increase in temperature and decrease in density
  SN2ND_ScaleTotalPower(&perturbation_factor);
  PetscFunctionReturn(0);
}

#include "../src/physics/impls/nek/nekimpl.h"
PetscErrorCode PerformPerturbation_NEK(Physics nek)
{
  PetscReal perturbation_factor = 1.2;
  // scale the input boundary condition power by 2.0; should see
  // increase in temperature and decrease in density
  NEK_Perturb_BCVars(&perturbation_factor);
  PetscFunctionReturn(0);
}
```

In all the cases, the number of sub-cycling steps performed in thermal-hydraulics was specified to resolve the transient change in temperature. Several transients have been performed to test for sensitivity of the coupled field solutions to different feedback effects. Figure 3.7 shows the change in k_{eff} as a function of normalized time for different types of coupled feedback effects optionally turned on. As the frequency of coupling is increased, the accuracy of the coupled physics solution improves since the computed criticality converges towards the reference. The flow time of the sodium through the assembly is 0.9 s (characteristic time scale), and the overall time-steps are reduced consistently to resolve this spatial and temporal scales in successive simulations starting with Δt_{coarse}=0.02s. Note that feedback based on both temperature and density are necessary to show the complex nonlinear coupling between the neutronics and thermal-hydraulics physics for this test problem since the case where only Doppler feedback is considered shows larger sensitivity to the inlet temperature change. In other words, the density and Doppler expansion feedback are competing effects as validated from theory and experimental observations. The total power decrease in the assembly as the transient progresses can be observed in Figure 3.8a. The corresponding evolution of the temperature profiles is shown in Figure 3.8b. Note that the significant change in power profile corresponding to only a minor penetration of the high temperature front within the domain indicates a very fast response (high sensitivity) to the boundary condition in the system.

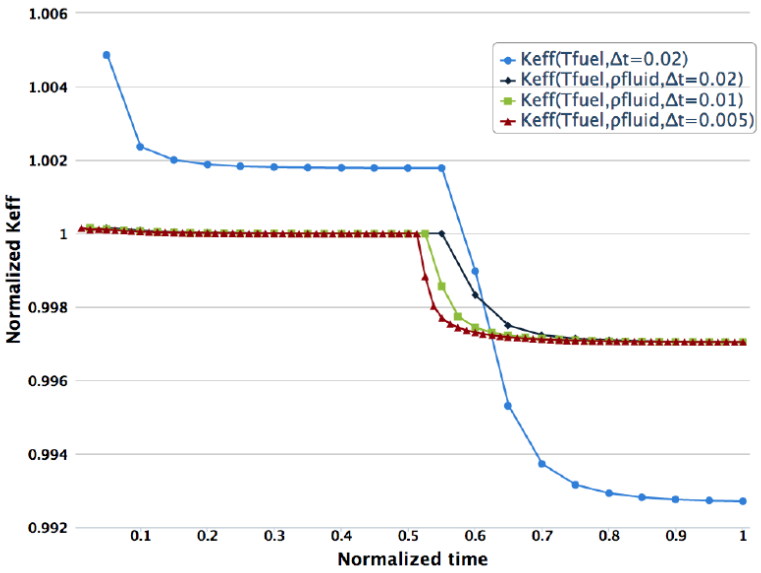


Figure 3.7 k_{eff} transient profile as a function of feedback and temporal resolution

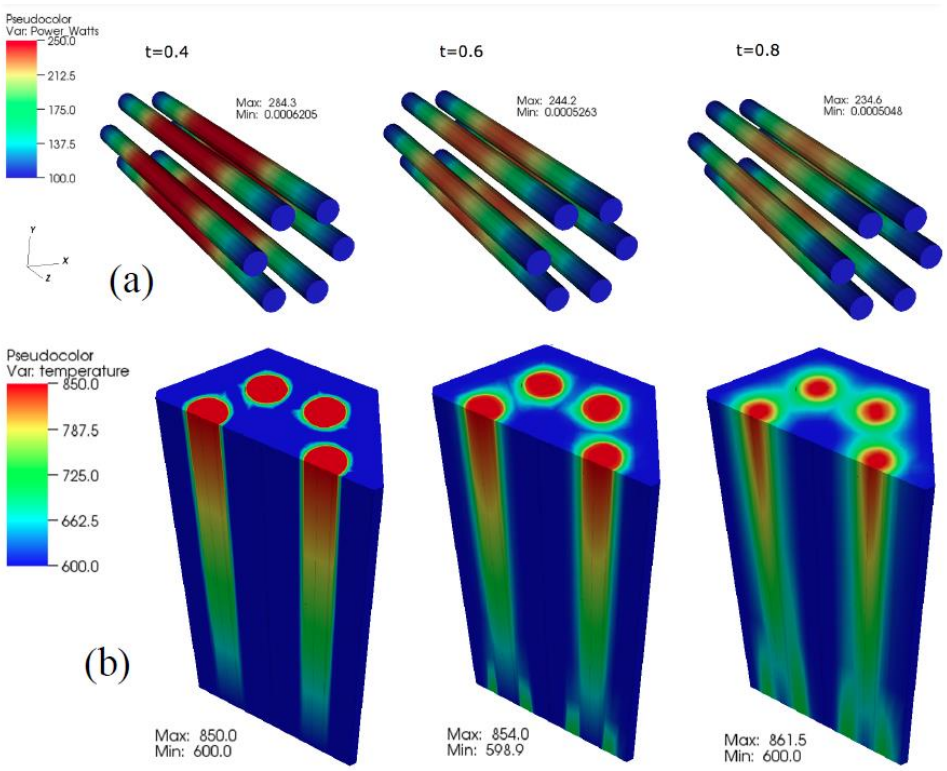


Figure 3.8 Transient evolution of coupled field profiles at the beginning, during and at the end of the perturbation a) power distribution (W) b) temperature (K)

The coupled physics simulation capability with SHARP framework was tested on the sahex problem for a loss-of-heat-sink transient and the results obtained have been verified by spatial and temporal solution convergence studies. The sensitivity tests lead to important conclusions on:

i) The time-step size necessary for this transient to maintain accuracy.

ii) The importance of the inclusion of all types of feedback effects.

3.2 sahex_core

3.2.1 Problem Description

In order to perform a full core simulation with PROTEUS and Nek5000, some form of homogenization is necessary as solving the full core heterogeneously is in fact likely to result in excessive computational requirements.

An important step toward a full-core simulation is therefore to test homogenization capabilities in both Nek5000 and PROTEUS. The chosen test case is an extension of the sahex model described in this section. It adds six homogenized neighboring assemblies to the full heterogeneous sahex model. A multi-resolution, homogeneous-heterogenous problem that contains explicitly represented pin cell assembly in the middle, surrounded by homogenized assemblies with a duct surrounding it.

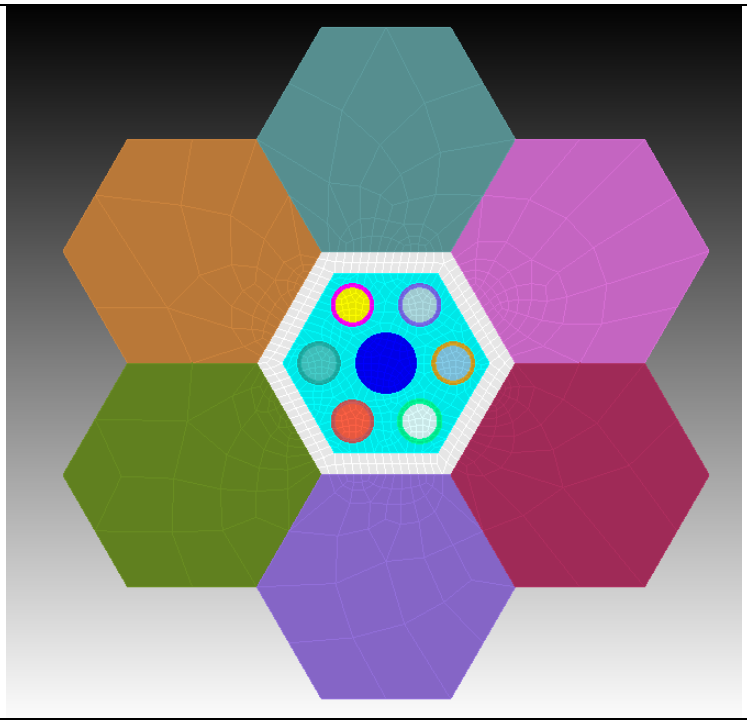
The PROTEUS and Nek5000 mesh for this model is shown in Figure 3.9. In the PROTEUS mode, l reflective boundary conditions are assumed at the boundaries of the domain. In the Nek5000 model the boundary is assumed adiabatic.

In the homogenized assemblies material properties and cross sections are homogenized in PROTEUS. In Nek5000, in each homogenized assembly two one-dimensional equations are solved for the temperature T in time t and axial coordinate x:

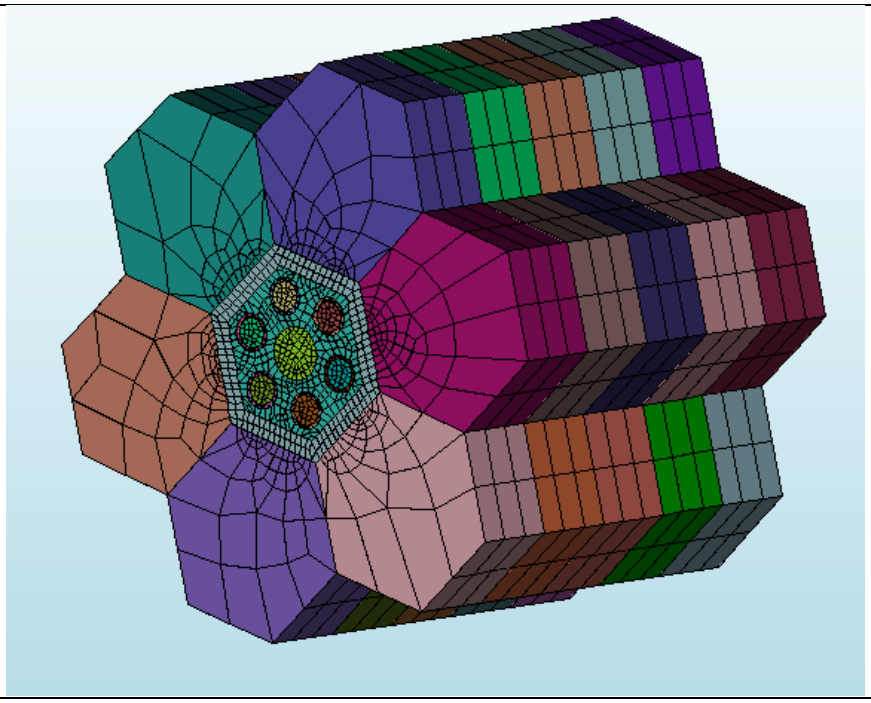
$$\rho^f {C_p}^f \frac{\partial}{\partial t} T^f = -hl^f (T^f - T^s) + \ddot{q}(\mathrm{x})$$

$$\rho^s {C_p}^s \frac{\partial}{\partial t} T^s = -hl^s (T^f - T^s) - v^s \rho^s {C_p}^s \frac{\partial}{\partial x} T^s$$

where the superscript s refers to sodium and the superscript f refers to fuel. This very simple model removes the thermal inertia of the ducts and the cladding; these can be added later. The variable h is the heat transfer coefficient to be determined by appropriate correlations. The variable vs is the bulk velocity of sodium in the assembly Here q is the volumetric heat, and is obtained from integration of PROTEUS results. The variables l^f and l^s are appropriate heat transfer length-scales. Both temperatures T_s and T_f are stored, but only the temperature of sodium is passed to PROTEUS which then recomputes the properties of the homogenized regions.



sahex_core_nek.h5m



sahex_core_proteus.h5m

Figure 3.9 sahex_core problem mesh

3.2.2 Input files

PROTEUS:

sahex_core_4g.ISOTXS sahex_core_9g.ISOTXS: cross sections file of different group numbers

sahex_core _proteus.inp: driver input file

sahex_core_proteus.h5m: mesh file

sahex_core_proteus.assignment: material assignment file

NEK5000:

sahex_core _nek.rea: driver input file

sahex_core _nek.usr: user file

sahex_core_nek.h5m: mesh file

SIZE: size file

3.2.3 Running

Step1: Prepare input files

Make sure all the input files are available in the working directory: ~/sharp/build/tests/sahex_core

Step2: Build driver

```
cd ~/sharp/build/tests/sahex_core
make all #build all the drivers
```

The above command lines create the following executable files

single_physics_sahex_core,

single_physics_with_mesh_sahex_core,

proteusnek_sahex_core

proteusnek_pseudo_sahex_core

Step3: Run the case

Serial:

```
./proteusnek_sahex_core -session sahex_core # Run PROTEUS-Nek5000
coupled calculation
```

Step4: check the output files

The following output files are created after calculation:

PROTEUS: sahex_core_4g_L1T2.hdf5.xx

NEK5000: sahex_core_nek.fldxx

MOAB: target_proteus_nek.h5m

3.2.4 Analyzing

The velocity distribution shown in Figure 3.10 can be visualized with pseudocolor plot in Visit. The precise velocity distribution in the center sahex assembly can be predicted with CFD calculation. Higher velocity can be clearly observed in the gap regions between walls and rods. In the neighbor homogenized assemblies, the velocity distribution which is not our most concern in coupled calculation is uniform since the porous medium model is adopted.

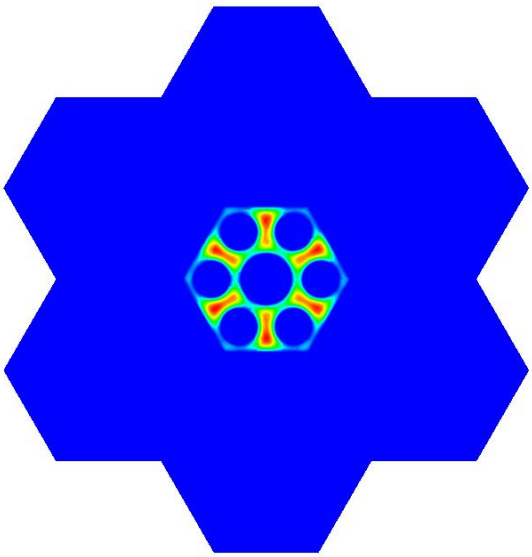


Figure 3.10 Velocity distribution for sahex_core

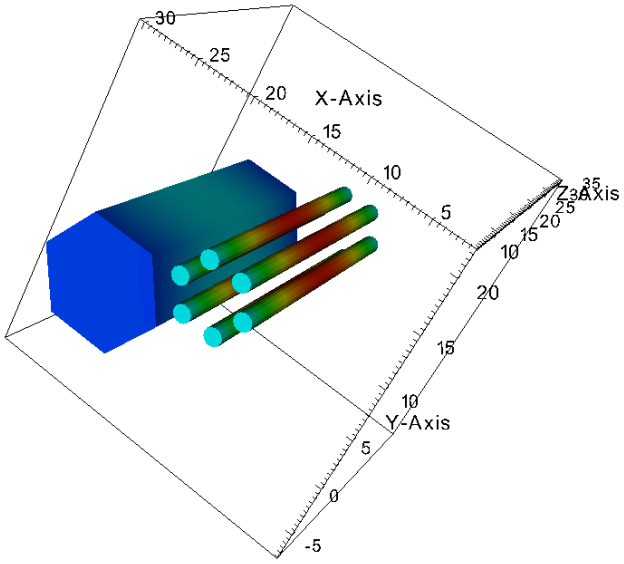
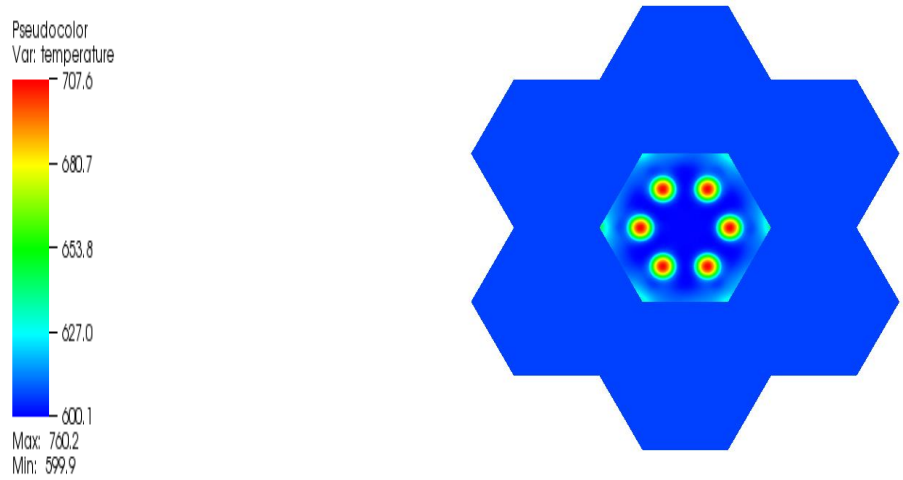


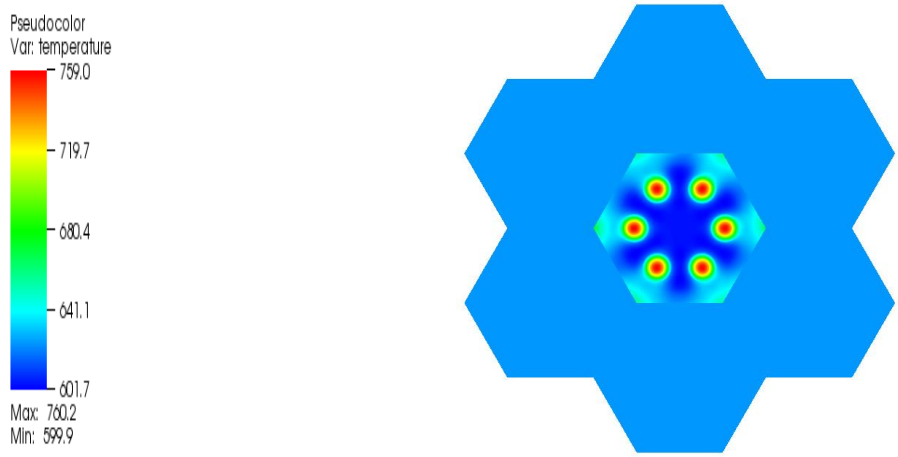
Figure 3.11 Power distributions for heterogeneous and homogenized assembly

The profile of the integral power in both center sahex assembly and neighbor homogenized assembly are shown in Figure 3.11.The cosine shape power distribution can also be well predicted in the homogenized neighboring assemblies as well which supports the feasibility of using some form of homogenization in the coupled calculation for full core simulation.

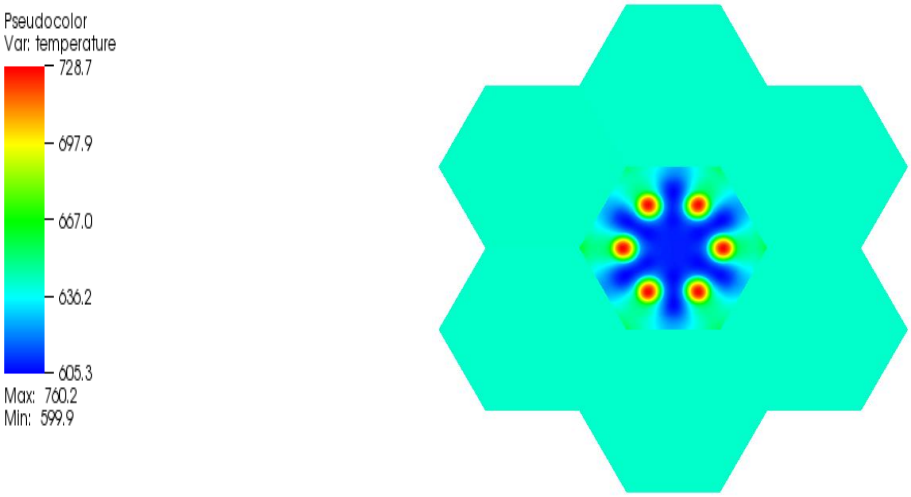
The temperature distributions on different elevation are shown in Figure 3.12 with distinguishing temperature in the center assembly and uniform temperature in the neighbor six assemblies. Although the temperature near the duct wall which is our most concern for radial core expansion between the center assembly and neighbor assembly looks similar, our recent research [10] indicates that this simple one region porous medium model can over predict 80K of the averaged duct wall temperature than the relatively precise CFD results, not to mention the local temperature discrepancy. After all, the model of homogenization needs to be improved to better serve the full core multi-physics simulation. Nevertheless, this test case is still useful for both testing some form of homogenization to reduce the computational cost and helping users to understand the homogenized model in SHARP.



a) 20% of the total elevation



b) 50% of the total elevation



c) 80% of the total elevation

Figure 3.12 Temperature distributions on difference elevations

4 Create Own Test Case

This section describes the procedure of how to create users' own test cases. To be concise, the input files of sahex is renamed as sa and assumed to be the input files of a new case. Certainly, users can used their own input files to create their own case by following this procedure.

Step 1:

Create a new test case called sa in test directory and prepare the input files.

```
$cd ~/SHARP/tests
$cp -r sahex sa
```

Change name of all the input files from sahex to sa. For instance, sahex.rea to sa_nek.rea; sahex_proteus.inp to sa_proteus.inp. Remember to change content in sa_proteus.inp to make sure the case can be run properly. For example,

```
SOURCEFILE_MESH          sa_proteus.h5m
SOURCEFILE_XS            sa_proteus_4g.ISOTXS
SOURCEFILE_MATERIAL      sa_proteus.assignment
EXPORT_FILE              sa_4g_L1T2.hdf5
```

Step 2:

Modify the makefile.am and configure.ac and rebuild SHARP.

Modify SHARP/tests/Makefile.am by including sa in new SUBDIRS:

```
SUBDIRS = dbgprb sahex sahex_core xx09 xx09_core sa
```

Modify SHARP/tests/sa/Makefile.am by changing PROBLEM variable and driver name:

```
PROBLEM = sa
DRIVERS = ex2_sa ex3_sa

# Drivers
ex2_sa_SOURCES = ../ex2.cxx
ex3_sa_SOURCES = ../ex3.cxx

if SHARP_ENABLE_PROTEUS
proteusnek_sa_SOURCES = ../proteusnek.cxx
proteusnek_pseudo_ss_sa_SOURCES = ../proteusnek_pseudo_ss.cxx
DRIVERS         += proteusnek_sa proteusnek_pseudo_ss_sa

if SHARP_ENABLE_DIABLO
DRIVERS += proteusnekdiablo_sa
```

```
proteusnekdiablo_sa_SOURCES = ../proteusnekdiablo.cxx
endif
```

Modify SHARP/configure.ac to add test directory

```
# Create test directories
SHARP_PREP_TESTDIR([sa])
```

Step 3:

Rebuild SHARP

```
$cd ~/SHARP/
$./bootstrap
$cd build
$make
```

Make the drivers

```
$cd ~/SHARP/build/tests/sa
$make all #create all drivers
$make ex2_sa # create specific driver
```

5 Reference

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. T.J. Tautges, R. Meyers, K. Merkley, C. Stimpson, C. Ernst, *MOAB: A Mesh-Oriented Database*, Sandia National Laboratories report SAND2004-1592, April 2004.
3. M.A. Smith, D. Kaushik, A. Wollaber, W.S. Yang, B. Smith, C. Rabiti, G. Palmiotti, "Recent Research Progress on PROTEUS at Argonne National Laboratory," in *Proceedings of the International Conference on Mathematics, Computational Methods and Reactor Physics (M&C)*, American Nuclear Society, April 2009.
4. P.F. Fischer, J.W. Lottes, S.G. Kerkemier, Nek5000 Web Page, http://nek5000.mcs.anl.gov, 2008.
5. 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.
6. T.J. Tautges, H.-J. Kim, A. Caceres, R. Jain, "Coupled Multi-Physics simulation frameworks for reactor simulation: A Bottom-Up approach," in *Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C)*, American Nuclear Society, Rio de Janeiro, Brazil, May 2011.
7. D. Gaston, C. Newman, G. Hansen, D. Lebrun-Grandi, "MOOSE: a parallel computational framework for coupled systems of nonlinear equations," *Nuclear Engineering and Design*, **239**(10):1768–1778, Oct. 2009.
8. D.E. Keyes et al., "Multiphysics Simulations: Challenges and Opportunities," *International Journal of High Performance Computing Applications*, **27**(1):4-83, 2012.
9. T. Tautges, P. Fischer, I. Grindeanu, R. Jain, V. Mahadevan, A. Obabko, M.A. Smith, E. Merzari, "SHARP Assembly-Scale Multiphysics Demonstration Simulations," Technical Milestone Report ANL/NE-13/9, Argonne National Laboratory, 2013.
10. Yiqi Yu, Elia Merzari, Aleksandr Obabko, Justin Thomas, A porous medium model for predicting the duct wall temperature of sodium fast reactor fuel assembly, Nuclear Engineering and Design, Volume 295, 15 December 2015, Pages 48-58,
11. G. Marleau, R. Roy, and A. Hébert, "DRAGON: A Collision Probability Transport Code for Cell and Supercell Calculations," Report IGE-157, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec, 1994.



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