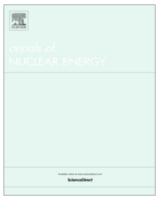
Annals of Nuclear Energy xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Annals of Nuclear Energy

journal homepage: www.elsevier.com/locate/anucene



An approach for coupled-code multiphysics core simulations from a common input

Rodney Schmidt[a,*], Kenneth Belcourt[a], Russell Hooper[a], Roger Pawlowski[a], Kevin Clarno[b], Srdjan Simunovic[b], Stuart Slattery[b], John Turner[b], Scott Palmtag[c]

[a] Sandia National Laboratories, Albuquerque, NM 87185, United States
[b] Oak Ridge National Laboratories, Oak Ridge, TN, United States
[c] Core Physics, Inc., Wilmington, NC, United States

ARTICLE INFO

Article history:
Received 30 April 2014
Accepted 2 November 2014
Available online xxxx

Keywords:
Couple-code
Multiphysics
Reactor core simulations

ABSTRACT

This paper describes an approach for coupled-code multiphysics reactor core simulations that is being developed by the Virtual Environment for Reactor Applications (VERA) project in the Consortium for Advanced Simulation of Light-Water Reactors (CASL). In this approach a user creates a single problem description, called the "VERAIn" common input file, to define and setup the desired coupled-code reactor core simulation. A preprocessing step accepts the VERAIn file and generates a set of fully consistent input files for the different physics codes being coupled. The problem is then solved using a single-executable coupled-code simulation tool applicable to the problem, which is built using VERA infrastructure software tools and the set of physics codes required for the problem of interest.

The approach is demonstrated by performing an eigenvalue and power distribution calculation of a typical three-dimensional 17×17 assembly with thermal–hydraulic and fuel temperature feedback. All neutronics aspects of the problem (cross-section calculation, neutron transport, power release) are solved using the Insilico code suite and are fully coupled to a thermal–hydraulic analysis calculated by the Cobra-TF (CTF) code. The single-executable coupled-code (Insilico-CTF) simulation tool is created using several VERA tools, including LIME (Lightweight Integrating Multiphysics Environment for coupling codes), DTK (Data Transfer Kit), Trilinos, and TriBITS. Parallel calculations are performed on the Titan supercomputer at Oak Ridge National Laboratory using 1156 cores, and a synopsis of the solution results and code performance is presented. Ongoing development of this approach is also briefly described.

1. Introduction

VERA, a "Virtual Environment for Reactor Applications", is a collection of software being developed under DOE sponsorship by CASL, the Consortium for Advanced Simulation of Light Water Reactors (U.S. DOE, 2011). A central goal of VERA development is to "*predict, with confidence, the performance of nuclear reactors through comprehensive, science-based modeling and simulation technology that is deployed and applied broadly throughout the nuclear energy industry to enhance safety, reliability, and economics.*"

As part of the CASL project an approach has been developed to enable coupled-code multiphysics simulations of various reactor core problems of interest. This approach is embodied within an analysis capability called VERA-CS that utilizes a subset of VERA components. The objective is for a user to have the ability to define, set-up, and solve a variety of different challenging reactor-core multiphysics problems by following a simple workflow:

Abbreviations: ASCII, American Standard Code for Information Interchange; BWR, boiling water reactor; CASL, Consortium for Advanced Simulation of Light Water Reactors; CFD, computational fluid dynamics; CRUD, corrosion related unidentified deposit, a term used to describe radioactive scaling that builds up on internal reactor components; CTF, COBRA-TF, coolant-boiling in rod arrays – two fluids; GUI, graphical user interface; JFNK, Jacobian-free Newton–Krylov nonlinear solver; ORNL, Oak Ridge National Laboratory; PWR, pressurized water reactor; T/H, thermal–hydraulic; TPLs, third-party libraries, e.g. BLAS; TriBITS, tribal build, integrate, and test system; VERAIn, VERA common input file; VERA, Virtual Environment for Reactor Applications; VERA-CS, Virtual Environment for Reactor Applications – Core Simulator; VUQ, verification and uncertainty quantification; XML, extensible markup language; XSLT, extensible style sheet language transformation.

* Corresponding author. Tel.: +1 801 733 8568.
E-mail addresses: rcschmi@sandia.gov (R. Schmidt), kbelco@sandia.gov (K. Belcourt), rhoope@sandia.gov (R. Hooper), rppawlo@sandia.gov (R. Pawlowski), clarnokt@ornl.gov (K. Clarno), simunovics@ornl.gov (S. Simunovic), slatterysr@ornl.gov (S. Slattery), turnerja@ornl.gov (J. Turner), scott.palmtag@corephysics.com (S. Palmtag).

http://dx.doi.org/10.1016/j.anucene.2014.11.015
0306-4549/© 2014 Elsevier Ltd. All rights reserved.

1. Choose from a collection of single or coupled-code physics codes as appropriate for the problem of interest,
2. Create a single common input file description for the problem of interest, and
3. Perform a complete simulation, start to finish, with essentially no manual intervention on serial or parallel computing platforms appropriate for the problem size.

There are several important elements that are required to accomplish the objective. They include:

- The availability of a suite of standalone or coupled physics codes that can efficiently solve the various problems of interest.
- An approach and associated software tools to couple different physics codes together (as needed) to solve the multiphysics problems of interest.
- The definition of a common input file that can accommodate the information needed by all participating codes.
- Input adapters to process the common input file and generate all code-specific input files that will be required by the participating physics codes.
- The ability to easily submit a problem for execution on a desired computational platform.

In subsequent sections of this paper each element in this approach is described in greater detail.

Current capabilities are demonstrated by performing an eigenvalue and power distribution calculation of a representative three-dimensional 17 × 17 PWR assembly with thermal–hydraulic and fuel temperature feedback.

2. Elements of the approach

2.1. VERA physics codes

As a whole, VERA consists of a large collection of different computer codes, utilities and software components that address a range of computational needs for reactor modeling and simulation. Conceptually indicated in Fig. 1, VERA can be divided into two broad categories:

- Infrastructure Components and Third Party Libraries (TPLs): The computational infrastructure (e.g. code-coupling and VUQ related software, etc.) and software development environment including compilers and TPLs.

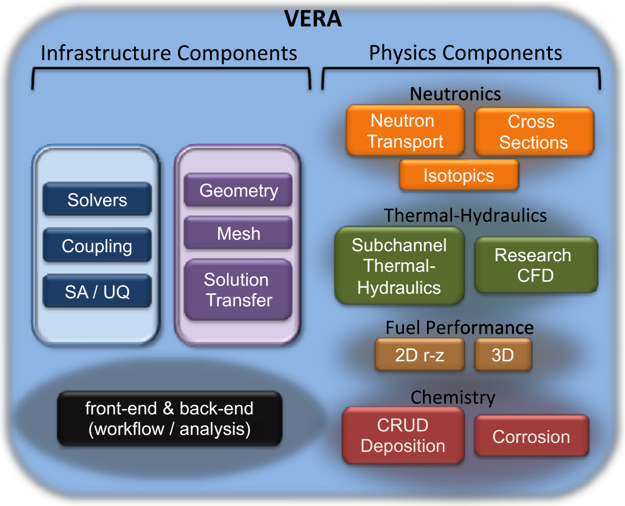


Fig. 1. VERA consists of a collection of application codes and software components.

- Physics Components: These include standalone, integrated and coupled-code multiphysics applications representing component models which together model nuclear reactor performance.

Physics-related VERA application codes can themselves be grouped into several modeling sub-categories; neutronics, thermal–hydraulics, fuel performance, and chemistry:

- Neutronics includes neutron transport modeling (of various degrees of fidelity and cost), cross section calculations, isotopic decay and depletion calculations, fission-based energy release and so forth. Neutronics-related physics codes in VERA include Insilico (see Mervin, 2013), Scale (Bowman, 2011; Oak Ridge National Laboratory, 2011) and MPACT (Kochunas et al., 2013).
- Thermal-hydraulics spans the topics of single and two-phase turbulent flow modeling, conduction, convective and boiling heat transfer, pressure drop modeling, high-fidelity CFD codes as well as engineering-type subchannel codes. Thermal–hydraulics related codes in VERA include the Cobra-TF (CTF) (Avramova, 2009; Salko and Avramova, 2012; Salko et al., this issue) subchannel code and the Hydra-TH (Nourgaliev et al., 2013) CFD code.
- Fuel Performance concerns the thermal–mechanical response of reactor fuel rod materials as well as the time-dependent changes in material properties and morphology that occur to the fuel during the operation of a reactor. The Peregrine fuel performance code (a variation of Bison (Hales et al., 2013), is currently being developed and incorporated into VERA.
- Chemistry issues of particular importance to VERA include the chemical mechanisms causing corrosion and CRUD buildup that contribute to clad failures and/or affect power shape changes. The MAMBA (Kendrick and Barber, 2012) code is currently being developed and incorporated into VERA to address these chemistry issues.

2.2. Multiphysics coupling

To computationally simulate a reactor problem, relevant physical processes are described by models and equation sets that define how the important state variables (e.g. temperature, density, velocity, neutron flux, etc.) behave over the spatial and temporal domain of relevance. When these physical processes interact with one another we say they are “coupled”, and the equations must be solved so that changes in any state variable are properly reflected in all equation sets that are affected by that variable. If the effect of changes in variable “X” on the evolution of variable “Y” is small, then the coupling is said to be weak. Conversely, if the effect is large, then the coupling is said to be strong. Another important aspect of coupling is whether the effect is a linear effect, or a nonlinear effect. The most challenging problems to model accurately are those where the coupling is both strong and nonlinear. Another potential challenge is that the degree of weak versus strong coupling can vary over time in a transient simulation.

Note that when describing multiphysics simulations, we use the word “coupled” in two related but different contexts; one with respect to physics, and one with respect to computer codes. One computer code might be written to solve only one set of physics, but another might be designed to internally solve several coupled physical processes (i.e. a single physics code is not restricted to a single physics). The reactor core problems being targeted for solution with VERA-CS involve a range of physics including thermal–hydraulics, fuel performance, and various aspects of neutronics. For some problems a single physics code is capable of modeling the applicable phenomenon. However, for other problems of interest the important phenomena will span a range of physics

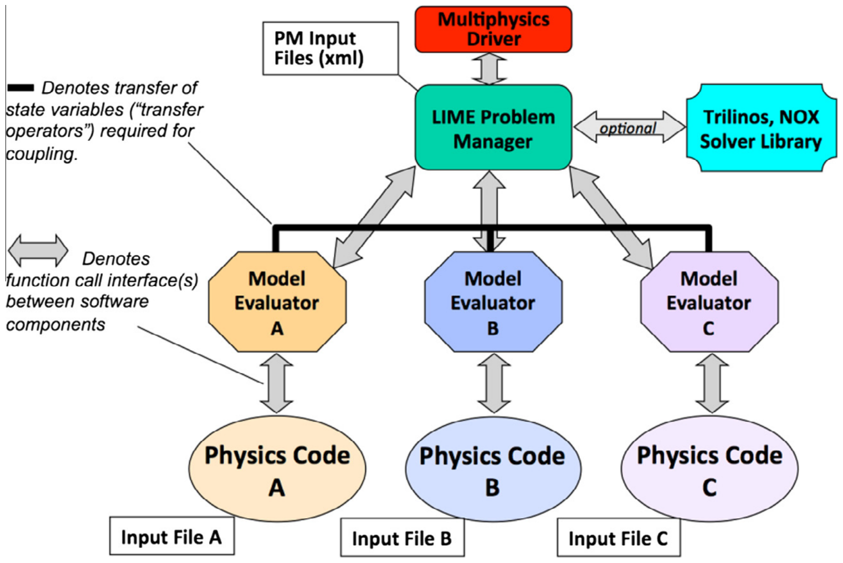


Fig. 2. Key components of a generic LIME-based coupled-code application created from three physics codes.

not adequately modeled by a single physics code. VERA-CS has been designed to enable multiple physics codes to be coupled together so as to accurately simulate all aspects of a multiphysics system. In this setting a fully consistent solution to the overall problem is sought while leveraging the ability of each physics code to solve its part of the overall coupled multiphysics system. CASL adopted this approach early to allow codes of varying fidelity and computational cost to be considered and interchanged as opposed to hard-wiring specific choices by writing a single monolithic code.

Currently, two software packages, LIME 1.0 (Schmidt et al., 2011; Pawlowski et al., 2011) and DTK Slattery et al., 2013) are used to combine the various physics codes into a single coupled-code multiphysics application.

LIME is a small software package that is intended to be especially useful when separate computer codes already exist to solve different parts of a multiphysics problem. LIME provides the key high-level software (written in C++), a well-defined approach, and interface requirements to enable the assembly of multiple physics codes into a single-executable coupled-code multiphysics simulation capability. In order to create a new multiphysics application using LIME, physics codes are required to support a minimal software interface. Depending on the structure and design of the original physics codes, some degree of code refactoring may be required.

Fig. 2 illustrates key components of a generic LIME-based coupled-code multiphysics application created from three individual physics codes. For each unique combination of physics codes, a single "Multiphysics Driver" plus a "Model Evaluator" for each physics code are created. LIME defines the functional interface requirements that must be met for different types of coupling scenarios, and can optionally leverage the Trilinos NOX (trilinos.org, 2014) solver library for performing JFNK-based nonlinear solution of the global coupled problem.[1] The Model Evaluator for each physics code is an interface to allow various functionality of the physics code to be called. In addition, a customized data transfer operator for each direction of data exchanged between coupled physics codes must be created and made available to the Multiphysics Driver.

Details about each of these components, what they do, and how to create them for a specific coupled-code problem are described by Schmidt et al. (2011). An overview of the mathematical theory associated with multiphysics coupling in general, and multiphysics code coupling in particular as it relates to the use of LIME is given by Pawlowski et al. (2011).

As noted in Fig. 2, data must be transferred between the participating physics codes in order to obtain a solution to the overall coupled physics problem. The characteristics of this data transfer can vary widely based on the different physics codes being coupled, the spatial and temporal domain and discretization of the problem, and so forth. LIME does not prescribe the details of the data transfer operation, so this important aspect of creating a coupled-code application is left to the multiphysics code developer to write and properly implement. Depending on the problem being solved, and for serial computer codes, writing the needed data transfer operators can often be a straightforward exercise. However, there are several issues that can complicate the data transfer, including inconsistent spatial discretizations (i.e. non-conformal mesh differences) and/or different spatial discretization schemes. And for codes written to execute in parallel, there are additional challenges because memory may be distributed among different processors. This is illustrated in Fig. 3, where two physics codes (whose physics equations are to be coupled) share the same simple rectangular problem domain, but have different mesh discretizations and different parallel decompositions.

As a worst case one can imagine a problem scenario where the coupled-code geometric domains do not conform spatially, the discretization schemes are different (e.g. finite element vs. finite volume), the parallel decompositions are uncorrelated and completely independent of one another, and the problem requires adaptive meshing where fields must be moved between meshes after refining and coarsening. Research continues on how to best address these various challenges, but an important step has been taken by developing software to address the parallel decomposition issues and some of the geometric mapping issues in an automated way.

The Data Transfer Kit (DTK (Slattery et al., 2013) is a CASL-developed software library designed to provide parallel data transfer services for arbitrary physics components. The algorithms implemented in DTK are based on the concept of geometric rendezvous as developed by Plimpton et al. (2004) originally implemented as part of the SIERRA framework (Stewart and Edwards, 2004). The rendezvous algorithm provides a means to geometrically correlate two geometric domains that may be arbitrarily decomposed in a parallel simulation. By repartitioning both

[1] Simpler fixed-point solution strategies do not require the use of the Trilinos NOX library.

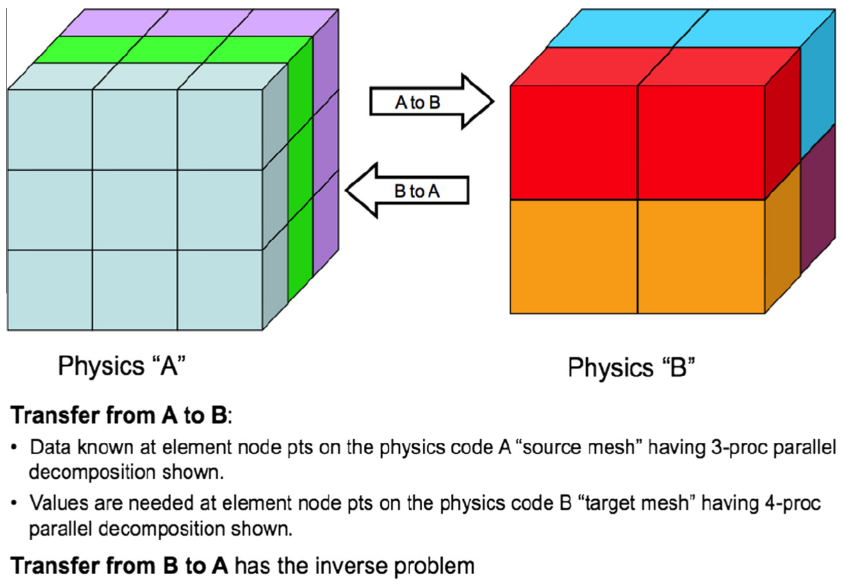

A to B
B to A
Physics "A"
Physics "B"
Transfer from A to B:
• Data known at element node pts on the physics code A "source mesh" having 3-proc parallel decomposition shown.
• Values are needed at element node pts on the physics code B "target mesh" having 4-proc parallel decomposition shown.
Transfer from B to A has the inverse problem

Fig. 3. Simple example of how transfer operations between physics codes can be complicated by different spatial discretizations and by different parallel decompositions.

domains such that they have the same geometric domain on each parallel process, efficient and load balanced search operations and data transfer can be performed at a desirable algorithmic time complexity with low communication overhead relative to other types of mapping algorithms.

In DTK, the work of Plimpton et al. (2004) has been extended to move towards a component software design for use with arbitrary physics codes such that varying representations of mesh, geometry, and fields are able to access these services. In addition, the original mesh-based rendezvous algorithms have been expanded to be used with both mesh and geometry representations of a geometric domain. Three different mappings are currently available in DTK to address different transfer needs:

1. Shared Domain Map, to transfer data from a source mesh to a target point.
2. Integral Assembly Map, to transfer data from a source mesh to a target geometric volume.
3. Shared Volume Map, to transfer data from a source geometry to a target point.

The use of small software tools such as LIME and DTK to create a single-executable coupled-code multiphysics application is a key aspect of the approach described in this paper. This approach differs from alternative approaches to creating parallel multiphysics applications that are based on placing all physics equations into a common framework. For example, in the MOOSE (Gaston et al., 2009) approach, complex multiphysics applications can be developed if all participating phenomena can be captured within the constraints of an overall finite-element-based multiphysics framework. Here we do not compare the advantages and disadvantages of these different approaches but simply note that MOOSE-based applications are also being included in VERA (e.g. the Peregrine fuel performance code), and that these can themselves be coupled to other non-MOOSE-based applications leveraging the VERA approach and tools described here.

Compiling different physics codes together into a single-executable coupled-code application is not without challenges, especially when the physics codes rely heavily on additional third-party libraries (TPLs). To address these complications, VERA utilizes the TriBITS (see github.com/TriBITSPub/TriBITS) build system. TriBITS stands for the "Tribal Build, Integrate, and Test System" and was originally developed for Trilinos, but was later extended for VERA, SCALE and other projects. TriBITS is based on the well-known Kitware open-source toolset CMake, CTest, and CDash (see www.cmake.org). In addition to providing the build system, TriBITS also provides an integrated testing platform to help automate developer testing.

When performing coupled-code multiphysics simulations for transient problems some additional details relating to the temporal discretization of coupled problem must be addressed. At present, LIME can only support simple two-step time integrations schemes such as the Crank–Nicholson or Euler methods. Future work will look at how best to support codes that leverage higher-order multi-step methods. In this paper we focus on discussing a steady-state problem. However, the fully coupled solution strategies presented are directly relevant to all implicit formulations, as these require the time-state conditions of the coupled physics codes to be temporally consistent. In contrast, fully explicit time integration schemes are not of particular interest here because all transferred quantities between physics codes are by definition time-lagged.

2.3. VERA common input (VERAIn)

VERAIn (Palmtag, 2013a) is an ASCII input file designed to contain all information needed to define and model a nuclear reactor core using the VERA-CS.

Defining and using a single common input file has several benefits. First, it provides a mechanism to insure consistency-of-representation between different codes that may potentially be coupled together when solving a problem. All input data required by physics codes being used for a given problem is automatically generated from VERAIn, eliminating errors due to inconsistencies that might arise if a user was required to do this manually. Second, it enables a user to be productive without having to first learn the detailed input requirements of each code that might be leveraged. Third, the VERAIn file can provide an important aspect of clearly documenting any analysis performed using the VERA-CS. Finally, it can be structured in an intuitive way to describe a nuclear reactor core.

VERAIn uses a free-form input style that is based on keyword inputs. The format was designed with current industry core design tools in mind and with the goal that it would be easy for industry users to understand. Choosing to use a simple ASCII input file format has several advantages:

- Users can easily transfer input and output between different computer systems.
- Users can easily edit the file on remote computers.
- Users can easily "diff" input files on a variety of remote computers.
- The ASCII format is a format that users can readily read and understand.
- ASCII input files are an approved archive format recognized by the NRC (ASCII, PDF, or TIFF).
- Users can archive inputs in standard source code repositories and/or directories with read-only permissions.

VERAIn has been designed to have an intuitive and flexible structure with the idea of building up the definition of the core geometry in a systematic fashion. To do so VERAIn is divided into groups of input blocks summarized in Table 1. Not all blocks are required for all problems.

Note that in Table 1 the last entry refers to "*Code Specific*" blocks. Current examples of these include CTF, INSILICO, MPACT, and MAMBA. As VERA matures and additional physics codes become available, additional code specific blocks will be added.

A partial listing of the VERAIn file defined for the example problem discussed in Section 3 is provided in the Appendix A. This provides a concrete example of the simple and logical format of the VERAIn file used in this approach. A complete and detailed description of the VERAIn file and all available parameters is included in documentation that is included with a VERA installation.

2.4. Processing the VERAIn common input file

Fig. 4 conceptually illustrates how the information contained in the VERAIn file is processed to create the native input files needed by the physics codes in the VERA-CS for a particular multi-physics simulation. This is essentially a two-step process that begins with an input parser reading the ASCII input file, processing the data, and creating an associated parameter list in the form of an XML file. The XML version of the common input data serves as an equivalent representation of the problem definition that can be viewed by standard XML file browsers.

Fig. 5 gives an example of how the XML file can be viewed in a web browser by employing an Extensible Style sheet Language Transformation (XSLT) developed in VERA. The XML file also provides a means to address other forms of input such as graphical user interfaces.

In step 2, the XML-based file is processed by code-specific input adapters to produce native inputs for the individual VERA physics codes and for the LIME coupling software. Of fundamental importance is that the input data used by each physics code is generated from the same VERAIn data, eliminating errors due to inconsistencies that might arise if these files were created independently.

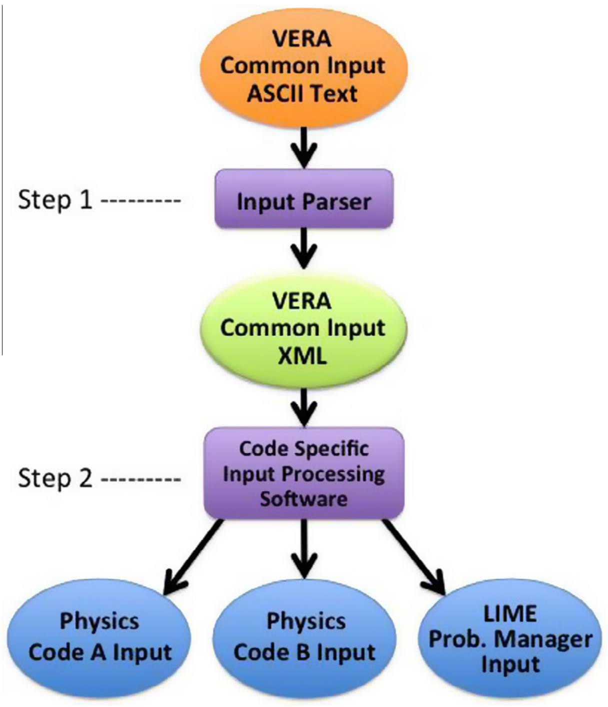


Fig. 4. Steps to processing the VERA Common Input file.

The small utility codes (the input parser and the code specific XML input adapters) required for steps 1 and 2 are part of the VERA infrastructure tools and stored in the VERA repository.

2.5. Submitting a problem for solution

Once a VERAIn file has been created by a user, the process for submitting the problem for solution consists of three steps. The first two were described above in Section 2.4 and illustrated in Fig. 4. Step 3 is simply submitting the job for execution.

When VERA is properly installed, all required executables are compiled and available for use in known locations on the computer system. Thus the three steps are typically automated by creating a shell script. For example, given a VERAIn file named "my_vci_file.inp" that uses the couple-code executable "VRIPSScobra_denovo_coupled.exe", a shell script containing something

Table 1
List of blocks in the VERAIn input file.

Name	Comments	Required
CASEID	Defines the problem name	No
STATE	Describes the reactor state at the current depletion step, e.g. values such as power level, flow, rod positions, etc	Yes
CORE	Provides the geometric description of any core-wide parameters, including maps of assembly layouts	Yes
ASSEMBLIES	Holds sublists for all assemblies in the reactor core. These sublists in turn include additional sublists for Fuel, Cells, CellMaps, and SpacerGrids	Yes
CONTROLS	Holds sublists for all control rod descriptions. These sublists include additional sublists for Materials, Cells, and CellMaps	No
INSERTS	Defines the removable burnable poison inserts that are inserted into the fuel assemblies	No
DETECTORS	Describes the detector inserts that are inserted into the fuel assemblies	No
EDITS	Values controlling what information to output	No
COUPLING	Code-coupling parameters such as convergence criteria, relaxation factors, etc	No
[*Code Specific*]	Code specific options, settings and parameters to be defined, which are unique to the VERA Physics codes	No

similar to the following would be run in a directory containing the VERAIn file;

```
#1 Convert ASCII text to XML using the VERAIn utility:
[Path_1]/react2xml.pl my_vci_file.inp my_vci_file.xml

#2 Generate COBRA-TF input file using VERA and CTF utilities
# Note Insilico reads its input directly from the vci.xml file
[Path_2]/xml2cs.exe –xmlfile=my_vci_file.xml

#3 Run the coupled cobra_denovo executable using 36
   processors
mpirun –n 36 [path_3]/VRIPSScobra_denovo_coupled.exe
   deck.inp
```

where *[path_1]*, *[path_2]* ... etc. must contain the full paths to the associated executables. In practice, the shell script will have additional functionality to generalize file naming, redirect output, do error checking and so forth. But the above lines illustrate the essential steps.

3. Example solution of a multiphysics coupled-code problem

The purpose of this section is to illustrate use of the VERA-CS approach on a representative multiphysics coupled-code problem of interest to CASL.

3.1. Problem description and its VERAIn file

Our example problem is an eigenvalue and power distribution calculation of a representative three-dimensional 17×17 assembly at Hot Full Power (HFP) conditions with thermal–hydraulic and fuel temperature feedback. The assembly is a standard Westinghouse fuel design with uniform fuel enrichment based on the dimensions and state conditions of Watts Bar Unit 1 Cycle 1. There are no axial blankets or enrichment zones. The assembly has 264 fuel rods, 24 guide tubes, and a single instrument tube in the center. There are no control rods or removable burnable absorber assemblies in this problem. Details of the geometric and material specifications are provided Palmtag (2013b). All dimensions are non-proprietary and were derived from the publically available Watts Bar Unit 1 FSAR (U.S. NRC, 2009).

Selected thermal–hydraulic conditions for this problem are listed in Table 2.

A partial listing of the VERAIn file used for this example problem is listed in the Appendix A (a complete listing is available elsewhere (Palmtag (2013b). Note that the five values listed in Table 2 are found on lines 9, 10, 11, and 19, respectively. Of note is the simplicity and readability of the input format.

3.2. Participating physics codes

All neutronics aspects of the problem (cross-section calculation, neutron transport, power release) are solved using the Insilico code suite and all thermal–hydraulic aspects are calculated using the Cobra-TF (CTF) code. The two codes are coupled together using VERA infrastructure tools (LIME, DTK) to create a single executable coupled-code multiphysics application. The different physics associated with these two codes are strongly coupled and nonlinear.

3.2.1. Insilico

Insilico is the reactor toolkit package of Exnihilo and includes the cross section generation package based on XSProc. Insilico uses the Denovo module (Evans et al., 2010; Davidson et al., 2010) to solve for the flux and eigenvalue solutions of a 3D problem using either a discrete ordinates (S_N) solver or the Simplified Legendre (SP_N) solver. All of the results in this paper were generated with the S_N solver.

Multigroup cross sections are generated in Insilico using the SCALE code XSProc. XSProc performs resonance self-shielding with full range Bondarenko factors using either the narrow resonance approximation or the intermediate resonance approximation. The fine energy group structure of the resonance self-shielding calculation can optionally be collapsed to a coarse group structure through a one-dimensional (1D) discrete ordinates transport calculation internal to XSProc. For our example problem the fine energy group structure is collapsed to a 23-group coarse group structure to be used in the Denovo transport solver.

The cross section library used in this study is the SCALE 6.2 252 group ENDF/B-VII.0 neutron cross section data library. This library contains data for 417 nuclides and 19 thermal-scattering moderators.

3.2.2. Cobra-TF (CTF)

COBRA-TF (CTF) is a thermal–hydraulic simulation code designed for Light Water Reactor (LWR) analysis (Avramova, 2009; Salko and Avramova, 2012; Salko et al., this issue). CTF has a long lineage that goes back to the original COBRA computer developed in 1980 by Pacific Northwest Laboratory under sponsorship of the Nuclear Regulatory Commission (NRC). The original COBRA began as a thermal–hydraulic rod-bundle analysis code, but subsequent versions of the code have been continually updated and expanded over the past several decades to cover almost all of the steady-state and transient analysis of both PWR's and BWR's. CTF has been developed and maintained by the Reactor Dynamics and Fuel Management Group (RDFMG) at the Pennsylvania State University (PSU), and is currently being collaboratively improved as part of the CASL project.

CTF includes a wide range of thermal–hydraulic models important to LWR safety analysis including flow regime dependent two-phase wall heat transfer, inter-phase heat transfer and drag, droplet breakup, and quench-front tracking. CTF also includes several internal models to help facilitate the simulation of actual fuel assemblies. These models include spacer grid models, a fuel rod conduction model, and built-in material properties for both the structural materials and the coolant (i.e. steam tables).

CTF uses a two-fluid, three-field representation of the two-phase flow. The equations and fields solved are:

- Continuous vapor (mass, momentum and energy).
- Continuous liquid (mass, momentum and energy).
- Entrained liquid drops (mass and momentum).
- Non-condensable gas mixture (mass).

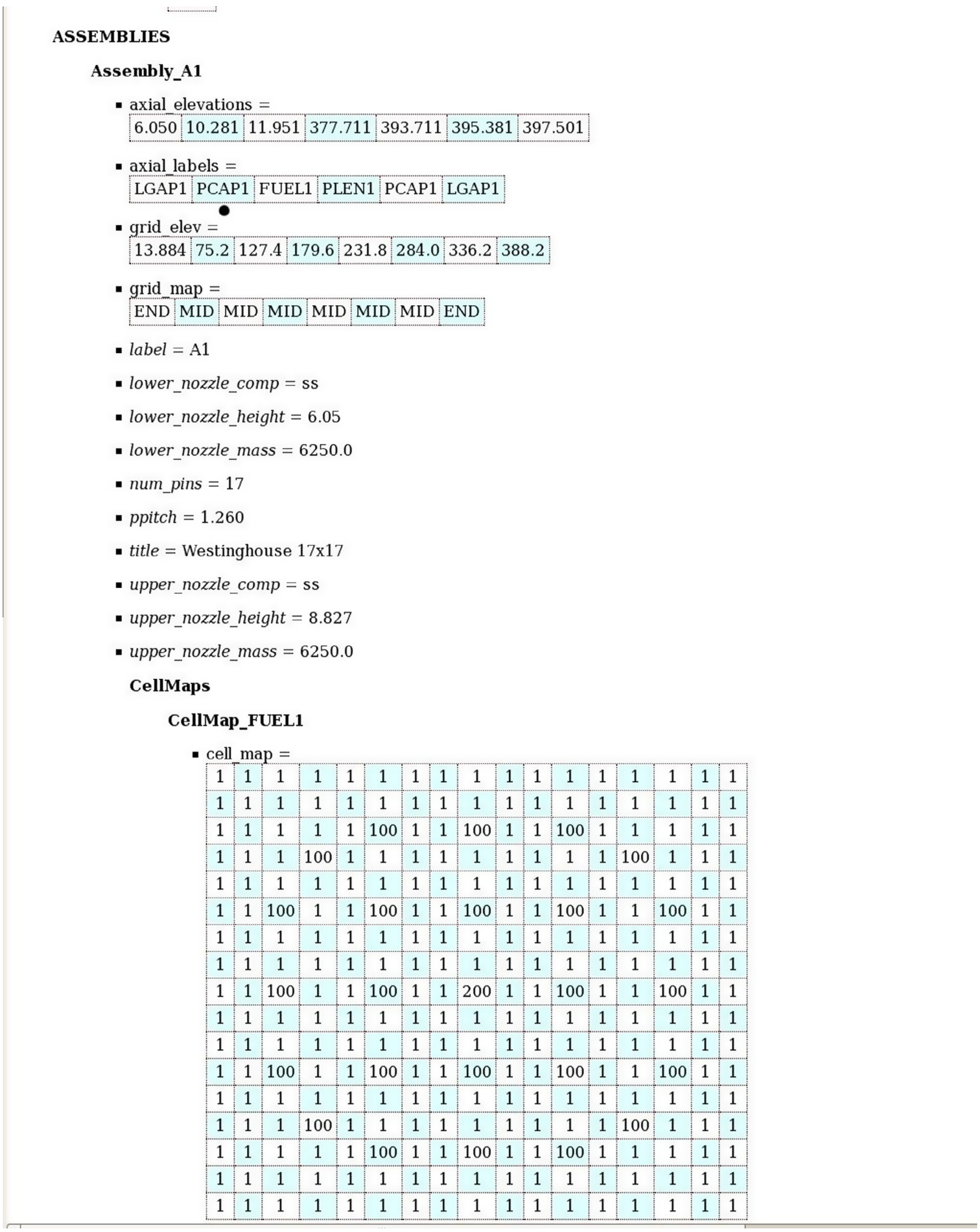
3.2.3. Coupled Insilico-CTF

Fig. 6 is a coupling diagram of the Insilico-CTF coupled-code simulation tool used to solve our example problem. In this figure the role of DTK in transferring data between the coupled codes is explicitly shown (compare to Fig. 2).

To solve the neutronics part of the overall problem, Insilico must have values for the following quantities associated with each fuel pin in each axial level:

1. Average fuel temperature, T_f
2. Average clad temperature, T_c
3. Coolant temperature, T
4. Coolant density, ρ

These quantities are calculated in the Cobra-TF code and transferred to Insilico at designated times during the overall solution procedure. Of note is that Insilico is itself solving a multiphysics



ASSEMBLIES

Assembly_A1

- axial_elevations =

6.050	10.281	11.951	377.711	393.711	395.381	397.501

- axial_labels =

LGAP1	PCAP1	FUEL1	PLEN1	PCAP1	LGAP1

- grid_elev =

13.884	75.2	127.4	179.6	231.8	284.0	336.2	388.2

- grid_map =

END	MID	MID	MID	MID	MID	MID	END

- *label* = A1
- *lower_nozzle_comp* = ss
- *lower_nozzle_height* = 6.05
- *lower_nozzle_mass* = 6250.0
- *num_pins* = 17
- *ppitch* = 1.260
- *title* = Westinghouse 17x17
- *upper_nozzle_comp* = ss
- *upper_nozzle_height* = 8.827
- *upper_nozzle_mass* = 6250.0

CellMaps

CellMap_FUEL1

- cell_map =

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	100	1	1	100	1	1	100	1	1	1	1	1
1	1	1	100	1	1	1	1	1	1	1	1	1	100	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	100	1	1	100	1	1	100	1	1	100	1	1	100	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	100	1	1	100	1	1	200	1	1	100	1	1	100	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	100	1	1	100	1	1	100	1	1	100	1	1	100	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	100	1	1	1	1	1	1	1	1	1	100	1	1	1
1	1	1	1	1	100	1	1	100	1	1	100	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig. 5. Browser rendering via XSLT of part of XML input for ASCII text in the Appendix A.

Table 2
The nominal thermal–hydraulic conditions.

Parameter	Value	Unit
Inlet temperature	559	°F
Boron concentration	600	ppm
System pressure	2250	psia
Rated flow (100% flow)	0.6824	Mlb/h
Rated power (100% power)	17.67	MWt

neutronics problem that involves calculating cross sections, neutron transport, and energy release.

Conversely, to solve the thermal–hydraulics part of the problem, Cobra-TF needs the energy release rate Q in each fuel pin at each axial level. These values are computed by Insilico and transferred to Cobra-TF. Cobra-TF also solves several coupled-physics equation sets internally, i.e. conservation of mass, momentum

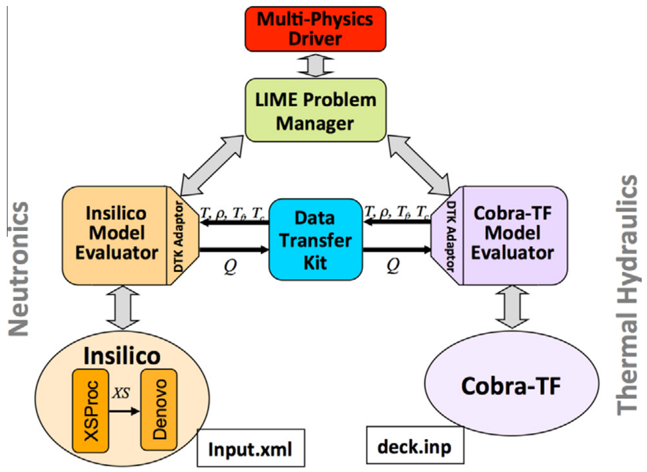


Fig. 6. Coupling diagram for the Insilico-CTF application used to solve the example problem.

and energy in the fluid together with heat transfer to fuel rods where energy is being released and conducted within the rods.

The transfer of data between Insilico and Cobra-TF is enabled and directed by several software components illustrated in Fig. 6 – Insilico and Cobra-TF Model Evaluators and DTK adaptors. These small components leverage LIME and DTK and provide the additional functionality needed to create the overall coupled-code simulation capability. In particular, they address the details of how and where the transfer data is stored in each code, and how to correctly transfer that data in the form required by both the "source" and the "target" during each transfer operation.

As described by Schmidt et al. (2011) and Pawlowski et al. (2011) LIME supports several different types of nonlinear solution strategies (i.e. Newton, JFNK, fixed-point) depending on the capabilities available from the physics codes being coupled. In this case, we solve the overall coupled nonlinear system using a simple "Seidel" mode fixed-point algorithm. This is an iterative method where each physics code is sequentially solved independently within a global iteration loop, and updated transfer data is passed between

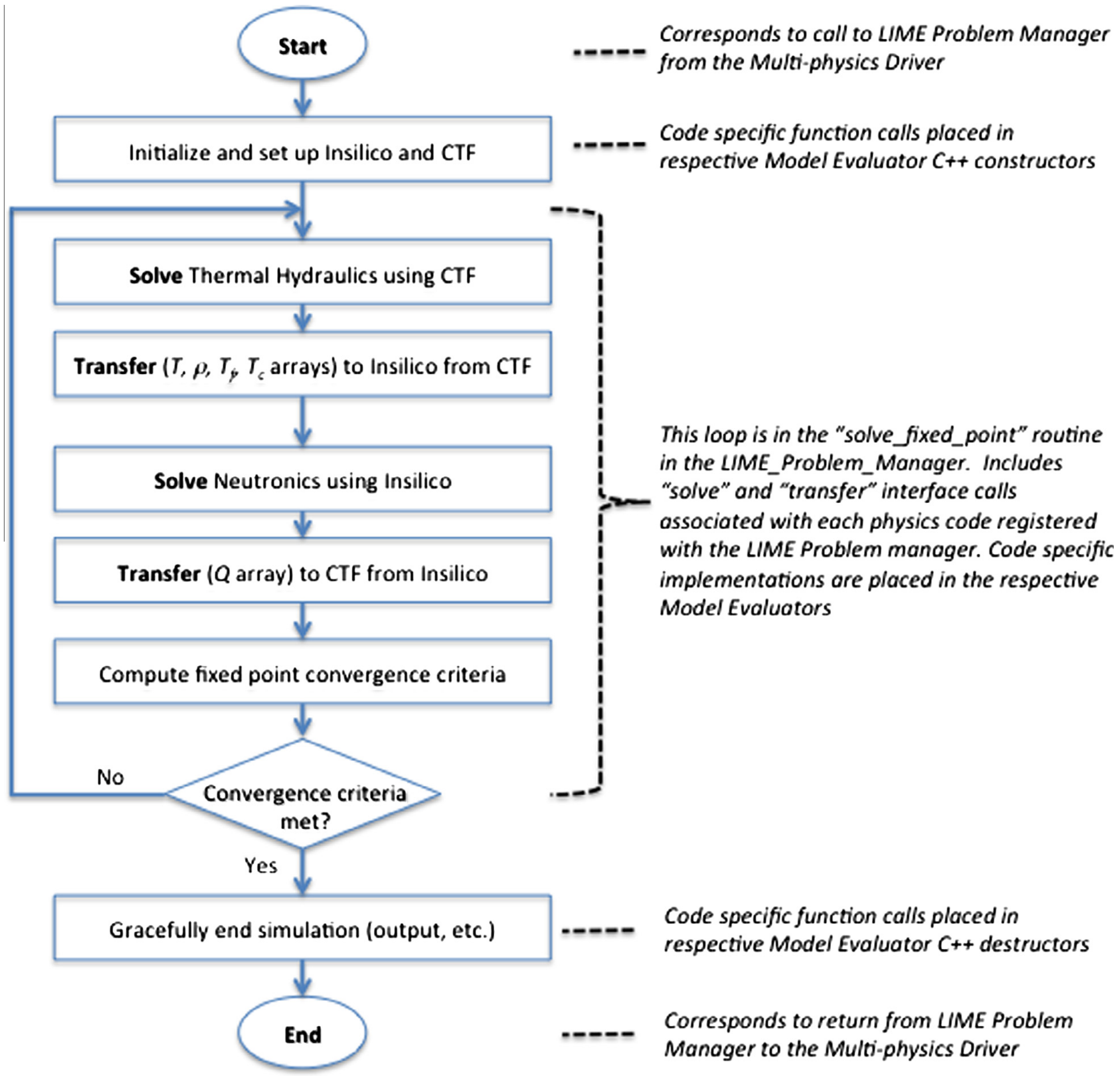
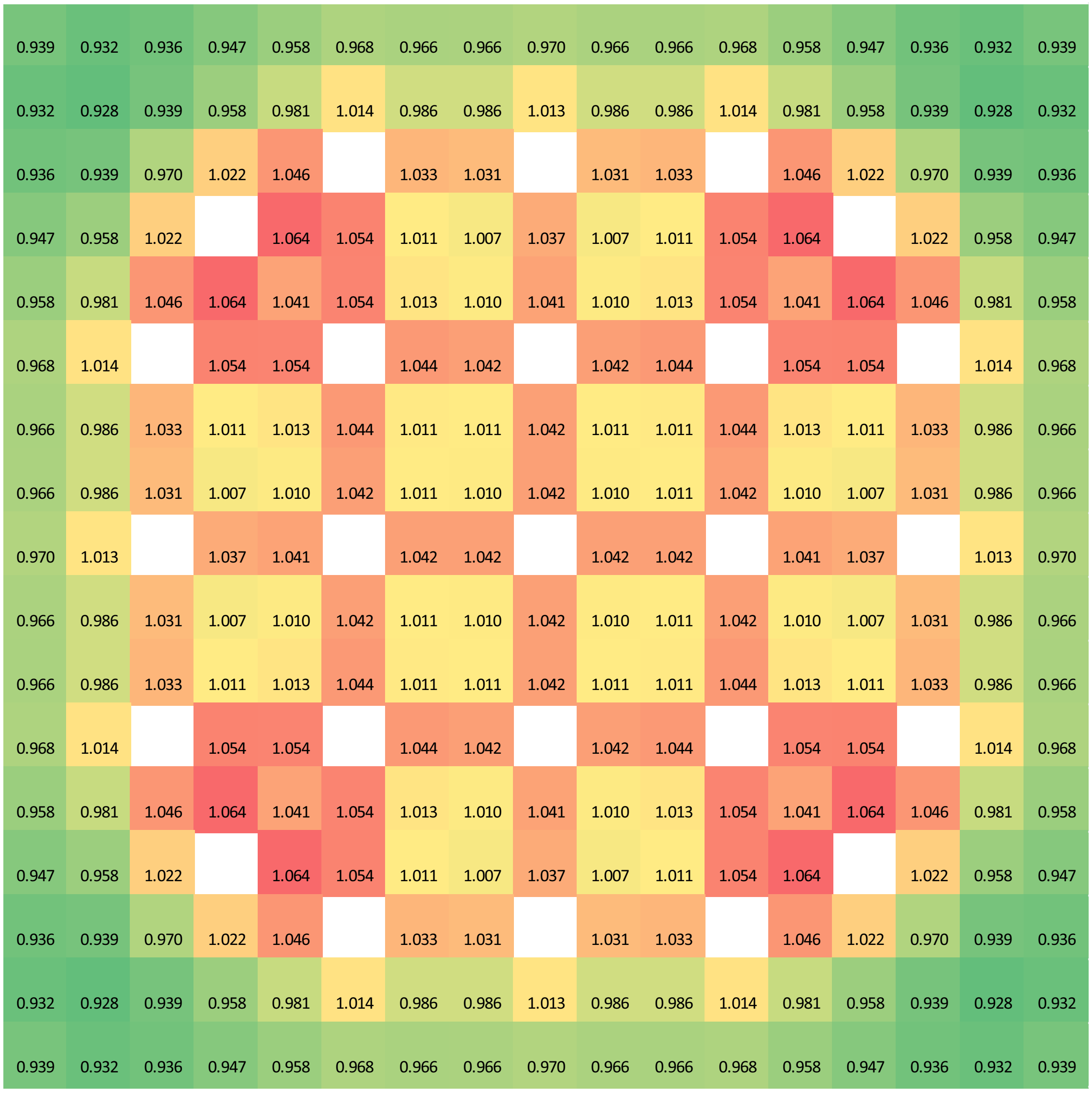

Start
Corresponds to call to LIME Problem Manager from the Multi-physics Driver
Initialize and set up Insilico and CTF
Code specific function calls placed in respective Model Evaluator C++ constructors
Solve Thermal Hydraulics using CTF
Transfer (T, ρ, T_f, T_c arrays) to Insilico from CTF
Solve Neutronics using Insilico
Transfer (Q array) to CTF from Insilico
Compute fixed point convergence criteria
This loop is in the "solve_fixed_point" routine in the LIME_Problem_Manager. Includes "solve" and "transfer" interface calls associated with each physics code registered with the LIME Problem manager. Code specific implementations are placed in the respective Model Evaluators
No
Convergence criteria met?
Yes
Gracefully end simulation (output, etc.)
Code specific function calls placed in respective Model Evaluator C++ destructors
End
Corresponds to return from LIME Problem Manager to the Multi-physics Driver

Fig. 7. Execution diagram showing how a fixed-point "Seidel" mode algorithm is used to solve the global coupled-code non-linear problem.

Table 3
Convergence history during problem solution.

fp	its	keff_dif (pcm)	power_dif	$\Delta T_{cool,mx}$ (F)	$\Delta T_{clad,mx}$ (F)	$\Delta T_{fuel,mx}$ (F)
1	27	**18947.0**	**1.000000**	**64.55**	**128.00**	**1218.00**
2	22	**60.0**	**0.222356**	0.03	**−7.00**	**−69.94**
3	19	**28.2**	**0.040185**	**0.12**	**−0.74**	**−35.53**
4	15	2.2	**0.004610**	−0.09	**0.33**	**10.28**
5	9	1.3	**0.000644**	**−0.04**	**0.31**	**4.64**
6	5	1.5	**0.000166**	**0.12**	**0.12**	**2.85**
7	2	0.8	0.000015	−0.02	0.10	**1.46**
8	1	0.1	0.000000	−0.01	0.05	**0.71**
9	1	0.1	0.000000	−0.01	0.03	**0.37**
10	1	0.1	0.000000	−0.00	0.01	**0.18**
11	1	0.1	0.000000	−0.00	0.01	0.10

fp: fixed point iteration count; its: number of Denovo iterations taken per coupled iteration; keff_dif: the change in eigenvalue between coupled iterations; power_dif: change in L2 norm of local normalized power difference between coupled iterations; ΔT: maximum change in peak temperatures (coolant, cladding, and fuel) between coupled iterations.



0.939	0.932	0.936	0.947	0.958	0.968	0.966	0.966	0.970	0.966	0.966	0.968	0.958	0.947	0.936	0.932	0.939
0.932	0.928	0.939	0.958	0.981	1.014	0.986	0.986	1.013	0.986	0.986	1.014	0.981	0.958	0.939	0.928	0.932
0.936	0.939	0.970	1.022	1.046		1.033	1.031		1.031	1.033		1.046	1.022	0.970	0.939	0.936
0.947	0.958	1.022		1.064	1.054	1.011	1.007	1.037	1.007	1.011	1.054	1.064		1.022	0.958	0.947
0.958	0.981	1.046	1.064	1.041	1.054	1.013	1.010	1.041	1.010	1.013	1.054	1.041	1.064	1.046	0.981	0.958
0.968	1.014		1.054	1.054		1.044	1.042		1.042	1.044		1.054	1.054		1.014	0.968
0.966	0.986	1.033	1.011	1.013	1.044	1.011	1.011	1.042	1.011	1.011	1.044	1.013	1.011	1.033	0.986	0.966
0.966	0.986	1.031	1.007	1.010	1.042	1.011	1.010	1.042	1.010	1.011	1.042	1.010	1.007	1.031	0.986	0.966
0.970	1.013		1.037	1.041		1.042	1.042		1.042	1.042		1.041	1.037		1.013	0.970
0.966	0.986	1.031	1.007	1.010	1.042	1.011	1.010	1.042	1.010	1.011	1.042	1.010	1.007	1.031	0.986	0.966
0.966	0.986	1.033	1.011	1.013	1.044	1.011	1.011	1.042	1.011	1.011	1.044	1.013	1.011	1.033	0.986	0.966
0.968	1.014		1.054	1.054		1.044	1.042		1.042	1.044		1.054	1.054		1.014	0.968
0.958	0.981	1.046	1.064	1.041	1.054	1.013	1.010	1.041	1.010	1.013	1.054	1.041	1.064	1.046	0.981	0.958
0.947	0.958	1.022		1.064	1.054	1.011	1.007	1.037	1.007	1.011	1.054	1.064		1.022	0.958	0.947
0.936	0.939	0.970	1.022	1.046		1.033	1.031		1.031	1.033		1.046	1.022	0.970	0.939	0.936
0.932	0.928	0.939	0.958	0.981	1.014	0.986	0.986	1.013	0.986	0.986	1.014	0.981	0.958	0.939	0.928	0.932
0.939	0.932	0.936	0.947	0.958	0.968	0.966	0.966	0.970	0.966	0.966	0.968	0.958	0.947	0.936	0.932	0.939

Fig. 8. Normalized vertically-integrated fission rate distribution across the fuel assembly.

physics codes immediately after each physics code solution. In addition, the change in transferred values between iterations is "relaxed" to improve the convergence rate of the approach.

Fig. 7 is a simplified execution diagram that illustrates the basics of how the "Seidel" mode fixed-point algorithm is implemented in the LIME Problem Manager for our example problem.

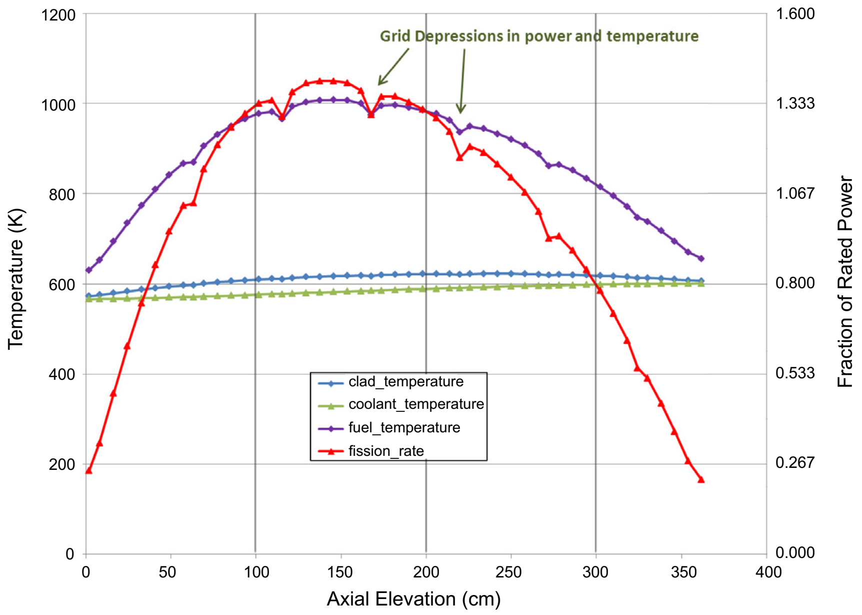


Fig. 9. Axial distribution of horizontally averaged temperatures and power (with fuel temperature shown).

Because neither Insilico nor Cobra-TF currently provide a residual vector to LIME, the convergence criteria used is based on checking that key global metrics associated with the solutions in each code have reached a steady invariant condition to within a user-specified tolerance. In this problem the following five parameters are checked for convergence within the indicated tolerance:

1. Eigenvalue	5 pcm
2. Change in local normalized power difference	1.e-4 (L2 diff/L2 value)
3. Maximum change in peak local fuel temperature	0.1°
4. Maximum change in peak local clad temperature	0.1°
5. Maximum change in peak local coolant temperature	0.1°

3.3. Selected simulation results

The example problem was run on the ORNL Titan supercomputer and selected results are presented here. Insilico calculations are the dominant computational cost and used 1156 cores (17×17 rods $\times$ 4 energy sets), whereas COBRA-TF was run in serial. Total run time for this problem is approximately 4.5 h.

In the neutronics solution, the Insilico S_N solver is used with the "qr" quadrature set (4 azimuthal angles and 4 polar angles per octant). The pincell calculation used 252 energy groups and is collapsed to 23 energy groups for the 3D transport solution. Insilico uses a 4×4 mesh in each fuel rod and a maximum 2.54 cm in the axial direction. Axial boundaries are positioned at each material and edit interfaces. The neutron flux is calculated from below the lower core plate to above the upper core plate in order to capture the axial leakage effects. Note that in this example problem a

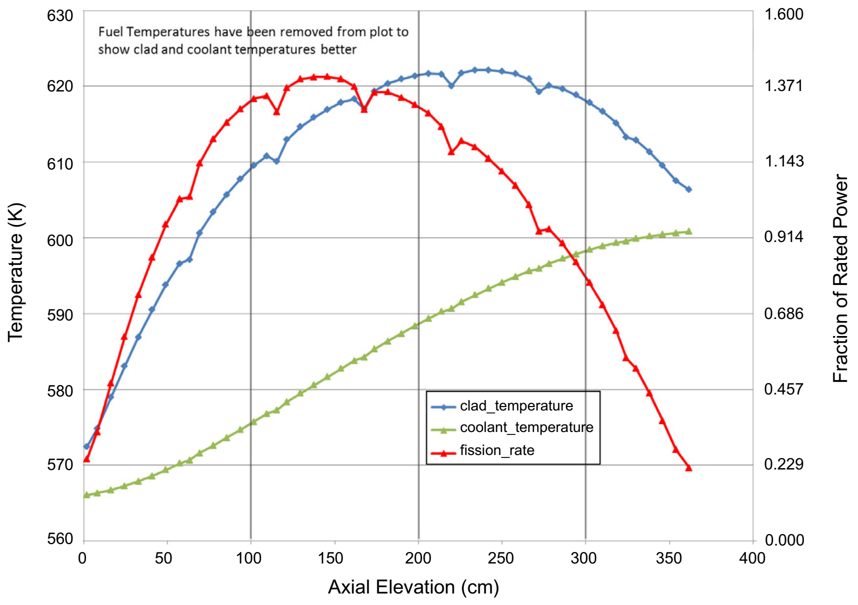


Fig. 10. Axial distribution of horizontally averaged temperatures and power (without fuel temperature).

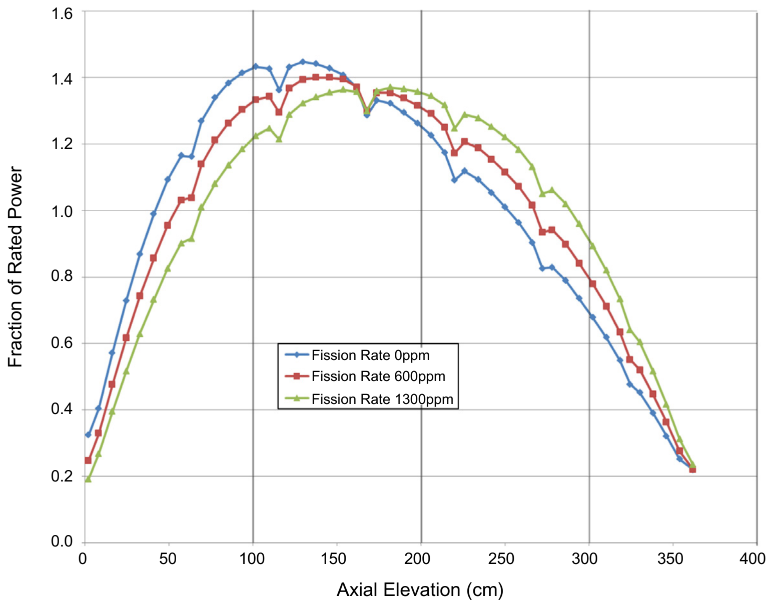


Fig. 11. Axial plot of fission rates at different boron concentrations.

somewhat coarse spatial discretization is used in Denovo to reduce problem run-times.

In the thermal–hydraulic (T/H) solution CTF has 49 axial levels over the active fuel region. The axial levels are defined to explicitly include the spacer grid heights, and to use uniform mesh spacing between the spacer grids. The maximum axial mesh is approximately 7 cm. The CTF fuel rod heat conduction model uses 5 radial rings in each fuel rod.

Table 3 shows how the convergence metric quantities evolved during the fixed point solution procedure. Values reported in bold text do not meet the convergence criteria specified. All five convergence metrics were met after 11 iterations, with the most limiting criteria being the maximum change in peak fuel temperature. Also note the large number of internal Denovo iterations required during the early fixed-point iterations, which results in a greater computational expense. As expected from using a fixed-point solution algorithm for coupling, the observed convergence rate is linear.

To illustrate results for the variation in power in the horizontal plane Fig. 8 shows the normalized fission rate distribution, integrated over the axial length. Note that the results are octant symmetric and there is no power in the guide tubes or instrument tubes.

A plot of the horizontally averaged axial distribution of power and temperatures for this problem are shown in Figs. 9 and 10. Fig. 9 includes the fuel temperature, which peaks at just above 1000 K. Fig. 10 does not include the fuel temperature so that variations in the coolant and clad temperature profiles, which peak at much lower temperatures, are easier to see. Note the small “dips” in the axial fission rate and fuel temperature profiles. These dips are due to the presence of spacer grids. The spacer grids displace moderator in the coolant channels and thus decrease the neutron

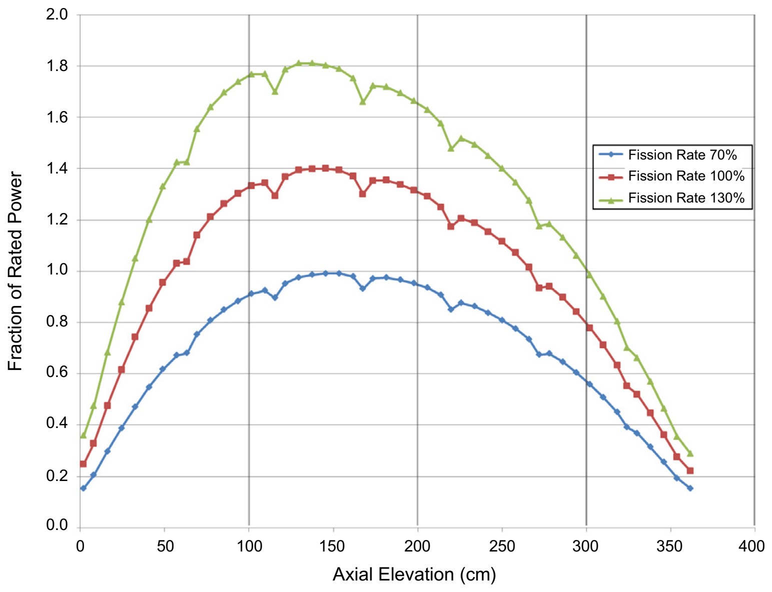


Fig. 12. Axial plot of fission rates at different power levels.

Table 4
Computed eigenvalues for nominal and perturbed cases run.

Case	Boron conc. ppm	Power level %	Eigenvalue	Coupled iterations
1	600	100	1.23344	11
2	0	100	1.31286	11
3	1300	100	1.15336	11
4	600	70	1.24012	11
5	600	130	1.22643	15

moderation around the grids, leading to a local depression in the flux and power.

Four additional runs were made to help quantify the effect of perturbing several key conditions specified for the problem. In order to see the effects of different boron concentrations on the results, the problem was rerun at boron concentrations of 0 and 1300 ppm (the nominal value was 600 ppm). To see the effects of different power levels on the results, additional runs were made at two different power levels, 70% and 130% power.

The fission rate profiles for the three different boron concentrations (0, 600, and 1300 ppm) are shown in Fig. 11. With T/H feedback, the fission rate shape is shifted lower in the core from the normal cosine-shaped distribution expected for a case with no T/H feedback. The reason for this downward shift in the fission rate is that the coolant density is higher at the bottom of the core, and the higher coolant density increases the neutron moderation. As more boron is added, the additional neutron absorption counters the higher moderation, and less power shift towards the bottom of the core is observed.

The fission rate profiles for the three power cases (70%, 100%, and 130%) are shown in Fig. 12. At higher power levels, the fission rate shape is shifted lower in the core from the normal cosine-shaped distribution for a case with no T/H feedback.

The computed eigenvalues for the nominal case and each of the perturbed conditions are shown in Table 4.

4. Concluding remarks

In this paper we have described and demonstrated an approach to coupled-code multiphysics reactor core simulations that is an important aspect of the Virtual Environment for Reactor Applications project being developed in CASL. Key elements include:

- standalone and coupled physics codes included in VERA,
- an approach and associated software to couple different physics codes together,
- a common input description that can accommodate the information needed by all participating physics codes, and
- software to process the common input into native input needed by each physics code in the coupled simulation.

Continued development and improvements to this approach are ongoing. They include expanding the number of physics codes available for use in VERA and also enhancing the capability of the current physics codes. For example, major improvements to the computational infrastructure of the CTF sub-channel code have recently been made to enable full-core “sub-channel-resolved” simulations to be performed more quickly, either in stand-alone mode or as part of coupled-code multi-physics calculations (Salko et al., 2013; Salko et al., this issue). Future work is also expected to potentially include improvements to the LIME and DTK coupling software tools and also the development or inclusion of additional capabilities to better address the complexities of data transfer between coupled codes.

Acknowledgements

This work was funded by the DOE-sponsored “Consortium for Advanced Simulation of Light Water Reactors” (CASL) project.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.

Appendix A.

A.1. Partial listing of the VERAIn File for the example problem

To illustrate clearly what a VERAIn file looks like, this appendix contains a partial input listing of the VERAIn file used for the example problem described in Section 3. Skipped lines are clearly noted in the text and a complete listing is provided by Palmtag (2013b).

```
[CASEID]
 title 'CASL Problem 6a'
!============================================================
! Sample input for Problem 6 (Single-assembly with T/H feedback)
!============================================================

[STATE]
  power  100.0        ! %
  tinlet 559.0        ! F
  boron   600         ! ppmB
  pressure 2250       ! psia
  tfuel  900.0        ! K - 600K  Not used with T/H feedback!
  modden 0.743        ! g/cc Not used with T/H feedback!
  feedback on
  sym full

[CORE]
  size 1              ! 1x1 single-assembly
  rated 17.67   0.6824  ! MW, Mlbs/hr
  apitch 21.50
  height 406.328

  core_shape
    1

  assm_map
    A1

  lower_plate ss 5.0 0.5   ! mat, thickness, vol frac
  upper_plate ss 7.6 0.5   ! mat, thickness, vol frac
  lower_ref   mod 26.0 1.0
  upper_ref   mod 25.0 1.0

  bc_rad reflecting

  mat he     0.000176
  mat inc    8.19
  mat ss     8.0
  mat zirc   6.56
  mat aic   10.20
  mat pyrex  2.23
  mat b4c    6.56

[ASSEMBLY]
  title "Westinghouse 17x17"
  npin 17
  ppitch 1.260

  fuel U31 10.257 95.0 / 3.1

  cell 1     0.4096 0.418 0.475 / U31 he zirc
  cell 100          0.561 0.602 / mod    zirc      ! guide tube
  cell 200       0.561 0.602 / mod    zirc      ! instrument tube
  cell 7         0.418 0.475 / mod    mod       ! empty location
  cell 8         0.418 0.475 /     he zirc      ! plenum
  cell 9               0.475 /        zirc      ! pincap

  lattice FUEL1
     200
       1 1
       1 1 1
     100 1 1 100
       1 1 1   1 1
       1 1 1   1 1 100
     100 1 1 100 1   1 1
       1 1 1   1 1   1 1 1
       1 1 1   1 1   1 1 1 1
.
. next 56 lines not shown, see (Palmtag, 2013b)
.

[EDITS]

!  approximately 3in intervals in active fuel
  axial_edit_bounds
         11.951
         15.817
.
. next 48 lines not shown, see (Palmtag, 2013b)
.

[COBRATF]
  nfuel   3            ! number of fuel rings in conduction model
  nc      1            ! conduction option - radial conduction
  irfc    2            ! friction factor correlation default=2
  dhfrac  0.02         ! fraction of power deposited directly into coolant
  hgap    5678.3       ! gap conductance
  epso    0.001
  oitmax  5
  iitmax  40
  gridloss END 0.9070   ! spacer grid loss coefficient
  gridloss MID 0.9065   ! spacer grid loss coefficient
  dtmin   0.000001
  dtmax   0.1
  tend    0.1
  rtwfp   1000.0
  maxits  10000
  courant 0.8

[INSILICO]
  mat_library casl_comp.sh5
  xs_library  lib252_hetbondoneabs-noabssigp

  max_delta_z   2.54
  num_blocks_i  17
  num_blocks_j  17
  num_z_blocks  12
  num_groups    23
  num_sets      4
  pin_partitioning true

  mesh          4
  dimension     3
  eq_set        sc

   eigen_solver  arnoldi
   tolerance     1e-6
   Pn_order      0
.
. next 41 lines not shown, see (Palmtag, 2013b)
.

[COUPLING]
  epsk        5.0  ! pcm
  rlx_power   0.5  ! power relaxation factor between coupled iterations
  rlx_tfuel   1.0  ! fuel temperature relaxation factor
  rlx_den     1.0  ! coolant density relaxation factor
  maxiter     100
```

References

Avramova, M.N., 2009. CTF: A Thermal-hydraulic Sub-Channel Code for LWR Transient Analyses, User's Manual. Pennsylvania State University.

Bowman, S.M., 2011. SCALE 6: comprehensive nuclear safety analysis code system. Nucl. Technol. 174 (2), 126–148 (see also scale.ornl.gov).

Davidson, G.G. et al., 2010. Massively parallel solutions to the k-eigenvalue problem. Trans. Am. Nucl. Soc. 103, 318–320.

Evans, T.M. et al., 2010. DENOVO: A new three-dimensional parallel discrete ordinates code in SCALE. Nuclear Technol. 171, 171–200.

Gaston, D. et al., 2009. MOOSE: a parallel computational framework for coupled systems of nonlinear equations. Nuclear Eng. Des. 239 (10), 1768–1778.

Hales, J.D. et al., 2013. BISON theory manual: The equations behind nuclear fuel analysis, Technical Report, Idaho National Laboratory, Idaho Falls, ID.

Kendrick, B.K., Barber, J., 2012. Initial validation and benchmark study of 3D MAMBA v2.0 against the WALT loop experiment and BOA v3.0, Internal CASL Technical Report, CASL-I-2012-0238-000.

Kochunas, B. et al., 2013. Overview of development and design of MPACT: Michigan parallel characteristics transport code, In: Proceedings of M&C 2013, Sun Valley, Idaho.

Mervin, B.T., 2013, Monte Carlo and depletion reactor analysis for high-performance computing applications, Doctoral Dissertation, University of Tennessee, Knoxville.

Nourgaliev, R. et al., 2013. Hydra-TH: a thermal-hydraulics code for nuclear reactor applications. In: Proceedings of NURETH-15, Pisa, Italy.

Palmtag, S., 2013a. User manual for the VERA input processor, CASL Technical Report: CASL-U-2014-0014-000 (available from www.casl.gov).

Palmtag, S., 2013b. 2013. Coupled single assembly solution with VERA (Problem 6), CASL Technical Report: CASL-U-2013-0150-000, Rev. 1 (available from www.casl.gov).

Pawlowski, R., et al., 2011. A theory manual for multiphysics code coupling in LIME version 1.0, Sandia National Laboratories Technical Report, SAND2011-2195.

Plimpton, S., Hendrickson, B., Stewart, J., 2004. A parallel rendezvous algorithm for interpolation between multiple grids. Journal of Parallel and Distributed Computing 64, 266–276.

Oak Ridge National Laboratory, 2011. Scale: A Comprehensive Modeling and Simulation Suite for Nuclear Safety Analysis and Design, Version 6.1, ORNL/TM-2005/39.

Salko, R.K., Avramova, M.N., 2012. CTF Theory Manual. Pennsylvania State University.

Salko, R.K. et al., 2013. Improvements, Enhancements, and Optimizations of COBRA-TF". In: Proceedings of M&C 2013, Sun Valley, Idaho.

Salko, R.K., Schmidt, R.C., Avramova, M.N., this issue. Optimization and Parallelization of the Thermal-Hydraulic, Sub-channel Code CTF for High-Fidelity Multi-physics Applications, Annals of Nuclear Engineering.

Schmidt, R. et al., 2011. An Introduction to LIME 1.0 and it's Use in Coupling Codes for Multiphysics Simulation. Sandia National Laboratories Technical Report, SAND2011-8524.

Slattery, S.R., Wilson, P.P.H., Pawlowski, R.P., 2013. The data transfer kit: a geometric rendezvous-based tool for multiphysics data transfer. In: Proceedings of M&C 2013, Sun Valley Idaho.

Stewart, J., Edwards, H., 2004. A framework approach for developing parallel adaptive multiphysics applications. Finite Elements Anal. Des. 40, 1599–1617.

U.S. DOE, 2011. CASL – a project summary, CASL-U-2011-0025-000, Consortium for Advanced Simulation of LWRs (available from www.casl.gov).

U.S. NRC, 2009. Watts Bar Unit 2 Final Safety Analysis Report (FSAR), Amendment 93, Section 4, ML091400651 (available at http://adamswebsearch2.nrc.gov/idmws/ViewDocByAccession.asp?AccessionNumber=ML091400651).