**U. S. Department of Energy
Nuclear Engineering Research Initiative
Award Number: DE-FC07-06ID14743
Award Recipient: Purdue University**

**Uncertainty Quantification in the Reliability and Risk Assessment of
Generation IV Reactors**

**Final Scientific/Technical Report
March 13, 2006 through June 12, 2009**

Contact and Principal Investigator:

Dr. Karen Vierow
Texas A&M University (TAMU)
Department of Nuclear Engineering
3133 TAMU
College Station, TX 77843-3133
Tel.: 979-458-0600
Email: vierow@ne.tamu.edu

Co-Principal Investigator:

Dr. Tunc Aldemir
The Ohio State University (OSU)

Contributing Authors:

Mr. Brad Beeny, Undergraduate Research Assistant, TAMU
Mr. Kevin Hogan, Graduate Research Assistant, TAMU
Mr. Kyle Metzroth, Graduate Research Assistant, OSU

Submitted to the U.S. Department of Energy
September 10, 2009

Executive Summary

The project entitled, “Uncertainty Quantification in the Reliability and Risk Assessment of Generation IV Reactors”, was conducted as a DOE NERI project collaboration between Texas A&M University and The Ohio State University between March 2006 and June 2009.

The overall goal of the proposed project was to develop practical approaches and tools by which dynamic reliability and risk assessment techniques can be used to augment the uncertainty quantification process in probabilistic risk assessment (PRA) methods and PRA applications for Generation IV reactors.

Specific project objectives included the development of practical approaches and computationally efficient software package(s) to:

- 1) Test event tree completeness for Generation IV reactors
- 2) Integrate a reactor safety code with PRA
- 3) Assess and propagate uncertainty on the plantstate in PRA analysis for Generation IV reactors

The project consisted of two phases with Phase 1 for Dynamic Event Tree (DET) generation and Phase 2 for uncertainty propagation and quantification.

In Phase 1, current Dynamic Event Tree generation software was modified to accommodate linkage with a best-estimate code (MELCOR for demonstration, but any code could be linked) and the actual code linkage was performed. Key uncertainties in Generation IV modeling were identified and the integrated software package will be tested on selected initiating events.

In Phase 2, the computational efficiency was improved by coupling the Dynamic Event Tree generation software with the sampling software developed by Sandia National Laboratories. This software package will be tested for selected initiating events from Phase 1.

Research contributions to the understanding of the area investigated

The completed project contributes to the state-of-the-art in the PRA of existing reactors, as well as the future Generation IV reactors. The techniques and software have been made flexible and adaptable to a variety of reactor safety codes, thereby producing a tool that is widely usable by both the designer and the licensing/regulatory side.

A MELCOR input deck to model a Generation IV system was modified and capabilities were extended to enable simulations of the chosen event, a Pressurized Loss of Forced Circulation event.

An uncertainty quantification algorithm has been developed that includes both uncertainty analysis and sensitivity analysis. The form determines the distribution of an output value, such as the Figure of Merit, caused by variations in the input. The latter shows to what extent variations on each input parameter affect the output value. A two-loop algorithm for uncertainty propagation and quantification was constructed with an inner loop treating the correlated uncertainties and an outer loop treating the uncorrelated uncertainty data.

The project has confirmed the ability to use the DET generation software with a reactor safety code for integrated analysis.

Technical effectiveness and economic feasibility of the methods or techniques investigated or demonstrated.

One of the major additions to the software is the DET generation software web interface. The web interface provides a more user-friendly environment for executing experiments. Functions include serving as a front-end to the server and database management system, launching experiments, visualizing the current event tree status and downloading all files associated with a particular branch.

The DET generation software has been developed in a code-neutral manner, enabling its use with a variety of reactor safety codes and simulators.

By performing experiments with various techniques and variables, the computational infrastructure has been made more efficient with respect to the required number of cases that need to be run. For the sensitivity analysis, three experiment designs were employed: full factorial; Latin Hypercube; and Taguchi orthogonal array. Parameters were ranked by Response Surface, Linear Correlation Coefficient and Partial Correlation Coefficient methods.

Benefit to the public of the project

Specific benefits of the computational infrastructure regarding Generation IV systems include:

- 1) Determination of the pathway of a scenario by mechanistic reactor safety codes within a probabilistic context. Each of the scenario sequences determined by ADAPT is a pathway.
- 2) Elimination of unnecessary conservatisms in design due to reduced uncertainty in calculated plant performance parameters
- 3) New PRA techniques to select the initiating events with the highest risk profile. If the Figure of Merit is low risk, the scenarios, and hence the initiating events leading to these scenarios, can be ranked with regard to their contribution to risk. The novelty in this approach is the use of Taguchi arrays to account for the epistemic uncertainties in the initiating events, which could affect the ranking.
- 4) Timely guidance in the design and PRA-informed licensing and regulation of Generation IV systems
- 5) Characterization of the safety performance and accident behavior of Generation IV reactors

Table of Contents

List of Figures	iv
List of Tables	v
1. Comparison of Actual Accomplishments with Project Goals and Objectives	1
2. Project Activities	3
3. References	18
4. Products Developed Under the Award and Technology Transfer Activities	19
Appendix A MELCOR PBMR Input Deck Revisions for the Current Project	A-1
Appendix B ADAPT Admin Guide	B-1
Appendix C ADAPT Users Guide	C-1

List of Figures

Figure 1 Revised MELCOR PBMR Model Nodalization	10
Figure 2 Quantification of the epistemic uncertainties associated with simulator input parameter $a \leq x \leq b$ for two example pdfs (uniform, non-uniform) by fixed points x_n ($n=1, \dots, N$)	13
Figure 3 Cumulative probability of core stored energy resulting from 250 Latin Hypercube simulations (epistemic-only analysis)	15
Figure 4 Results of the both the combined analysis and the epistemic only analysis	15

List of Tables

Table 1	Proposed Task Schedule.....	1
Table 2	Project Milestones.....	2
Table 3	Uncertainty Parameter Values and Ranges.....	9
Table 4	Results of the Parameter Rankings.....	17

1. Comparison of Actual Accomplishments with Project Goals and Objectives

The actual accomplishments are compared with the goals and objectives described in Section 4 of the proposal's Project Narrative and the schedule is in the Project Management Plan.

All tasks of Phase 1 were completed by March 12, 2009. In the third year of the project, for Phase 1 Task 3, values for the key uncertainties and ranges of these values to test were decided. The values and ranges were incorporated into the analysis.

All tasks of Phase 2 were completed by the end of the no-cost extension period, June 12, 2009.

Table 1 Proposed Task Schedule

Task	2006			2007				2008				2009
Phase 1: DET Generation												
1. Install PVM and communication software												
2. Linkage with MELCOR												
3. Determination of key modeling uncertainties												
4. Software testing												
Phase 2: Uncertainty Propagation and Quantification												
1. Linking MELCOR/DET to Monte Carlo/Latin Hypercube Sampling tool												
2. Testing of Phase 2 Task 1 software												
3. Modification of methodology to purely numerical algorithm												
4. Incorporation of Phase 2 Task 3 Software to MELCOR/DET												
5. Testing of Phase 2 Task 4 software												
6. User's manual documentation												

Table 2 Project Milestones

Task / Milestone Description	Planned Completion	Actual Completion	Comments
Phase 1: DET Generation			
1. Install PVM and communication software	10/30/06	10/30/06	
2. Linkage with MELCOR	12/31/06	06/12/07	Issues with installation on the TAMU computing system
3. Determination of key modeling uncertainties	09/30/08	04/01/09	Modified per preliminary analysis
4. Software testing	06/30/07	06/30/07	
Phase 2: Uncertainty Propagation and Quantification			
1. Linking MELCOR/DET to Monte Carlo/Latin Hypercube Sampling tool	12/31/07	03/12/09	Change in methodology
2. Testing of Phase 2 Task 1 software	06/30/08	06/30/08	Task as originally defined was completed on schedule. Since further features were found to be necessary, the task was extended.
3. Modification of methodology to purely numerical algorithm	09/30/08	03/12/09	
4. Incorporation of Phase 2 Task 3 Software to MELCOR/DET	12/31/08	03/12/09	
5. Testing of Phase 2 Task 4 software	03/12/09	06/12/09	
6. User's manual documentation	03/12/09	06/12/09	

2. Project Activities

The activities scheduled for each year of the project are listed in the Project Management Plan and detailed in Section 4 of the proposal's Project Narrative. The research accomplishments for the project duration are discussed below.

Phase 1: Dynamic Event Tree Generation

- Task 1: Development of the SCHEDULER, PROBABILITY MODULE and DATABASE MANAGEMENT SYSTEM
- Task 2: Linking the SCHEDULER and the DATABASE MANAGEMENT SYSTEM to MELCOR
- Task 3: Determination of key uncertainties in reactor modeling and high risk initiating events
- Task 4: Testing of the software on selected initiating events

A computational infrastructure has been developed outside of this NERI project that supports the generation of multiple dynamic event trees (DETs) on a distributed computing architecture composed of a heterogeneous collection of computational and storage nodes. The DET generation is managed by a driver called ADAPT (Analysis of Dynamic Accident Progression Trees) that 1) determines when branching is to occur, 2) initiates multiple restarts of system code analyses, 3) determines the probabilities of scenarios, 4) determines when a scenario can be terminated, and 5) combines similar scenarios to reduce the scope of the analysis [Rutt, 2006, Hakobyan, 2006].

Task 1: Development of the SCHEDULER, PROBABILITY MODULE and DATABASE MANAGEMENT SYSTEM

New features were added to the DET generation software server and graphical user interface (GUI) specifically in the SCHEDULER, PROBABILITY MODULE and DATABASE MANAGEMENT SYSTEM. Initially, the user could only launch experiments via a command line utility and view the software-produced event tree with a Java GUI with limited analysis capabilities. The new features added allow for better user control of experiment execution and better front-end tools for ease of use. Also, analysis capabilities have been added to the GUI which are described below.

A few additions have been made to the server software itself with regards to the SCHEDULER and the PROBABILITY MODULE. First, the SCHEDULER now has an option to prefer assignment of child tasks to the parent's host. This feature will help to reduce the startup time between parent and child branches. Secondly, the PROBABILITY MODULE has been altered to allow the user the option of truncating scenarios based off of a probability threshold. The PROBABILITY MODULE keeps track of individual branch probabilities and scenario probabilities. With the new feature, the child branch is not executed when a new child branch is to be submitted and its computed probability falls below the user-desired threshold. This helps to prevent a numerical catastrophe in running too many branches.

Several advancements have been made in the Java GUI which acts as a front-end to the DATABASE MANAGEMENT SYSTEM. The Java GUI is client software which is installed on the user's local machine and will download information from the DET generation software database. The Java GUI allows the same functionality as the web interface (launch experiments, download files, etc.) as well as some analysis capabilities. The coupling of the GUI with STORM [Narayanan et. al, 2003] (which runs server-side) allows for the concatenation of a branch's plot files with that of its parent's allowing the user to view the entire history of that particular scenario. With this coupling, the user can plot selected attributes from simulator output files within the GUI. The user may place many different plot variables on one plot, combine multiple single-variable plots into one and quantify physical variables for branches

or scenarios. This capability allows the grouping of branches and scenarios based off of user-specified bins of plot data.

Finally, one of the major additions to the software is the DET generation software web interface. The web interface serves as a front-end to the server and DATABASE MANAGEMENT SYSTEM and can be used to launch experiments, register new types of simulators, halt and resume experiments, visualize the current event tree status, and download all files associated with a particular branch. The web interface can be viewed from any internet browser and outside of the web-server host's local network (barring firewall restrictions). The web interface has provided a more user-friendly means of executing experiments since a command line tool is no longer necessary. This interface does not contain any analysis tools and was designed as a first step in making a more user-friendly experiment execution system.

Task 2: Linking the SCHEDULER and the DATABASE MANAGEMENT SYSTEM to MELCOR

A plant simulator is being used to follow the transient along each branch. The ADAPT framework is designed to be simulator-independent, however, MELCOR was linked to ADAPT for demonstration purposes. Branches are generated using user-specified criteria which are supplied in a simulator-independent input file. MELCOR Control Functions are utilized to provide the necessary simulator control for use with ADAPT. The Ohio State University developers of the computational infrastructure provided a training session on their software in September 2006. This included explanation of the system requirements, software capabilities, overview of the software, demonstration of a test problem, and many other aspects of the software. The TAMU PI and one grad student attended the training session. The PI and student provided input to the infrastructure developers regarding operating system compatibility, output parameters and other factors that would render the software compatible with TAMU user equipment and the MELCOR code.

Task 3: Determination of key uncertainties in reactor modeling and high risk initiating events

The detailed consideration of the key uncertainties and event selection that were conducted are described below. A Generation IV Pebble Bed Modular Reactor (PBMR) was selected as the reactor design to be simulated. This high-temperature gas-cooled reactor design is a good test of the ADAPT framework and computational system. Further, the MELCOR input deck was modified to accommodate the event and evaluation of the figures of merit.

Considerations for Selection of Event to Analyze

Gas-cooled reactors are very different from light water reactors with respect to the safety challenges they present. First, the coolant is not one which can vaporize and result in fuel overheating by dryout. Light water has been replaced by a gas-phase coolant and passive heat removal systems which prevent significant fuel failure, and therefore, reduce risk to the public. This passive decay heat removal by conduction through reactor structures to the environment is made possible by the lower power densities and power ratings. Second, the containment for the pebble bed fuel is considered to be the protective layers on the pebbles themselves. Therefore, some gas-cooled reactor proponents maintain that the safety goal is to confirm that risk to the plant investment due to elevated temperatures that shorten component lifetimes is at an acceptable level. Past PIRTs on gas-cooled reactors are reviewed below to analyze gas-cooled reactor performance in the context of these considerations.

Previous PIRTs and PBMR Safety Descriptions

A description of the PBMR previously slated for construction by Eskom in South Africa is used for this analysis. The safety criteria, or Figures of Merit, are [IAEA, 2001]:

- integrated heat loss from the reactor that exceeds the decay heat production under hypothetical accident conditions
- peak core temperature below the demonstrated fuel degradation temperature (1600°C)
- oxygen concentrations in high-temperature regions of the core (>1000°C) that are not sufficient to induce graphite oxidation

The Reactor Core Cooling System (RCCS) dissipates decay heat during a loss of heat transfer functions, such as a loss of forced cooling. Air convection and thermal radiation transport heat from the uninsulated reactor vessel walls to the RCCS cooling panels. The RCCS does not provide the safety function of keeping the nuclear fuel below its maximum design pressure. Instead, the RCCS protects the reactor vessel, supports, instrumentation and concrete. Heat transport through the concrete structures of the reactor building provides this function [p. 53 of IAEA, 2001].

Under loss of coolant flow, heat conduction and radiation to the reactor vessel and then heat transport through the passive heat removal system to the reactor cavity will prevent the maximum fuel temperature from reaching 1600°C and the reactor vessel temperature from reaching its material temperature limit [p. 62 of IAEA, 2001]. No other events are specifically discussed.

Vilim et al. [2004] set out to qualify NGNP thermal hydraulic codes using the CSAU methodology. The goal was to identify needs for extended modeling capabilities to assure proper representation of the phenomena and to address safety and design issues. As the first step, a Phenomena Identification and Ranking Technique (PIRT) was conducted to determine key events and phenomena on an operating regime and equipment-based approach. The GT-MHR 600 MWt design by General Atomics was selected as the reference Very High Temperature Reactor design.

For the first year of the project which is reported by Vilim, a Loss of Forced Cooling (LOFC) event, also known as a “conduction cooldown” event, was selected. Additional events will be analyzed in future years of the project. LOFC corresponds to loss of both the Heat Transport System and the Shutdown Cooling Circulator System. The RCCS then becomes the ultimate heat sink for decay heat. Based on material restrictions, the Figures of Merit are:

- a maximum fuel temperature below 1600°C
- a maximum reactor vessel temperature below 425°C pressurized and 530°C depressurized

Three key variables with respect to the operating conditions are:

- pressurized vs. depressurized primary system
- forced convection vs. conduction cooling
- neutronic power vs. decay heat source

From this report, an appropriate scenario for the DOE NERI project is a depressurized conduction cooling scenario in which the reactor is scrammed and decay heat must be removed. Phenomena associated with this scenario include:

- multi-fluid coolant fluid flow, as air or water may enter the system
- decay heat release from the fueled pebbles
- thermal stratification in the lower plenum
- jet discharging into the lower plenum

A PIRT was performed on GA’s MHTGR design [Wilson et al., 1992] to identify the thermal hydraulic phenomena that determine the plant response under normal, off-normal and accident conditions. The phenomena were ranked in order of importance in satisfying safety, operational and investment

criteria. The study's objective was to provide a basis for evaluating the applicability of existing computer codes and to define computer code requirements for analysis of this reactor design. The RCCS was discussed as providing the key safety feature of passively removing decay heat during long term accident conditions which are within the Design Basis Accident (DBA) scope. This analysis did not include severe accidents but summarizes useful methods for severe accident considerations.

The CSAU methodology was employed to quantify uncertainties. In an earlier paper, Wulff et al. [1990] assessed the reactor safety margins and ranges of parameters in a gas-cooled reactor.

Selection of Event

Safety analysis for the PBMR must be performed under three sets of conditions: normal operation; DBA conditions; and severe accident conditions.

In evaluating reactor performance under normal conditions, important phenomena for the PBMR include:

- Mixing at core outlet, which defines the hottest region in the reactor
- Radiation heat transfer, which helps define the highest temperatures for fuel and surrounding materials

Five operating regimes are possible for Design Basis Accidents:

- Pressurized loss of forced circulation (P-LOFC)
- Depressurized loss of forced circulation (D-LOFC)
- D-LOFC with Air Ingress
- P-LOFC with Anticipated Transient Without Scram (ATWS)
- D-LOFC with ATWS

Important phenomena for a LOFC event include flow reversal and accompanying low heat removal rates at no-flow conditions.

Regarding severe accidents in the PBMR, a LOCA such as pressurized conduction cooldown is a key event because potentially harmful amounts of air and/or water may ingress into the core depending on assumptions. Important phenomena include:

- Conduction cooldown
- Complicated helium recirculation patterns
- Graphite oxidation
- Characterization of fission product transport through primary circuit

For the current project, a pressurized LOFC event in which the reactor has been scrammed has been selected. The MELCOR code with the PBMR input deck is capable of simulating decay heat generation and decay heat removal via the graphite reflectors.

Figure of Merit

Previous work [IAEA, 2001] has identified possible Figures of Merit for gas-cooled reactor analysis as:

- integrated heat loss from the reactor that exceeds the decay heat production under hypothetical accident conditions
- peak core temperature below the demonstrated fuel degradation temperature (1600°C)
- oxygen concentrations in high-temperature regions of the core (>1000°C) that are not sufficient to induce graphite oxidation
- reactor vessel temperatures

The Figure of Merit used in this analysis was the stored energy in the reactor core materials at the end of the transient. Typically, the peak fuel temperature (or analogously in LWRs the peak cladding temperature) is one of the most important metrics in determining the risk to the reactor in the case of a transient. However, for the MELCOR PBMR model that was used, the fuel temperatures were being over-predicted [Hogan, 2006]. Since this was the case, it was decided to use the total stored energy in the core materials as a surrogate for the temperature. If there is sufficient heat removal capability the stored energy in the core will begin to decrease. However, if the core energy continues to increase throughout the transient, then there is insufficient heat removal capability. Uncertainties in this analysis need to be resolved for any of the current analysis tools, rendering the pressurized LOFC analysis not only a good demonstration problem but also an analysis to be referenced by current researchers.

Uncertainty parameters:

Ranges of all parameters and their distributions are provided below.

Description of aleatory uncertainties:

Within the scope of this project, aleatory uncertainties are defined as those which are stochastic in nature or change as a function of plant conditions. Two aleatory uncertainties under consideration in this study were: the bypass flow (defined as the fraction of gas coolant that does not pass through the pebble-filled core volume) and the air velocity along the RPV wall outer surface.

Description of epistemic uncertainties:

Within the scope of this project, epistemic uncertainties are defined as those associated with code inputs. The epistemic uncertainties considered in this study were: the heat transfer coefficient between the fuel spheres and gas coolant, view factors for radiation heat transfer, friction factors within the core and core power after SCRAM.

Values, ranges and distributions of aleatory uncertainties:

Due to the high uncertainty in the process phenomena, all parameters were considered to be uniformly distributed in order to be conservative. The uncertainties given here are considered to be the upper and lower bounds of the distribution

1. Bypass flow
 - a. Value: nominal 10% of total core flow
 - b. Range: +/- 50% of bypass flow nominal value.
2. Air velocity along the RPV wall outer surface
 - a. Value: a first estimate of 0.1 m/s was adopted. Choosing a value that alternates between laminar and turbulent flow would allow for testing of the greatest uncertainty.
 - b. Range: +/- 30% of nominal air velocity.

Values, ranges and distributions of epistemic uncertainties:

1. Heat transfer coefficient between the fuel spheres and gas coolant
 - a. Value: reference values are already in the input deck (see Table 3).
 - b. Range: +/- 20% of the leading coefficient(s) and +/-20% of the powers
2. View factors for radiation heat transfer
 - a. Value: currently using default values.
 - b. Range: +/- 20% of nominal value. There is no physical basis for this value.
3. Friction factors within the core
 - a. Value: reference values are already in the input deck (see Table 3).
 - b. Range: +/- 10% of bypass flow nominal value.
4. Core power after scram

- a. Value: a decay heat curve is in the input deck.
- b. Range: A range of +/-10% was used where the greatest lack of knowledge is with respect to the decay heat curve presented in the literature.

Implementation of the values and their ranges into the MELCOR input deck is shown in Table 3.

MELCOR input model revisions and updates

The MELCOR model of the PBMR developed by Hogan [2006] has been revised and updated in order to both better represent the event selected for analysis, a pressurized loss of forced cooling event, as well as to more accurately represent the configuration of the 268MW PBMR. An updated nodalization diagram is provided in Figure 1. A calculation notebook for the modifications to the PBMR input is provided in Appendix A.

The active core region in the PBMR is defined in control volumes CV111-CV154. Axial and radial power profiles describe the distribution of core power, which is 268MWt at steady-state, in the COR cells within these control volumes. An upper void region is given as CV170, and the lower plenum is represented by CV100. A bypass flow region, CV300-CV305, represents the area of bypasses around the PBMR core. While the flow area and flow resistance of the bypass are known to have complicated parametric dependencies, the bypass is implemented as volumes with a flow area that restricts flow to 10% of the nominal inlet flow rate of the reactor, 129 kg/s [Reitsma, 2006], at steady-state operation. The aforementioned control volumes contain all of the volume that is referred to as the reactor core. Helium enters the system via CV180, which is a time-independent volume representing the inlet to the reactor pressure vessel, and CV200 represents the outlet of the pressure vessel as a time-independent volume.

During the pressurized LOFC event with reactor scram, the reactor power will be equivalent to the decay heat produced by the reactor, which is modeled using the following equations as a function of the initial power (P_0) [Terry, 2001]:

$$P = P_0 0.0603t^{-0.0639} \text{ for } t < 10s$$

$$P = P_0 0.0766t^{-0.1810} \text{ for } 10s \leq t < 150s$$

$$P = P_0 0.1300t^{-0.2830} \text{ for } 150 \leq t < 10^7 s$$

A decay heat model may be added to the PBMR deck following the methodology of Alkaabi [2007].

Circulation of the helium coolant will cease to be driven by compressors and will occur due to natural circulation. The system pressure will fall from 7.0 MPa to 4.5 MPa and the wall temperature will increase from 300°C to 350°C [IAEA, 2001].

The 330XX series of heat structures represents the core shroud, which is the radial boundary of the core region. Adjacent to the outer side of the shroud are a group of control volumes representing a layer of stagnant air, which is bounded by heat structures representing the core barrel. The core barrel is then bounded by another set of control volumes representing stagnant helium, and then the heat structures presenting the reactor pressure vessel. The final radial boundary of the model is a group of control volumes representing the reactor containment, CV500-CV504. Air flows in these control volumes by density-driven natural circulation as the air is heated.

Table 3 Uncertainty Parameter Values and Ranges

<u>Uncertainty</u>	<u>Card/SC Array</u>	<u>Word/SC</u>	<u>Nominal Value</u>	<u>Uncertainty (+/- %)</u>	<u>Minimum</u>	<u>Maximum</u>
Bypass Flow	FLnnn01	FLARA	12.9 kg/s	50	6.45 kg/s	19.35 kg/s
Air Velocity Along RPV Outer Wall	CVnnnA1	PVOL	101.325 kPa	10	91.1925 kpa	111.4575 kPa
Core Power After SCRAM	CF040	See Remark	268 MW	10	241.2 MW	294.8 MW
Fuel Sphere/Gas Heat Transfer Coefficient	SC1231	1	2	20	1.6	2.4
		2	0.6	20	0.48	0.72
	SC1232	1	2	20	1.6	2.4
		2	0.6	20	0.48	0.72
Radiation Heat Transfer View Factors	COR00003	FCELR/FCELA	0.25	20	0.2	0.3
Core Friction Factors	SC4413	1	1.75	10	1.575	1.925
		2	320	10	288	352
		3	20	10	18	22
		4	0.4	10	0.36	0.44

<u>Uncertainty</u>	<u>Remarks</u>
Bypass Flow	FLARA modified in CF300; Nominal = 10% of 129 kg/s
Air Velocity Along RPV Outer Wall	Change pressure in PVOL of CV220 by 10% to alter air velocity
Core Power After SCRAM	CF040 can change nominal value of 268 MW in input deck
Fuel Sphere/Gas Heat Transfer Coef.	SC1231 uses a correlation for forced convection Note that all constants are assumed linearly dependent SC1232 uses a correlation for free convection Note that all constants are assumed linearly dependent
Radiation Heat Transfer View Factors	MELCOR default values are used.
Core Friction Factors	Note that all constants are assumed linearly dependent

Figure 1 Revised MELCOR PBMR Model Nodalization

The input deck is executed by running the calculation to a steady-state, then initiating the pressurized loss of forced cooling event. Steady-state operation of the PBMR input deck is achieved by running the initial calculation for 160000 seconds. At this time, all values appear to be approximately constant, with the last parameters to reach a steady-state being the heat structures representing the core barrel and reactor pressure vessel. Following this steady-state initialization, a SCRAM is initiated and the power generated by the reactor core is that of decay heat, as prescribed by Terry [2001]. Simultaneous to the reactor SCRAM, the flow path areas of FL210 and FL200 are switched to zero, so helium is no longer entering or exiting the core region. As the core coolant temperature rises, the temperatures of HS330XX control volumes rise, which in turn radiate to HS440XX, which then radiate to HS550XX heat structures. Heat is finally dissipated via convection from the air in the containment volumes flowing over the HS550XX heat structures. The integrated heat loss from these heat structures by convection are used as the Figure of Merit in these calculations.

Task 4: Testing of the software on selected initiating events

The MELCOR model of the PBMR for steady state operation was tested under a variety of parametric conditions for uncertainty propagation and quantification (i.e. to observe the effect of the uncertainties on calculation results) As an example, the convective heat transfer coefficient from the fuel spheres to the coolant is a key parameter that was shown to affect the fuel temperatures and determine whether safety limits are violated.

Phase 2 Uncertainty Propagation and Quantification

The computational infrastructure software was completed at The Ohio State University and initial testing was performed at Texas A&M University. The MELCOR code was successfully run on TAMU's machines. The uncertainty propagation and quantification process consisted of the following tasks:

- Task 1: Linking MELCOR/DET to the Sampling tool
- Task 2: Testing of Task 1 software
- Task 3: Uncertainty quantification algorithm development
- Task 4: Incorporation of Task 3 software to MELCOR/DET
- Task 5: Testing of Task 4 software
- Task 6: User's manual documentation

Task 1: Linking MELCOR/DET to the Sampling tool

As originally proposed, a Monte Carlo/Latin Hypercube approach was to be used for epistemic uncertainty quantification. The DAKOTA [<http://www.cs.sandia.gov/DAKOTA/index.html>, 2009] software has the capability to generate both Monte Carlo and Latin Hypercube samples according to a variety of distributions. The DAKOTA software was leveraged for this project to generate Latin Hypercube samples for the purpose of epistemic uncertainty analysis.

Task 2: Testing of Task 1 software

In order to use the PBMR input deck with the Task 1 software, a MELCOR restart file is used to change values of the uncertainty parameters. This restart file includes input to modify the identified uncertainty parameters (see Phase 1, Task 3) including:

- Bypass flow percentage: control function multiplicative factor
- Heat transfer coefficient between the fuel spheres and gas coolant: sensitivity coefficients
- View factors for radiation heat transfer: COR00003 card on restart

- Friction factors within the core: sensitivity coefficients
- Core power after scram: control function multiplicative factor

Task 3: Uncertainty quantification algorithm development

As originally proposed, Phase 2 efforts included investigation of the numerical experiment designs other than Monte Carlo/Latin Hypercube Sampling. When considering the variations in a code-calculated output parameter (in this case, the stored energy in the core at the end of the transient) due to variations in code input parameters, two types of analyses must be considered: uncertainty analysis and sensitivity analysis. Uncertainty analysis is concerned with determining the distribution of the output value caused by variations in input, while the purpose of sensitivity analysis is to determine how strongly variations in each input parameter affect the output value.

The algorithm for uncertainty propagation and quantification was constructed in the form of two loops:

- An inner loop that generates a set of DETs for each set of uncorrelated uncertainty data to evaluate the correlated uncertainties through branching according to user-specified branching rules.
- On outer loop that runs through the uncorrelated uncertainty data.

Figure 2 illustrates this process graphically for an example, uniform and non-uniform probability distribution functions (pdfs) $p(x)$ on uncorrelated uncertainties. Each uncorrelated uncertainty branch corresponds to a contiguous inner loop run. The impact of uncorrelated uncertainties is quantified by sampling over the relevant MELCOR inputs as shown in Fig. 2. This two loop process allows the evaluation of correlated and uncorrelated uncertainties in a systematic and coherent fashion. Sensitivity analyses associated with the choice of pdfs for the sampling can be investigated without repeating the inner loop runs.

With regard to sensitivity analysis, Taguchi methods were utilized to assess the importance of each input parameter with respect to the output metric. Taguchi methods are statistical methods that have found widespread use in the process industry to improve the quality of manufactured goods. Taguchi methods allow for the extraction of a maximal amount of information about the main effects of parameters or the interaction effects results from a combination of parameters on a particular single output metric utilizing a minimal number of experiments. Taguchi methods utilize orthogonal array experiment designs, which are partial factorial experiment designs. Orthogonal experiment designs can offer a significant reduction in the number of trials required to determine the desired effects of a parameter or set of parameters. Compared to full factorial experiments, the experiment design size can be orders of magnitude smaller [Sharma, 2007], while yielding similar information regarding the impact of models inputs on the chosen output metric.

Task 4: Incorporation of Phase 2 Task 3 software to MELCOR/DET

The algorithm described under Phase 2, Task 3 was developed in the form of software compatible with ADAPT.

Task 5: Testing of Phase 2 Task 4 software

As is well known, it is often difficult to distinguish between epistemic and aleatory uncertainties. In order to illustrate the difference between treating a parameter's uncertainty vs. treating it as aleatory, two different uncertainty analyses were performed. First, a study was first performed considering the uncertainty in all of the chosen parameters as epistemic in nature (henceforth called the epistemic-only

analysis). Namely, for those parameters listed as aleatory (see Phase 1, Task 3), their uncertainties were considered in this case to represent uncertainties on input values and not considered to be a function of any process in the transient. Next, a second uncertainty analysis was performed in which the aleatory uncertainties were treated as branching conditions for an ADAPT dynamic event tree (henceforth called the combined analysis). While the aleatory uncertainties are modeled by the branching conditions themselves, multiple event trees were constructed with varying epistemic inputs to evaluate the impact of epistemic uncertainties. The scenario studied was that of a Loss of Forced Convection (LOFC) accident and the output metric considered was the stored energy in the core materials at the end of the simulation. Five of the six uncertain parameters were considered for this study. The sixth uncertain parameter, the air velocity along the outer RPV wall, was omitted due to an inability to modify the value through MELCOR input at the beginning of the transient.

Figure 2 Quantification of the epistemic uncertainties associated with simulator input parameter $a \leq x \leq b$ for two example pdfs (uniform, non-uniform) by fixed points x_n ($n=1, \dots, N$)

For the epistemic-only analysis, 250 Latin Hypercube simulations were performed to assess the uncertainty on the output metric due to uncertainty on the input. A plot of the cumulative distribution

function of the stored energy in the core materials which resulted from the Latin Hypercube simulations is shown in Figure 3.

For the combined analysis, the uncertainty on the core bypass flow was considered to be aleatory in nature (with the remaining parameters considered to be epistemic). The core bypass flow can be considered a randomly varying variable since in reality MELCOR is not capturing all of the turbulent effects which results in the region above the pebble-bed core. Instead of looking at a time-independent multiplier on the core bypass flow to represent uncertainty, it was considered that there can be fluctuations in the core bypass flow which are time dependent. To this end it was assumed that the bypass flow varied with a timescale of Δt . Namely, at every Δt , the system would branch with each child branch taking on a different core bypass flow value. This scenario was modeled with ADAPT using 3 points to represent the core bypass flow uncertainty distribution (the points correspond to probability values of 5%, 50%, and 95%). The time interval Δt was chosen to be 4000s (since this is for demonstration purposes, a Δt was chosen which did not result in prohibitive compute times).

In order to account for epistemic uncertainties in the combined analysis, multiple event trees were constructed with randomly varied inputs for the epistemic parameters. For this analysis, 50 Latin Hypercube samples were used for the “outer loop” (see Figure 2). The results for the combined analysis are summarized on Figure 4 along with the results from the epistemic-only analysis.

In Figure 4, the mean, median, 95th percentile and 5th percentile curves are all part of the combined analysis. For the combined analysis, multiple event trees were constructed and a cumulative distribution function (cdf) of the stored energy was developed for each. The shape of each cdf is influenced by the effects of the aleatory uncertainties, while the variations in the cdfs themselves represent the influence of the epistemic parameters. Viewing the uncertainties in this manner allows for a different interpretation of the results. When treating all parameters as epistemic, only the mean risk can be estimated. However, when aleatory parameters are considered, the shape of the risk curve is estimated by the treatment of aleatory uncertainties and confidence bounds are given by the treatment of epistemic uncertainties. The mean curve is the mean risk as calculated from the results of the combined analysis (includes the effects of epistemic and aleatory uncertainties). For comparison the mean risk curve as calculated by the epistemic analysis is also plotted (note that this is the same curve shown in Figure 3.)

For the purposes of the sensitivity analysis, three experiment designs were used. The results for each different experiment design were compared, with the prime focus being the comparison of Taguchi methods with more common sensitivity analysis approaches. For each experiment design, several methods were used to determine the importance ranking (with respect to the output metric) of the parameters listed in the earlier section. The three experiment designs are:

- 1) A full factorial experiment (3 levels for 5 parameters = 243 simulations)
- 2) A Latin Hypercube experiment with 250 samples (sample size chosen to be approximately the same size as the full factorial experiment)
- 3) A Taguchi orthogonal array experiment, with all 5 parameters at 3 levels. An L27(3⁵) orthogonal array was used which required 27 MELCOR simulations.

For each experiment design the parameters were ranked according to the following methods:

- 1) Linear Response Surface. The output data for each design was fit to a linear polynomial of the form:

$$Y = \sum_{i=1}^n a_i x_i \quad (1)$$

where Y is the output metric, the x_i 's uncertain parameters and the a_i 's are the fit coefficients. The importance of each parameter was determined by the absolute value of its associated fit coefficient.

Figure 3 Cumulative probability of core stored energy resulting from 250 Latin Hypercube simulations (epistemic-only analysis)

Figure 4 Results of the both the combined analysis and the epistemic only analysis

- 2) Linear correlation coefficient. A linear correlation coefficient was calculated for each uncertain parameter with respect to the output metric. The linear correlation coefficient has the form:

$$r_{x_i Y} = \frac{n \sum_{j=1}^n x_{ij} Y_j - \sum_{j=1}^n x_{ij} \sum_{j=1}^n Y_j}{\sqrt{n \sum_{j=1}^n x_{ij}^2 - (\sum_{j=1}^n x_{ij})^2} \sqrt{n \sum_{j=1}^n Y_j^2 - (\sum_{j=1}^n Y_j)^2}} \quad (2)$$

where $r_{x_i Y}$ is the linear correlation coefficient between the output Y and the uncertain parameter x_i and n is the number of trials in the experiment design. The value of r is between -1 and +1, with values close to +1 indicating strong positive linear correlation and values close to -1 indicating strong linear negative correlation. The importance of each parameter in this method was determined by the absolute value of r for each variable.

- 3) Partial correlation coefficients: Partial correlation coefficients (PCCs) are similar to linear correlation coefficients except that the partial correlation coefficient for a parameter subtracts out the linear effects of the remaining parameters. PCCs can be calculated by first determining the correlation matrix, C between all of the input variables, and between the input variables and the output metric. The PCC of an uncertain parameter x_i and the output metric Y is then determined by:

$$\rho_{x_i Y} = \frac{d_{i6}}{\sqrt{d_{ii} d_{66}}} \quad (3)$$

where d_{ij} are the elements of $D = C^{-1}$ (Note that C will be a 6 by 6 matrix, 5 rows and columns for the input parameters and a sixth column for the output metric). The value of ρ varies between -1 and +1. The importance of each parameter in this method was determined by the absolute value of ρ for each variable.

The above ranking methods were chosen based on their common use in literature and the fact that they are applicable to both randomized and factorial experiment designs. For the Taguchi orthogonal array design, in addition to computing parameter rankings based on the above methods, the Taguchi methodology has its own means of determining parameter ranking by examining the Signal-to-Noise (S/N) ratios of the output data. The advantage of the using the S/N ratio is its ability to account for uncertainty in the input data.

Table 4 shows a listing of the results of the parameter rankings utilizing each experiment design and test metric. The results shown are the relative influence demonstrated by each parameter (normalized to the parameter with the strongest influence).

Note that in this sensitivity study, all parameters were treated as epistemic in nature (i.e., as uncertainties in initial conditions). There is currently no standard technique for assessing importances of parameters in an environment of mixed epistemic and aleatory uncertainties.

The results given by the Taguchi method are consistent with the results given by all other methods. The first and second most important parameters are predicted as being the 5th parameter (core power after SCRAM) and the 2nd parameter (heat transfer coefficient between gas and fuel spheres). All methods give differing results as to which parameters are the 3rd through 5th most important parameters. Also of note is that the response surface, linear correlation coefficient and signal-to-noise ratio (Taguchi only) methods give similar relative rankings, namely that the 2nd through 5th most important parameters are at least one order of magnitude below the most important parameter. However, the partial correlation coefficient results (for the most part) give less separation between the most important parameter and the remainder. With regards to the partial correlation coefficient results, the Latin Hypercube and Full

Factorial designs give the exact same ranking, but the Taguchi method shows a different ordering for the 3rd through 5th most important parameters. However, the difference in relative importance between these parameters is small. The Taguchi method seems to apply more importance to the 1st parameter (bypass flow) in the partial correlation results than the other two experiment designs. Partial correlation coefficients can tend to give a more accurate picture of the effects of a parameter on an output value than the response surface or linear correlation coefficient methods can, since, in the partial correlation coefficient method, the linear effects of the other parameters are removed.

Table 4 Results of the Parameter Rankings

	Response Surface	Linear Correlation Coefficient	Partial Correlation Coefficient	Signal-to- Noise Ratio
Design/Parameter				
Latin Hypercube (250 runs)				
1	0.002 (5)	0.002 (5)	0.009 (5)	--
2	0.052 (2)	0.103 (4)	0.520 (2)	--
3	0.014 (3)	0.028 (3)	0.161 (4)	--
4	0.006 (4)	0.029 (2)	0.165 (3)	--
5	1.000 (1)	1.000 (1)	1.000 (1)	--
Full Factorial (243 runs)				
1	0.020 (4)	0.038 (4)	0.155 (5)	--
2	0.055 (2)	0.143 (2)	0.459 (2)	--
3	0.024 (3)	0.037 (5)	0.224 (4)	--
4	0.010 (5)	0.080 (3)	0.254 (3)	--
5	1.000 (1)	1.000 (1)	1.000 (1)	--
Taguchi (27 runs)				
1	0.034 (3)	0.034 (4)	0.232 (4)	0.067 (4)
2	0.039 (2)	0.079 (2)	0.480 (2)	0.079 (2)
3	0.025 (4)	0.051 (3)	0.333 (3)	0.078 (3)
4	0.006 (5)	0.032 (5)	0.215 (5)	0.032 (5)
5	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)

Task 6: User's manual documentation

A user's guide has been constructed for both the ADAPT server and ADAPT client software as well as an administrator's guide for installation of the ADAPT server software on a Linux cluster. The ADAPT user's and administrative guide are provided in Appendix B.

3. References

- Alkaabi, H., *Severe Accident Analysis Methods for Pebble Bed Modular Reactors*, M. S. Thesis, Purdue University, U.S. (2007).
- “The DAKOTA Project”, <http://www.cs.sandia.gov/DAKOTA/index.html>, viewed 9/8/2009.
- Hakobyan, A., T. Aldemir, R. Denning, B. Rutt, U. V. Catalyurek, S. Dunagan And D. Kunsman, "Dynamic Generation of Accident Progression Event Trees", *Nuclear Engineering and Design*, Vol. 238, pp. 3457-3467, (2006).
- Hogan, K. J., *Pebble Bed Modular Reactor Analysis with MELCOR*, M.S. thesis, Purdue University, (2006).
- International Atomic Energy Agency, *Current Status and Future Development of Modular High Temperature Gas Cooled Reactor Technology*, IAEA-TECDOC-1198, Austria, (2001).
- Narayanan, S., T. Kurc, U. Catalyurek, and J. Saltz, “Database support for data-driven scientific applications in the grid,” *Parallel Processing Letters*, vol. 13, no. 2, pp. 245–271 (2003).
- Reitsma, F., K. Ivanov, T. Downar, H. de Haas, H. D. Gougar, “The OECD/NEA/NSC PBRM Coupled Neutronics/Thermal Hydraulics Transient Benchmark: The PBMR-400 Core Design”, *PHYSOR-2006, ANS Topical Meeting on Reactor Physics*, Vancouver, BC, Canada. Paper D052 (2006).
- Rutt, B., U. V. Catalyurek, A. Hakobyan, K. Metzroth, T. Aldemir, R. Denning, S. Dunagan And D. Kunsman, "Distributed Dynamic Event Tree Generation for Reliability and Risk Assessment", *Proceedings of the International Workshop on Challenges of Large Applications in Distributed Environments (CLADE'06)*, 61-70, IEEE Computer Society, Los Alamitos, California (2006).
- Sharma, L., Aldemir, T., Parker, R., “Uncertainty Analysis Using Taguchi Methods with Virtual Experiments”, *Trans. Am. Nucl. Soc.*, 95, pp. 427 - 429 (2007).
- Terry, W., ed., Modular Pebble-Bed Reactor Project: Laboratory-Directed Research and Development Program FY 2001 Annual Report, INEEL/EXT-01-01623, INEEL (2001).
- Vilim, R. B., E. E. Feldman, W. D. Pointer, T. Y. C. Wei, “Initial VHTR Accident Scenario Classification: Models and Data”, *Nuclear Engineering Division*, Argonne National Laboratory, (2004).
- Wilson, G. E., D. C. Wadsworth, B. G. Miller, L. Lommers, P. Kroeger, M. Olszerski, R. Kochan, K. Liebelt, “Phenomena-based thermal-hydraulic modeling requirements for systems analysis of a Modular High Temperature Gas-Cooled Reactor”, *Nuclear Engineering and Design*, Vol. 136, pp. 319-333, (1992).
- Wulff, W., B. E. Boyack, I. Catton, R. B. Duffey, P. Griffith, K. R. Katsma, G. S. Lellouche, S. Levy, U. S. Rohatgi, G. E. Wilson, N. Zuber, “Quantifying Reactor Safety Margins Part 3: Assessment and Ranging of Parameters”, *Nuclear Engineering and Design*, 119, pp. 33-65 (1990).

4. Products Developed Under the Award and Technology Transfer Activities

4.1 Publications

- L. Sharma, T. Aldemir, R. Parker, “Uncertainty Analysis Using Taguchi Methods with Virtual Experiments”, *Trans. Am. Nucl. Soc.*, **95**, 427 - 429 (June 2007)
- L. Sharma, T. Aldemir, R. Parker, “Importance Ranking of Parameters Affecting Reactor Dynamics Using the Taguchi Method”, accepted for publication in *Nucl. Technol.* (April 7, 2009)

4.2 Networks or Collaborations Fostered

A new collaboration between the PI at Texas A&M University (TAMU) and the co-PI at The Ohio State University (OSU) has been formed as a result of this project. The opportunity to integrate the OSU PRA methodologies with the TAMU best estimate analysis is enabling technology advances. It is also providing the graduate students at both universities good exposure to other fields and experience in a collaborative environment.

APPENDIX A

The following is a review of recent additions made to the PBMR core cell input deck to model LOFC (loss of forced convection) severe accidents. Unless otherwise noted, design specifications are based on references [7] and [14]. The basis for the input of the model is reference [3]. If the value for a word or card is not written, the default value is assumed. Some default values are utilized when the input parameters are irrelevant.

Some assumptions for problem geometry should be enumerated:

1. The bypass (column CV's 30X) is constructed as a ring
2. The riser (column CV's 40X) is modeled as an annulus and not as cylindrical tubes

The 13 files included in the final PBMR input deck are:

mmgen.in	General MELGEN input; responsible for initializing calculations
mmcor.in	General MELCOR input; responsible for time-step advancement
core.gen	COR package input; core component input
core-cvh.gen	CVH package input; control volume and fluid information
core-fl.gen	FL package input; flow path parameters
core-hs.gen	HS package input; heat structure data
core-hs-sbsr	HS radiation heat transfer data for barrel and pressure vessel
core-pow-prof.gen	Core power information for COR package
ncg-mp.gen	Noncondensable gas and Materials package input
src-snk-ext.gen	CVH and CF package input for system source and sink
core-ext-cvh.gen	CVH input for bypass, riser, and containment/heat sink
core-ext-fl.gen	FL input for bypass, riser, and containment/heat sink
pbmr_edf.gen	EDF package input for external file writing/reading

The 10 particular files under examination are:

core-cvh.gen	CVH input for active core region
core-ext-cvh.gen	CVH, FL input for volumes outside the core (barrel, rpv, etc)
core-cv-sbsr.gen	CVH, CF input for stagnant He between reflector/barrel/rpv
core-ext-fl.gen	FL, CF input for inlet/outlet plenum and bypass
core-fl.gen	FL input for axial and radial core flowpaths
core-hs.gen	HS input for upper boundary and outer axial boundary
core-hs-sbsr.gen	HS input for barrel and rpv (structure-to-structure radiation)
src-snk-ext.gen	CVH, FL, CF package input for system source and sink
core-pow-prof.gen	COR input for axial/radial power density profiles and CF input
pbmr_edf.gen	EDF package input for external file writing/reading

A brief summary of the cylindrical heat structure surface area calculations follows the tenth file on page 43.

FILE.1 core-cvh.gen

Cavity, lower plenum, void region, and core control volume information:

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
CVTYPEEnn	CVTYPE	CAVITY= CV50 LWRPLNM= CV100 UPRVOID= CV170 CORCVH= others	User-defined CVTYPE names
CVnnn00	CVNAME	TABULATED VALUES IN TABLE A1,A2,A3	User-defined CV name
	ICVTHR		Nonequilibrium Thermodynamics switch
	ICVFF		CV flow flag; 0 = not defined
	ICVTYP		CV type number associated with CVTYPE names
CVnnn01	IPFSW		Pool and fog options; 0 = pool and fog allowed
	ICVACT		Active/inactive switch; 0 = active CV; -1= time-independent
CVnnnA0	ITYPTH		Thermodynamic input; 3 = separate pool/atmosphere records
CvnnnA1	MLFR.n		Specifies the mole fraction of NCG n (defined in NCG input)
	TATM		Specifies the temperature of the atmosphere of CV (Kelvin)
	PVOL		Specifies the CV pressure (Pascals)
	PH2O		Specifies the partial pressure of water vapor in atmosphere
CVnnnBk	_____		Altitude- Volume pairs; See <u>Table A2, Table A3</u>

Table A1 (CV's 50, 100, 170, 11X, 12X, 13X, 14X, 15X)

CV #	CVNAME	ICVTHR	ICVFF	ICVTYP	IPFSW	ICVACT
CV 50	CAVITY	2	0	4	0	0
CV 100	LOWER_PLENUM	2	0	2	0	0
CV 170	CORE_VOID_REG	2	0	3	0	0
CV 111	COR111	2	0	1	0	0
CV 112	COR112	2	0	1	0	0
CV 113	COR113	2	0	1	0	0
CV 114	COR114	2	0	1	0	0
CV 121	COR121	2	0	1	0	0
CV 122	COR122	2	0	1	0	0
CV 123	COR123	2	0	1	0	0
CV 124	COR124	2	0	1	0	0
CV 131	COR131	2	0	1	0	0
CV 132	COR132	2	0	1	0	0
CV 133	COR133	2	0	1	0	0
CV 134	COR134	2	0	1	0	0
CV 141	COR141	2	0	1	0	0
CV 142	COR142	2	0	1	0	0
CV 143	COR143	2	0	1	0	0
CV 144	COR144	2	0	1	0	0
CV 151	COR151	2	0	1	0	0
CV 152	COR152	2	0	1	0	0
CV 153	COR153	2	0	1	0	0
CV 154	COR154	2	0	1	0	0
CV #	ITYPTH	MLFR.4	TATM	PVOL	PH2O	
CV 50	3	1.0	773	6.825e6	0.0	
CV 100	3	1.0	773	6.825e6	0.0	
CV 170	3	1.0	773	7.0e6	0.0	
CV 1X1	3	1.0	773	6.860e6	0.0	
CV 1X2	3	1.0	773	6.895e6	0.0	
CV 1X3	3	1.0	773	6.930e6	0.0	
CV 1X4	3	10.	773	6.965e6	0.0	

*note that X varies from 1 to 5 over all core control volume numbers

Table A2 (CV's 50, 100, 170)

CV nnn	Altitude(m)	Volume(m^3)
CV 50	-1.0	0.0
	-0.5	4.810450
CV 100	-0.5508	0.0
	-0.3708	1.731762
	-0.2108	3.271106
	-0.0508	4.810450
	0.0	5.299317
CV 170	8.5	0.0
	8.75	2.4040625

Table A3 (CV 1(1,2,3,4,5)X)

CV nnn	Altitude(m)	Volume(m ³)
CV 111	0.0	0.0
	0.5	0.9704
	1.0	1.9408
	1.5	2.9119
	2.0	3.8823003
	2.5	4.8527
CV 112	2.5	0.0
	3.0	0.9704
	3.5	1.9408
	4.0	2.9112
	4.5	3.8816007
CV 113	4.5	0.0
	5.0	0.9704
	5.5	1.9408
	6.0	2.9112
	6.5	3.8816007
CV 114	6.5	0.0
	7.0	0.9704
	7.5	1.9408
	8.0	2.9112
	8.5	3.8816007
CV 121	0.0	0.0
	0.5	0.96145
	1.0	1.92290
	1.5	2.88435
	2.0	3.84580
	2.5	4.80725
CV 122	2.5	0.0
	3.0	0.96145
	3.5	1.92290
	4.0	2.88435
	4.5	3.84580
CV 123	4.5	0.0

	5.0	0.96145
	5.5	1.92290
	6.0	2.88435
	6.5	3.84580
CV 124	6.5	0.0
	7.0	0.96145
	7.5	1.92290
	8.0	2.88435
	8.5	3.84580
CV 131	0.0	0.0
	0.5	0.96060
	1.0	1.92120
	1.5	2.88180
	2.0	3.8424003
	2.5	4.80301
CV 132	2.5	0.0
	3.0	0.96060
	3.5	1.92120
	4.0	2.88180
	4.5	3.8424002
CV 133	4.5	0.0
	5.0	0.96060
	5.5	1.92120
	6.0	2.88180
	6.5	3.8424003
CV 134	6.5	0.0
	7.0	0.96060
	7.5	1.92120
	8.0	2.88180
	8.5	3.8424003
CV 141	0.0	0.0
	0.5	0.95960
	1.0	1.91920
	1.5	2.87880
	2.0	3.83840
	2.5	4.79800
CV 142	2.5	0.0
	3.0	0.95960

	3.5	1.91920
	4.0	2.87880
	4.5	3.83840
CV 143	4.5	0.0
	5.0	0.95960
	5.5	1.91920
	6.0	2.87880
	6.5	3.83840
CV 144	6.5	0.0
	7.0	0.95960
	7.5	1.91920
	8.0	2.87880
	8.5	3.83840
CV 151	0.0	0.0
	0.5	0.95840
	1.0	1.91680
	1.5	2.87520
	2.0	3.83360
	2.5	4.79200
CV 152	2.5	0.0
	3.0	0.95840
	3.5	1.91680
	4.0	2.87520
	4.5	3.83360
CV 153	4.5	0.0
	5.0	0.95840
	5.5	1.91680
	6.0	2.87520
	6.5	3.83360
CV 154	6.5	0.0
	7.0	0.95840
	7.5	1.91680
	8.0	2.87520
	8.5	3.83360

FILE.2 core-ext-cvh.gen

Containment, bypass, source, and sink control volume information:

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
CVTYPEEnn	CVTYPE	BYP CONT SOURCE SINK	User-defined CVTYPE names
CVnnn00	CVNAME	varies	User-defined CV name
	ICVTHR	2	Nonequilibrium Thermodynamics switch
	ICVFF	0	CV flow flag; 0 = not defined
	ICVTYP	1 for BYP/CONT 5 for SOURCE 6 for SINK	CV type number associated with CVTYPE names
CVnnn01	IPFSW	0	Pool and fog options; 0 = pool and fog allowed
	ICVACT	0 for BYP/CONT -1 for SOURCE/SINK	Active/inactive switch; 0 = active CV; -1= time-independent
CvnnnA0	ITYPTH	3	Thermodynamic input; 3 = separate pool/atmosphere records
CvnnnA1	MLFR.n	n=4 for BYP n=5 for CONT n=5 for SOURCE/SINK	Specifies the mole fraction of NCG n (defined in NCG input)
	TATM	773 for BYP 300 for CONT 300 for SOURCE/SINK	Specifies the temperature of the atmosphere of CV (Kelvin)
	PVOL	6.860e6 for BYP 101.325e3 for CONT 101.325e3 for SOURCE 101.0e3 for SINK	Specifies the CV pressure (Pascals)
	PH2O	0.0	Specifies the partial pressure of water vapor in atmosphere
CVnnnBk	_____	see BASIS column	Altitude- Volume pairs; See <u>table A4</u> , <u>table A5</u> , <u>table A6</u>

Table A4 (30X CV's)

CV nnn	Altitude(m)	Volume(m ³)
CV 300	-0.5508	0.0
	0.0	1.1667
CV 301	0.0	0.0
	2.5	5.2967
CV 302	2.5	0.0
	4.5	4.2374
CV 303	4.5	0.0
	6.5	4.2374
CV 304	6.5	0.0
	8.5	4.2374
CV 305	8.5	0.0
	8.75	0.5297

Table A5 (2X0 CV's)

CV nnn	Altitude(m)	Volume(m ³)
CV 220	-1.5508	0.0
	-0.5508	1.0
CV 230	8.5	0.0
	9.0	1.0

Table A6 (50X CV's)

CV nnn	Altitude(m)	Volume(m^3)
CV 500	-0.5508	0.0
	0.0	12.70105
CV 501	0.0	0.0
	2.5	57.64818
CV 502	2.5	0.0
	4.5	46.11854
CV 503	4.5	0.0
	6.5	46.11854
CV 504	6.5	0.0
	8.5	46.11854

FL for containment – FL 510 through 513

FL for source/sink – FL 220 and 230

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
FLnnn00	FLNAME	<u>TABULATED VALUES IN TABLE A7</u>	User-defined FL name
	KCVFM		“From” CV number
	KCVTO		“To” CV number
	ZFM		Altitude of “From” junction
	ZTO		Altitude of “To” junction
FLnnn01	FLARA		Flow path area
	FLEN		Flow path length
	FLOPO		Fraction of FP open
FLnnn02	KFLGFL		FP type; 3= normal horizontal FP
	KACTFL		FP flag; 0= active
	IBUBF		No bubble rise physics
	IBUBT		No bubble rise physics
FLnnnS1	SAREA		Segment flow area
	SLEN		Segment length
	SHYD		Segment hydraulic diameter

Table A7 (51X and 2X0 FL's)

FL #	FLNAME	KCVFM	KCVTO	ZFM(m)	ZTO(m)
FL 510	Cont500-501	500	501	0	0
FL 511	Cont501-502	501	502	2.5	2.5
FL 512	Cont502-503	502	503	4.5	4.5
FL 513	Cont503-504	503	504	6.5	6.5
FL 220	AirIn	220	500	-0.5508	-0.5508
FL 230	AirOut	504	230	8.5	8.5
FL #	FLARA(m^2)	FLEN(m)	FLOPO	KFLGFL	KACTFL
FL 510	23.05927	1.52	1.0	0	0
FL 511	23.05927	2.25	1.0	0	0
FL 512	23.05927	2.0	1.0	0	0
FL 513	23.05927	2.0	1.0	0	0
FL220	23.05927	1.52	1.0	0	0
FL230	23.05927	1.52	1.0	0	0
FL #	IBUBF	IBUBT	SAREA(m^2)	SLLEN(m)	SHYD(m)
FL 510	0	0	23.05927	1.52	2.0
FL 511	0	0	23.05927	2.25	2.0
FL 512	0	0	23.05927	2.0	2.0
FL 513	0	0	23.05927	2.0	2.0
FL 220	0	0	23.05927	1.52	2.0
FL 230	0	0	23.05927	1.52	2.0

FILE.3 core-cv-sbsr.gen

CV401 through 404 and 411 through 414 – Stagnant Helium between reflector/barrel and barrel/rpv:

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
CVTYPEnn	CVTYPE	TABULATED VALUES IN TABLES A8, A9	User-defined CVTYPE names
CVnnn00	CVNAME		User-defined CV name
	ICVTHR		Nonequilibrium Thermodynamics switch
	ICVFF		CV flow flag; 0 = not defined
	ICVTYP		CV type number associated with CVTYPE names
CVnnn01	IPFSW		Pool and fog options; 0 = pool and fog allowed
	ICVACT		Active/inactive switch; 0 = active CV; -1= time-independent
CvnnnA0	ITYPTH		Thermodynamic input; 3 = separate pool/atmosphere records
CvnnnA1	MLFR.n		Specifies the mole fraction of NCG n (defined in NCG input)
	TATM		Specifies the temperature of the atmosphere of CV (Kelvin)
	PVOL		Specifies the CV pressure (Pascals)
	PH2O		Specifies the partial pressure of water vapor in atmosphere
CVnnnBk	_____		Altitude- Volume pairs; See <u>table A9</u>

Table A8 (CV's 401-404, 411-414)

CV #	CVNAME	ICVTHR	ICVFF	ICVTYP	IPFSW	ICVACT
CV 401	R-B1	2	0	1	0	0
CV 402	R-B2	2	0	1	0	0
CV 403	R-B3	2	0	1	0	0
CV 404	R-B4	2	0	1	0	0
CV 411	R-B1	2	0	1	0	0
CV 412	R-B2	2	0	1	0	0
CV 413	R-B3	2	0	1	0	0
CV 414	R-B4	2	0	1	0	0
CV #	ITYPTH	MLFR.4	TATM	PVOL	PH2O	
CV 401 to CV414	3	1.0	773	6.80e6	0.0	

Table A9 (CV's 40X,41X)

CV nnn	Altitude(m)	Volume(m^3)
CV 401	0.0	0.0
	2.5	5.2967
CV 402	2.5	0.0
	4.5	4.2374
CV 403	4.5	0.0
	6.5	4.2374
CV 404	6.5	0.0
	8.5	4.2374
CV 411	0.0	0.0
	2.5	3.7196
CV 412	2.5	0.0
	4.5	2.9757
CV 413	4.5	0.0
	6.5	2.9757
CV 414	6.5	0.0
	8.5	2.9757

CF600 used to sum heat flux [W/m^2] values from HS5500X structures

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
CFn...n00	CFNAME	QFLUXADD	User-defined CF name
	CFTYPE	ADD	CF type; ADD gives sum of CF arguments
	NCFARG	4	Number of arguments
	CFSCAL	1.0	Multiplicative scale factor for CF
	CFADCN	0.0	Additive constant for CF
CFn...nKK	ARSCAL	1.0	Multiplicative scale factor
		1.0	
		1.0	
		1.0	
		1.0	
	ARADCN	0.0	Additive constant
		0.0	
		0.0	
		0.0	
		0.0	
	CHARG	HS-QFLUX-ATMS-R.55004	Database element identifier
		HS-QFLUX-ATMS-R.55003	Calls the HS package for QFLUX from RHS of HS5500X
		HS-QFLUX-ATMS-R.55002	
		HS-QFLUX-ATMS-R.55001	

FILE.4 core-ext-fl.gen

FL for inlet plenum to bypass – FL601

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
FL60100	FLNAME	In2By	User-defined FL name
	KCVFM	170	“From” CV number
	KCVTO	305	“To” CV number
	ZFM	8.625	Altitude of “From” junction
	ZTO	8.625	Altitude of “To” junction
FL60101	FLARA	3.581416	Flow path area
	FLEN	1.935	Flow path length
	FLOPO	1	Fraction of FP open
FL60102	KFLGFL	3	FP type; 3= normal horizontal FP
	KACTFL	0	FP flag; 0= active
	IBUBF	0	No bubble rise physics
	IBUBT	0	No bubble rise physics
FL601S1	SAREA	3.581416	Segment flow area
	SLEN	1.935	Segment length
	SHYD	0.5	Segment hydraulic diameter
FL601V1	NVTRIP	-1	Negative, then FLOPO defined by “on-forward” CF
	NVFONF	300	CF “on-forward” to describe FLOPO at “on-forward” state of trip time
	NVFONR	300	CF “on-reverse” to describe FLOPO at “on-reverse” state of trip time

FL for bypass to outlet plenum – FL 600

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
FL60000	FLNAME	By2Out	User-defined FL name
	KCVFM	300	“From” CV number
	KCVTO	100	“To” CV number
	ZFM	-0.2754	Altitude of “From” junction
	ZTO	-0.2754	Altitude of “To” junction
FL60001	FLARA	7.890575	Flow path area
	FLEN	1.935	Flow path length
	FLOPO	1	Fraction of FP open
FL60002	KFLGFL	3	FP type; 3= normal horizontal FP
	KACTFL	0	FP flag; 0= active
	IBUBF	0	No bubble rise physics
	IBUBT	0	No bubble rise physics
FL600S1	SAREA	7.890575	Segment flow area
	SLEN	1.935	Segment length
	SHYD	1.1016	Segment hydraulic diameter
FL601V1	NVTRIP	-1	Negative, then FLOPO defined by “on-forward” CF
	NVFONF	300	CF “on-forward” to describe FLOPO at “on-forward” state of trip time
	NVFONR	300	CF “on-reverse” to describe FLOPO at “on-reverse” state of trip time

FL for bypass – FL31X

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
FLnnn00	FLNAME	<u>TABULATED VALUES IN TABLE A10</u>	User-defined FL name
	KCVFM		“From” CV number
	KCVTO		“To” CV number
	ZFM		Altitude of “From” junction
	ZTO		Altitude of “To” junction
FLnnn01	FLARA		Flow path area
	FLEN		Flow path length
	FLOPO		Fraction of FP open
FLnnn02	KFLGFL		FP type; 0= normal vertical FP
	KACTFL		FP flag; 0= active
	IBUBF		No bubble rise physics
	IBUBT		No bubble rise physics
FLnnnS1	SAREA		Segment flow area
	SLEN		Segment length
	SHYD		Segment hydraulic diameter

Table A10 (31X FP's)

FL #	FLNAME	KCVFM	KCVTO	ZFM(m)	ZTO(m)
FL 310	Bypass301-300	301	300	0	0
FL 311	Bypass302-301	302	301	2.5	2.5
FL 312	Bypass303-302	303	302	4.5	4.5
FL 313	Bypass304-303	304	303	6.5	6.5
FL 314	Bypass305-304	305	304	8.5	8.5
FP #	FLARA(m ²)	FLLEN(m)	FLOPO	KFLGFL	KACTFL
FL 310	2.118688	1.52	0.5	0	0
FL 311	2.118688	2.25	0.5	0	0
FL 312	2.118688	2	0.5	0	0
FL 313	2.118688	2	0.5	0	0
FL 314	2.118688	1.125	0.5	0	0
FP #	IBUBF	IBUBT	SAREA(m ²)	SLLEN(m)	SHYD(m)
FL 310	0	0	2.118688	1.52	0.24
FL 311	0	0	2.118688	2.25	0.24
FL 312	0	0	2.118688	2	0.24
FL 313	0	0	2.118688	2	0.24
FL 314	0	0	2.118688	1.125	0.24

CF300 used to define flow path blockage at trip time

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
CFn...n00	CFNAME	FRACOP	User-defined CF name
	CFTYPE	EQUALS	CF type; EQUALS sets the given parameter equal to CF argument
	NCFARG	1	Number of arguments
	CFSCAL	1.0	Multiplicative scale factor for CF
	CFADCN	0	Additive constant for CF
CFn...nKK	ARSCAL	0.0	Multiplicative scale factor
	ARADCN	0.010	Additive constant
	CHARG	TIME	Database element identifier

FILE.5 core-fl.gen

Core flow paths (11X. 12X. 13X. 14X. 15X) – axial

Core flow paths (1X1. 1X2, 1X3, 1X4) – radial

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
FLnnn00	FLNAME	TABULATED VALUES IN TABLE A11, A12	User-defined FL name
	KCVFM		“From” CV number
	KCVTO		“To” CV number
	ZFM		Altitude of “From” junction
	ZTO		Altitude of “To” junction
FLnnn01	FLARA		Flow path area
	FLEN		Flow path length
	FLOPO		Fraction of FP open
FLnnn02	KFLGFL		FP type; 0= normal vertical FP
	KACTFL		FP flag; 0= active
	IBUBF		No bubble rise physics
	IBUBT		No bubble rise physics
FLnnnS1	SAREA		Segment flow area
	SLEN		Segment length
	SHYD		Segment hydraulic diameter

Table A11 (FL's 11X, 12X, 13X, 14X, 15X) – axial

FL #	FLNAME	KCVFM	KCVTO	ZFM(m)	ZTO(m)
FL 110	CoreRingF11-T00	111	100	0.0	0.0
FL 111	CoreRingF12-T11	112	111	2.5	2.5
FL 112	CoreRingF13-T12	113	112	4.5	4.5
FL 113	CoreRingF14-T13	114	113	6.5	6.5
FL 114	CoreRingF60-T14	170	114	8.5	8.5
FL 120	CoreRingF21-T00	121	100	0.0	0.0
FL 121	CoreRingF22-T21	122	121	2.5	2.5
FL 122	CoreRingF23-T22	123	122	4.5	4.5
FL 123	CoreRingF24-T23	124	123	6.5	6.5
FL 124	CoreRingF60-T24	170	124	8.5	8.5
FL 130	CoreRingF31-T00	131	100	0.0	0.0
FL 131	CoreRingF32-T31	132	131	2.5	2.5
FL 132	CoreRingF33-T32	133	132	4.5	4.5
FL 133	CoreRingF34-T33	134	133	6.5	6.5
FL 134	CoreRingF60-T34	170	134	8.5	8.5
FL 140	CoreRingF41-T00	170	144	0.0	0.0
FL 141	CoreRingF42-T41	142	141	2.5	2.5
FL 142	CoreRingF43-T42	143	142	4.5	4.5
FL 143	CoreRingF44-T43	144	143	6.5	6.5
FL 144	CoreRingF60-T44	170	144	8.5	8.5
FL 150	CoreRingF51-T00	151	100	0.0	0.0
FL 151	CoreRingF52-T51	152	151	2.5	2.5
FL 152	CoreRingF53-T52	153	152	4.5	4.5
FL 153	CoreRingF54-T53	154	153	6.5	6.5
FL 154	CoreRingF60-T54	170	154	8.5	8.5
FL #	FLARA(m^2)	FLLEN(m) all 1XX	FLOPO	KFLGFL	KACTFL
FL 110-114	1.9408 all 11X	1.25 (e.g. 110)	1.0	0	0
FL 120-124	1.9229 all 12X	2.25 (e.g. 111)			
FL 130-134	1.9212 all 13X	2.00 (e.g. 112)			
FL 140-144	1.9192 all 14X	2.00 (e.g. 113)			
FL 150-154	1.9168 all 15X	1.125 (e.g. 114)			
FL #	IBUBF	IBUBT	SAREA(m^2)	SLEN(m)	SHYD(m)
FL 110-114	0	0	1.9408 all 11X	1.25 (e.g. 110)	FL114=0.05; others=1.572
FL 120-124			1.9229 all 12X	2.25 (e.g. 111)	FL124=0.05; others=0.646
FL 130-134			1.9212 all 13X	2.00 (e.g. 112)	FL134=0.05; others=0.496
FL 140-144			1.9192 all 14X	2.00 (e.g. 113)	FL144=0.05; others=0.418
FL 150-154			1.9168 all 15X	1.125 (e.g. 114)	FL154=0.05; others=0.368

Table A12 (FL's 1X1, 1X2, 1X3, 1X4) – radial

FL #	FLNAME	KCVFM	KCVTO	ZFM(m)	ZTO(m)
FL 161	CoreRingF21-T11	121	111	1.25	1.25
FL 162	CoreRingF22-T12	122	112	3.50	3.50
FL 163	CoreRingF23-T13	123	113	5.50	5.50
FL 164	CoreRingF24-T14	124	114	7.50	7.50
FL 171	CoreRingF31-T21	131	121	1.25	1.25
FL 172	CoreRingF32-T22	132	122	3.50	3.50
FL 173	CoreRingF33-T23	133	123	5.50	5.50
FL 174	CoreRingF34-T24	134	124	7.50	7.50
FL 181	CoreRingF41-T31	141	131	1.25	1.25
FL 182	CoreRingF42-T32	142	132	3.50	3.50
FL 183	CoreRingF43-T33	143	133	5.50	5.50
FL 184	CoreRingF44-T34	144	134	7.50	7.50
FL 191	CoreRingF51-T41	151	141	1.25	1.25
FL 192	CoreRingF52-T42	152	142	3.50	3.50
FL 193	CoreRingF53-T43	153	143	5.50	5.50
FL 194	CoreRingF54-T44	154	144	7.50	7.50
FL #	FLARA(m^2)	FLLEN(m) all 1XX	FLOPO	KFLGFL	KACTFL
FL 161	12.34646	0.5545	1.0	3	0
FL 162	9.877168	0.5545	1.0	3	0
FL 163	9.877168	0.5545	1.0	3	0
FL 164	9.877168	0.5545	1.0	3	0
FL 171	17.42013	0.2855	1.0	3	0
FL 172	13.93610	0.2855	1.0	3	0
FL 173	13.93610	0.2855	1.0	3	0
FL 174	13.93610	0.2855	1.0	3	0
FL 181	21.31570	0.2285	1.0	3	0
FL 182	17.052564	0.2285	1.0	3	0
FL 183	17.052564	0.2285	1.0	3	0
FL 184	17.052564	0.2285	1.0	3	0
FL 191	24.59867	0.1965	1.0	3	0
FL 192	19.678936	0.1965	1.0	3	0
FL 193	19.678936	0.1965	1.0	3	0
FL 194	19.678936	0.1965	1.0	3	0

FL #	IBUBF	IBUBT	SAREA(m^2)	SLEN(m)	SHYD(m)
FL 161	0	0	12.34646	0.5545	5.0
FL 162	0	0	9.877168	0.5545	5.0
FL 163	0	0	9.877168	0.5545	5.0
FL 164	0	0	9.877168	0.5545	5.0
FL 171	0	0	17.42013	0.2855	4.0
FL 172	0	0	13.93610	0.2855	4.0
FL 173	0	0	13.93610	0.2855	4.0
FL 174	0	0	13.93610	0.2855	4.0
FL 181	0	0	21.31570	0.2285	4.0
FL 182	0	0	17.052564	0.2285	4.0
FL 183	0	0	17.052564	0.2285	4.0
FL 184	0	0	17.052564	0.2285	4.0
FL 191	0	0	24.59867	0.1965	4.0
FL 192	0	0	19.678936	0.1965	4.0
FL 193	0	0	19.678936	0.1965	4.0
FL 194	0	0	19.678936	0.1965	4.0

FL blockage model implementation

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
FLnnnBk	PKG	TABULATED VALUES IN TABLE A13,A14	Must be set to COR
	OPTION		Flow geometry to be modeled
	ICORC1		One of two corner core cells defining limiting ring/axial levels
	ICORC2		Second of two corner cells defining limiting ring/axial levels
	FLMPTY		Form loss coeff. For “empty” geometry. Add to Ergun correlation results

Table A13 (axial blockage)

FLnnnBk	PKG	OPTION	ICORC1	ICORC2	FLMPTY
FL111B1	COR	AXIAL	111	108	1.0
FL112B1	COR	AXIAL	115	112	1.0
FL113B1	COR	AXIAL	119	116	1.0
FL121B1	COR	AXIAL	211	208	1.0
FL122B1	COR	AXIAL	215	212	1.0
FL123B1	COR	AXIAL	219	216	1.0
FL131B1	COR	AXIAL	311	308	1.0
FL132B1	COR	AXIAL	315	312	1.0
FL133B1	COR	AXIAL	319	316	1.0
FL141B1	COR	AXIAL	411	408	1.0
FL142B1	COR	AXIAL	415	412	1.0
FL143B1	COR	AXIAL	419	416	1.0
FL151B1	COR	AXIAL	511	508	1.0
FL152B1	COR	AXIAL	515	512	1.0
FL153B1	COR	AXIAL	519	516	1.0

Table A14 (radial blockage)

FLnnnBk	PKG	OPTION	ICORC1	ICORC2	FLMPTY
FL161B1	COR	RADIAL	105	209	1.0
FL162B1	COR	RADIAL	110	213	1.0
FL163B1	COR	RADIAL	114	217	1.0
FL164B1	COR	RADIAL	118	221	1.0
FL171B1	COR	RADIAL	205	309	1.0
FL172B1	COR	RADIAL	210	313	1.0
FL173B1	COR	RADIAL	214	317	1.0
FL174B1	COR	RADIAL	218	321	1.0
FL181B1	COR	RADIAL	305	409	1.0
FL182B1	COR	RADIAL	310	413	1.0
FL183B1	COR	RADIAL	314	417	1.0
FL184B1	COR	RADIAL	318	421	1.0
FL191B1	COR	RADIAL	405	509	1.0
FL192B1	COR	RADIAL	410	513	1.0
FL193B1	COR	RADIAL	414	517	1.0
FL194B1	COR	RADIAL	418	521	1.0

FILE.6 core-hs.gen

HS320XX and HS330XX structures:

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
HSCCCCC000	NP		Number of temp. nodes
	IGEOM		HS geometry; 2= cylindrical
	ISS		Steady-State initialization
HSCCCCC001	HSNAME		User-specified HS name
HSCCCCC002	HSALT		Elevation parameter
	ALPHA		Orientation parameter; 1= vertical
HSCCCCC004	IOPTL		Option for left inside HS surface; 0=using bulk fluid temp.
	IOPTR		Option for right inside HS surface; 0=using bulk fluid temp.
HSCCCCC100	NODLOC		Location of temp. node data
	IFRMAT		Temp. node location data format flag
	XI		Left inside boundary location of node
HSCCCCC1NN	XVALUE		In this case, length of mesh interval
	NXVALU		Number of mesh intervals with XVALUE length
HSCCCCC200	MCDLOC		Location of material composition data
HSCCCCC201	MATNAM		Name of material
	MSHNUM		Mesh interval
HSCCCCC300	ISRC		Internal power source type; 0= no internal power
HSCCCCC400	IBCL		Boundary condition type; 1= convective boundary
	IBVL		Identifies leftmost CV associated with this HS
	IFLOWL		Type of flow over left inside boundary
	CPFPL		Critical pool fraction – pool
	CPFAL		Critical pool fraction – atmosphere
	EMISWL		Wall emissivity of left surface; 0=No radiation heat transfer
	RMODL		Employed radiation model; equiv-band OR gray-gas-a
HSCCCCC401	PATHL		Radiation path length of left surface
	ASURFL		Boundary surface area
	CLNL		Boundary surface characteristic length (axial length)
HSCCCCC500	BNDZL		Axial length (same as CLNL for cylinders)
	IBCR		Boundary condition type; 1= convective boundary
	IBVR		Identifies rightmost CV associated with this HS
HSCCCCC600	IFLOWR		Type of flow over right inside boundary
	CPFPR		Critical pool fraction – pool
	CPFAR		Critical pool fraction – atmosphere
HSCCCCC700	ASURFR		Boundary surface area (right)
	CLNR		Characteristic length; axial length (right)
	BNDZR		Axial length (same as CLNR for cylinders)

TABULATED VALUES IN TABLE A15

Table A15 (HS's 320XX, 330XX)

HSnnnnn	NP	IGEOM	ISS	HSNAME	HSALT(m)	ALPHA	IOPTL
HS32001	2	1	0	UPPER_BOUNDARY_RING1	8.50	0.0	-
HS32002	2	1	0	UPPER_BOUNDARY_RING2	8.50	0.0	-
HS32003	2	1	0	UPPER_BOUNDARY_RING3	8.50	0.0	-
HS32004	2	1	0	UPPER_BOUNDARY_RING4	8.50	0.0	-
HS32005	2	1	0	UPPER_BOUNDARY_RING5	8.50	0.0	-
HS33001	2	2	0	OUTER_HS_LVL_1	-0.5508	1.0	501
HS33002	2	2	0	OUTER_HS_LVL_2	-0.3708	1.0	502
HS33003	2	2	0	OUTER_HS_LVL_3	-0.2108	1.0	503
HS33004	2	2	0	OUTER_HS_LVL_4	-0.0508	1.0	504
HS33005	2	2	0	OUTER_HS_LVL_5	0.0	1.0	505
HS33006	3	2	0	OUTER_HS_LVL_6	0.5	1.0	506
HS33007	3	2	0	OUTER_HS_LVL_7	1.0	1.0	507
HS33008	3	2	0	OUTER_HS_LVL_8	1.5	1.0	508
HS33009	3	2	0	OUTER_HS_LVL_9	2.0	1.0	509
HS33010	3	2	0	OUTER_HS_LVL_10	2.5	1.0	510
HS33011	3	2	0	OUTER_HS_LVL_11	3.0	1.0	511
HS33012	3	2	0	OUTER_HS_LVL_12	3.5	1.0	512
HS33013	3	2	0	OUTER_HS_LVL_13	4.0	1.0	513
HS33014	3	2	0	OUTER_HS_LVL_14	4.5	1.0	514
HS33015	3	2	0	OUTER_HS_LVL_15	5.0	1.0	515
HS33016	3	2	0	OUTER_HS_LVL_16	5.5	1.0	516
HS33017	3	2	0	OUTER_HS_LVL_17	6.0	1.0	517
HS33018	3	2	0	OUTER_HS_LVL_18	6.5	1.0	518
HS33019	3	2	0	OUTER_HS_LVL_19	7.0	1.0	519
HS33020	3	2	0	OUTER_HS_LVL_20	7.5	1.0	520
HS33021	3	2	0	OUTER_HS_LVL_21	8.0	1.0	521
HSnnnnn	IOPTR	NODLOC(m)	IFRMAT	XI(m)	XVALUE(m)	NXVALU	MATNAM
HS32001	-	-1	1	0.0	0.1	2	STAINLESS STEEL 304'
HS32002	-	-1	1	0.0	0.1	2	STAINLESS STEEL 304'
HS32003	-	-1	1	0.0	0.1	2	STAINLESS STEEL 304'
HS32004	-	-1	1	0.0	0.1	2	STAINLESS STEEL 304'
HS32005	-	-1	1	0.0	0.1	2	STAINLESS STEEL 304'
HS33001 To HS33021	-	-1	2	1.75	0.75	1	-
					0.25	1	

HSnnnnn	MSHNUM	ISRC	IBCL	IBVL	IFLOWL	CPFPL(m)	CPFAL(m)
HS32001	1	0	1	114	EXT	0.9	0.9
HS32002	1	0	1	124	EXT	0.9	0.9
HS32003	1	0	1	134	EXT	0.9	0.9
HS32004	1	0	1	144	EXT	0.9	0.9
HS32005	1	0	1	154	EXT	0.9	0.9
HS33001 To HS33004	-	0	1	100	EXT	0.9	0.9
HS33005 To HS33009	-	0	1	151	EXT	0.9	0.9
HS33010 To HS33013	-	0	1	152	EXT	0.9	0.9
HS33014 To HS33017	-	0	1	153	EXT	0.9	0.9
HS33018 To HS33021	-	0	1	154	EXT	0.9	0.9
HSnnnnn	EMISWL	RMODL	PATHL	ASURFL(m^2)	CLNL(m)	BNDZL(m)	IBCR
HS32001	0.8	equiv-band	0.25	1.9408	0.786	0.786	0
HS32002	0.8	equiv-band	0.25	1.9229	0.646	0.646	0
HS32003	0.8	equiv-band	0.25	1.9212	0.496	0.496	0
HS32004	0.8	equiv-band	0.25	1.9212	0.418	0.418	0
HS32005	0.8	equiv-band	0.25	1.9168	0.368	0.368	0
HS33001	-	-	-	1.97	3.50	0.18	0
HS33002	-	-	-	1.76	3.50	0.16	0
HS33003	-	-	-	1.76	3.50	0.16	0
HS33004	-	-	-	0.5586	3.50	0.0508	0
HS33005 To HS33021	-	-	-	5.50	3.50	0.5	1
HSnnnnn	IBVR	IFLOWR	CPFPR(m)	CPFAR(m)	ASURFR(m^2)	CLNR(m)	BNDZR(m)
HS32001 To HS32005	-1	-	-	-	-	-	-
HS33001 To HS33004	-	-	-	-	-	-	-
HS33005 To HS33009	401	EXT	0.9	0.9	5.50	3.50	0.5
HS33010 To HS33013	402	EXT	0.9	0.9	5.50	3.50	0.5
HS33014 To HS33017	403	EXT	0.9	0.9	5.50	3.50	0.5
HS33018 To HS33021	404	EXT	0.9	0.9	5.50	3.50	0.5

FILE.7 core-hs-sbsr.gen

Barrel and pressure vessel heat structures (HS440XX, 550XX) and radiation input:

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
HSCCCCC000	NP		Number of temp. nodes
	IGEOM		HS geometry; 2= cylindrical
	ISS		Steady-State initialization
HSCCCCC001	HSNAME		User-specified HS name
HSCCCCC002	HSALT		Elevation parameter
	ALPHA		Orientation parameter; 1= vertical
HSCCCCC100	NODLOC		Location of temp. node data
	IFRMAT		Temp. node location data format flag
	XI		Left inside boundary location of node
HSCCCCC1NN	XVALUE		In this case, length of mesh interval
	NXVALU		Number of mesh intervals with XVALUE length
HSCCCCC201	MATNAM		Name of material
	MSHNUM		Mesh interval
HSCCCCC300	ISRC		Internal power source type; 0= no internal power
HSCCCCC400	IBCL		Boundary condition type; 1= convective boundary
	IBVL		Identifies leftmost CV associated with this HS
	IFLOWL		Type of flow over left inside boundary
	CPFPL		Critical pool fraction – pool
	CPFAL		Critical pool fraction – atmosphere
	ASURFL		Boundary surface area
	CLNL		Boundary surface characteristic length (axial length)
HSCCCCC500	BNDZL		Axial length (same as CLNL for cylinders)
	IBCR		Boundary condition type; 1= convective boundary
	IBVR		Identifies rightmost CV associated with this HS
	IFLOWR		Type of flow over right inside boundary
	CPFPR		Critical pool fraction – pool
	CPFAR		Critical pool fraction – atmosphere
	ASURFR		Boundary surface area (right)
HSCCCCC700	CLNR		Characteristic length; axial length (right)
	BNDZR		Axial length (same as CLNR for cylinders)

TABULATED VALUES IN TABLE A16

Table A16 (HS's 440XX, 550XX)

HSnnnnn	NP	IGEOM	ISS	HSNAME	HSALT(m)	ALPHA	NODLOC
HS44001	2	2	0	OUTER_HS_LVL_1	0.0	1.0	-1
HS44002	2	2	0	OUTER_HS_LVL_2	2.5	1.0	-1
HS44003	2	2	0	OUTER_HS_LVL_3	4.5	1.0	-1
HS44004	2	2	0	OUTER_HS_LVL_4	6.5	1.0	-1
HS55001	2	2	0	OUTER_HS_LVL_1	0.0	1.0	-1
HS55002	2	2	0	OUTER_HS_LVL_2	2.5	1.0	-1
HS55003	2	2	0	OUTER_HS_LVL_3	4.5	1.0	-1
HS55004	2	2	0	OUTER_HS_LVL_4	6.5	1.0	-1
HSnnnnn	IFRMAT	XI(m)	XVALUE(m)	NXVALU	MATNAM	MSHNUM	ISRC
HS44001	2	2.75	0.05	1	STAINLESS STEEL 304'	1	0
HS44002	2	2.75	0.05	1	STAINLESS STEEL 304'	1	0
HS44003	2	2.75	0.05	1	STAINLESS STEEL 304'	1	0
HS44004	2	2.75	0.05	1	STAINLESS STEEL 304'	1	0
HS55001	2	3.00	0.17	1	STAINLESS STEEL 304'	1	0
HS55002	2	3.00	0.17	1	STAINLESS STEEL 304'	1	0
HS55003	2	3.00	0.17	1	STAINLESS STEEL 304'	1	0
HS55004	2	3.00	0.17	1	STAINLESS STEEL 304'	1	0
HSnnnnn	IBCL	IBVL	IFLOWL	CPFPL(m)	CPFAL(m)	ASURFL(m^2)	CLNL(m)
HS44001 to HS44004	1	44001=401 44002=402 44003=403 44004=404	EXT	0.9	0.9	44001=1.97 44002=1.76 44003=1.76 44004=0.5586	5.5
HS55001 to HS55004	1	55001=411 55002=412 55003=413 55004=414	EXT	0.9	0.9	55001=1.97 55002=1.76 55003=1.76 55004=0.5586	6.0
HSnnnnn	BNDZL(m)	IBCR	IBVR	IFLOWR	CPFPR(m)	CPFAR(m)	ASURFR(m^2)
HS44001 to HS44004	44001=2.5 44002=2.0 44003=2.0 44004=2.0	1	44001=411 44002=412 44003=413 44004=414	EXT	0.9	0.9	44001=1.97 44002=1.76 44003=1.76 44004=0.5586
HS55001 to HS55004	55001=2.5 55002=2.0 55003=2.0 55004=2.0	1	55001=501 55002=502 55003=503 55004=504	EXT	0.9	0.9	55001=1.97 55002=1.76 55003=1.76 55004=0.5586
HSnnnnn	CLNR(m)	BNDZR(m)					
HS44001 to HS44004	5.5	44001=2.5 44002=2.0 44003=2.0 44004=2.0					
HS55001 to HS55004	6.0	55001=2.5 55002=2.0 55003=2.0 55004=2.0					

Structure-to-structure radiation heat transfer data:

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
HSRDCCCCC0	IHSRD1	$\frac{TABLE}{A17}$	Number of first surface of pair; (LHS)/(RHS)=(-HS#)/(+HS#)
	IHSRD2		Number of other surface; same LHS/RHS convention
	VIEW		View factor between surfaces

TABLE A17 (radiation data)

HSRDCCCCC0	IHSRD1	IHSRD2	VIEW
HSRD000050	33005	44001	1.0
HSRD000060	33006	44002	1.0
HSRD000070	33007	44002	1.0
HSRD000080	33008	44002	1.0
HSRD000090	33009	44002	1.0
HSRD000100	33010	44003	1.0
HSRD000110	33011	44003	1.0
HSRD000120	33012	44003	1.0
HSRD000130	33013	44003	1.0
HSRD000140	33014	44003	1.0
HSRD000150	33015	44003	1.0
HSRD000160	33016	44003	1.0
HSRD000170	33017	44003	1.0
HSRD000180	33018	44004	1.0
HSRD000190	33019	44004	1.0
HSRD000200	33020	44004	1.0
HSRD000210	33021	44004	1.0
HSRD000310	44001	55001	1.0
HSRD000320	44002	55002	1.0
HSRD000330	44003	55003	1.0
HSRD000340	44004	55004	1.0

FILE.8 src-snk-ext.gen

CV 160 and CV200 – Time independent source and sink:

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
CVTYPEnn	CVTYPE	TABULATED VALUES IN TABLE A18, A19	User-defined CVTYPE names
CVnnn00	CVNAME		User-defined CV name
	ICVTHR		Nonequilibrium Thermodynamics switch
	ICVFF		CV flow flag
	ICVTYP		CV type number associated with CVTYPE names
CVnnn01	IPFSW		Pool and fog options
	ICVACT		Active/inactive switch
CvnnnA0	ITYPTH		Thermodynamic input
CvnnnA1	MLFR.n		Specifies the mole fraction of NCG n (defined in NCG input)
	TATM		Specifies the temperature of the atmosphere of CV (Kelvin)
	PVOL		Specifies the CV pressure (Pascals)
	PH2O		Specifies the partial pressure of water vapor in atmosphere
CVnnnBk	_____		Altitude-Hydrodynamic volume pairs

Table A18 (CV's 200, 160)

CV nnn	CVTYPE	CVNAME	ICVTHR	ICVFF	ICVTYP
CV 160	SRC	SOURCE	2	0	5
CV 200	SNK	SINK	2	0	6
CV nnn	IPFSW	ICVACT	ITYPTH	MLFR.4	TATM(K)
CV 160	0	-1	3	1.0	773
CV 200	0	-1	3	1.0	773
CV nnn	PVOL(Pa)	PH2O(Pa)			
CV 160	7.00E+06	0.0			
CV 200	6.825E+06	0.0			

Table A19 (CV's 200, 160)

CVnnn	Altitude(m)	Volume(m^3)
CV 160	8.5	0.00
	8.75	0.015625
CV 200	-0.50	0.00
	0.0	1.25

FL's for src-snk-ext.gen – FL 210 and 200 – System source to rpv and rpv to system sink

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
FLnnn00	FLNAME		User-defined FL name
	KCVFM		“From” CV number
	KCVTO		“To” CV number
	ZFM		Altitude of “From” junction
	ZTO		Altitude of “To” junction
FLnnn01	FLARA		Flow path area
	FLEN		Flow path length
	FLOPO		Fraction of FP open
FLnnn02	KFLGFL		FP type
	KACTFL		FP flag
	IBUBF		No bubble rise physics
	IBUBT		No bubble rise physics
FLnnnS1	SAREA		Segment flow area
	SLEN		Segment length
	SHYD		Segment hydraulic diameter
FLnnnT1	NTFLAG		Time dependent FL type flag; 1 for TF, 2 for CF
	NFUN		TF or CF number defining velocity vs. time
FL601V1	NVTRIP		Negative, then FLOPO defined by “on-forward” CF
	NVFONF		CF “on-forward” to describe FLOPO at “on-forward” state of trip time
	NVFONR		CF “on-reverse” to describe FLOPO at “on-reverse” state of trip time

TABULATED VALUES IN TABLE A20

Table A20 (FL's 200, 210)

FL #	FLNAME	KCVFM	KCVTO	ZFM(m)	ZTO(m)
FL210	SOURCEtoINLET	160	170	8.625	8.625
FL200	OUTLETtoSINK	100	210	-0.27	-0.27
FL #	FLARA(m^2)	FLEN(m)	FLOPO	KFLGFL	KACTFL
FL210	1.0207	6.15	1.0	3	0
FL200	1.0207	2.0	1.0	3	0
FL #	IBUBF	IBUBT	SAREA(m^2)	SLEN(m)	SHYD(m)
FL210	0	0	1.0207	6.15	0.19
FL200	0	0	1.0207	2.0	0.19
FL #	NTFLAG	NFUN	NVTRIP	NVFONF	NVFONR
FL210	2	200	-1	500	199
FL200	-	-	-1	500	199

CF499, 500, 198, 199, 200 – used to define steady state sources and FL openings

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
CFn...n00	CFNAME	TABULATED VALUES IN TABLE Q21	User-defined CF name
	CFTYPE		CF type; EQUALS sets the given parameter equal to CF argument
	NCFARG		Number of arguments
	CFSCAL		Multiplicative scale factor for CF
	CFADCN		Additive constant for CF
CFn...n01	CFVALR/LCFVAL		Initial CF values; CFVALR if real, LCFVAL if logical (integer)
CFn...nKK	ARSCAL		Multiplicative scale factor
	ARADCN		Additive constant
	CHARG		Database element identifier

Table A21

CF #	CFNAME	CFTYPE	NCFARG	CFSCAL	CFADCN
CF198	VSOURCE	MULTIPLY	2	1.0	0.0
CF199	VSOURCE	DIVIDE	2	1.0	0.0
CF200	chkVSRC	MULTIPLY	2	1.0	0.0
CF499	Lggo	L-GT	2	1.0	0.0
CF500	Lgtrip	L-A-IFTE	3	1.0	0.0
CF #	CFVALR	ARSCAL	ARADCN	CHARG	
CF198	-	1.0	0.0	CVH-RHO.160	
		0.0	1.0207	TIME	
CF199	-	1.0	0.0	CFVALU.198	
		0.0	129.0	TIME	
CF200	-	1.0	0.0	CFVALU.500	
		1.0	0.0	CFVALU.199	
CF499	-	1.0	0.0	CFVALU.005	
		1.0	0.0	TIME	
CF500	1	1.0	0.0	CFVALU.499	
		0.0	1.0	TIME	
		0.0	0.0	TIME	

FILE9 core-pow-prof.gen

CORR, CORZ for power density profile

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
COR00003	FCNCL	0.25	Rad. exchange factor; canister wall to rod cladding
	FSSCN	Same as FCNCL	Rad. exchange factor; “other structure” to canister or cladding
	FCELR		Rad. exchange factor; Radially out outward from cell to adjacent cell
	FCELA		Rad. exchange factor; Axially upward from cell to next adjacent cell
	FLPUP		Rad. exchange factor; Liquid pool to core components
CORZjj03	FZPOW	See basis	Relative power density at NAXL (axial level) = jj; <u>TABLEA22</u>
CORRii03	FRPOW	See basis	Relative power density at NRAD (radial ring) = ii; <u>TABLEA22</u>

Table A22 (CORRii,CORZjj)

jj (NAXL)	FZPOW	ii (NRAD)	FRPOW
21	0.613	5	0.904
20	0.756	4	0.879
19	0.891	3	1.000
18	0.972	2	0.689
17	1.000	1	0.000
16	0.985		
15	0.937		
14	0.866		
13	0.780		
12	0.688		
11	0.595		
10	0.505		
9	0.420		
8	0.341		
7	0.269		
6	0.205		
5	0.156		
4	0.000		
3	0.000		
2	0.000		
1	0.000		

CF's for power trip mechanism

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
CFn...n00	CFNAME	TABULATED VALUES IN TABLE A23	User-defined CF name
	CFTYPE		CF type; EQUALS sets the given parameter equal to CF argument
	NCFARG		Number of arguments
	CFSCAL		Multiplicative scale factor for CF
	CFADCN		Additive constant for CF
CFn...n06	MSGFIL		Flag indicating files to which CF writes message
			0=no message; 1=STDout; 2=STDout, message, and log files
	SWTMSG		Switching message; some user-specified text string : 'string'
CFn...nKK	CF006		MSGFIL=2; SWTMSG='BEGINNING LOFC TRANSIENT'
	ARSCAL		Multiplicative scale factor
	ARADCN		Additive constant
	CHARG		Database element identifier

TABLE A23 (CF5-CF60)

CF #	CFNAME	CFTYPE	NCFARG	CFSCAL	CFADCN	ARSCAL	ARADCN	CHARG
CF005	TRIPTM	EQUALS	1	1.0	0.0	0.0	1.6e5	TIME
CF006	ift	L-GT	2	1.0	0.0	1.0 1.0	0.0 0.0	TIME CFVALU.5
CF007	timeif	L-A-IFTE	3	1.0	0.0	1.0 0.0 -1.0	0.0 0.0 0.0	CFVALU.6 TIME CFVALU.5
CF010	DIFTIME	ADD	3	1.0	0.0	1.0 1.0 0.0	0.0 0.0 1.0e-6	TIME CFVALU.7 CFVALU.7
CF011	shift	L-A-IFTE	3	1.0	0.0	1.0 0.0 1.0	0.0 1.0e-6 0.0	CFVALU.6 TIME CFVALU.10
CF012	lt10	L-GT	2	1.0	0.0	0.0 1.0	10.0 0.0	TIME CFVALU.11
CF013	lt150	L-GT	2	1.0	0.0	0.0 1.0	150.0 0.0	TIME CFVALU.11
CF014	lt17	L-GT	2	1.0	0.0	0.0 1.0	1.0e7 0.0	TIME CFVALU.11
CF015	till10	L-A-IFTE	3	1.0	0.0	1.0 0.0 0.0	0.0 1.0 0.0	CFVALU.12 TIME CFVALU.11
CF016	till150	L-A-IFTE	3	1.0	0.0	1.0 0.0 0.0	0.0 1.0 0.0	CFVALU.13 TIME TIME
CF017	till1e7	L-A-IFTE	3	1.0	0.0	1.0 0.0 0.0	0.0 1.0 0.0	CFVALU.14 TIME TIME
CF018	ADDT	ADD	3	1.0	0.0	1.0 1.0 1.0	0.0 0.0 0.0	CFVALU.15 CFVALU.16 CFVALU.17
CF019	un10	L-EQ	2	1.0	0.0	0.0 1.0	3.0 0.0	TIME CFVALU.18
CF020	un150	L-EQ	2	1.0	0.0	0.0 1.0	2.0 0.0	TIME CFVALU.18
CF021	un1e7	L-EQ	2	1.0	0.0	0.0 1.0	1.0 0.0	TIME CFVALU.18
CF022	if10	L-A-IFTE	3	1.0	0.0	1.0 0.0 0.0	0.0 1.0 0.0	CFVALU.19 TIME TIME
CF023	if150	L-A-IFTE	3	1.0	0.0	1.0 0.0 0.0	0.0 1.0 0.0	CFVALU.20 TIME TIME
CF024	if1e7	L-A-IFTE	3	1.0	0.0	1.0 0.0 0.0	0.0 1.0 0.0	CFVALU.21 TIME TIME
CF040	INITPOW	EQUALS	1	1.0	0.0	0.0	268.0e6	TIME
CF051	t1	POWER-V	2	1.0	0.0	1.0 0.0	0.0 -0.0639	CFVALU.11 TIME
CF052	t2	POWER-V	2	1.0	0.0	1.0 0.0	0.0 -0.181	CFVALU.11 TIME
CF053	t3	POWER-V	2	1.0	0.0	1.0 0.0	0.0 -0.283	CFVALU.11 TIME
CF054	per1	MULTIPLY	4	1.0	0.0	0.0 1.0 1.0 1.0	0.0603 0.0 0.0 0.0	TIME CFVALU.40 CFVALU.51 CFVALU.22

CF055	per2	MULTIPLY	4	1.0	0.0	0.0 1.0 1.0 1.0	0.0766 0.0 0.0 0.0	TIME CFVALU.40 CFVALU.52 CFVALU.23
CF056	per3	MULTIPLY	4	1.0	0.0	0.0 1.0 1.0 1.0	0.130 0.0 0.0 0.0	TIME CFVALU.40 CFVALU.53 CFVALU.24
CF058	timead	ADD	3	1.0	0.0	1.0 1.0 1.0	0.0 0.0 0.0	CFVALU.54 CFVALU.55 CFVALU.56
CF059	PowTRIP	L-GT	2	1.0	0.0	1.0 1.0	0.0 0.0	CFVALU.5 TIME
CF060	CORPOW	L-A-IFTE	3	1.0	0.0	1.0 1.0 1.0	0.0 0.0 0.0	CFVALU.59 CFVALU.40 CFVALU.58

FILE10**pblr_edf.gen****External data file writing and reading directions**

<u>CARD</u>	<u>WORD</u>	<u>VALUE</u>	<u>BASIS</u>
EDFnnn00	EDFNAME	TABULATED VALUES IN TABLE A24	User-defined external data file name
	NCHAN		Number of channels (dep. var's) in each record of file
	MODE		Direction of information transfer (READ,WRITE,PUSH)
EDFnnn01	FILNAM		File name on operating system
EDFnnn02	IFMT		Format of records in external file
EDFnnn1k	TWEDF		Time at which following output increment takes effect
	DTWEDF		Time increment between output records
EDFnnnkk	CHARG		CF argument to identify element of database

Table A24 (EDF)

EDFnnn	EDFNAME	NCHAN	MODE	FILNAM	IFMT	TWEDF	DTWEDF	CHARG
EDF111	EDF_OUT	3	WRITE	edf.txt	4e12.5	For 1k=10, 0.0 For 1k=11, 15000.	For 1k=10, 500. For 1k=11, 10.	For kk=AA, CFVALU.605 For kk=AB, CFVALU.608 For kk=AC, CFVALU.609

HS surface area calculation summary:

For HS4400X, X=0...4, each structure is of cylindrical geometry has the following characteristics:

- Radius of LHS (inner radius) = 2.87 m
- HS thickness (Δr) = 0.05 m
- Radius of RHS (outer radius) = 2.92 m

Then, LHS and RHS (inner and outer) HS surface area values can be calculated by :

$$S.A._{LHS/RHS} = (\text{Circumference}_{LHS/RHS}) * (\text{Height}_{LHS/RHS})$$

Table A25(HS surface areas)

HS44nnn	CV on LHS	CV on RHS	$\Delta Z(m)$	Circumference(m)	S.A(m ²)
HS44000	300	400	0.5508	LHS = 18.0327266 RHS = 18.3468856	LHS = 9.9324 RHS = 10.1055
HS44001	301	401	2.5	LHS = 18.0327266 RHS = 18.3468856	LHS = 45.0818 RHS = 45.8672
HS44002	302	402	2.0	LHS = 18.0327266 RHS = 18.3468856	LHS = 36.0655 RHS = 36.6938
HS44003	303	403	2.0	LHS = 18.0327266 RHS = 18.3468856	LHS = 36.0655 RHS = 36.6938
HS44004	304	404	2.0	LHS = 18.0327266 RHS = 18.3468856	LHS = 36.0655 RHS = 36.6938

ADAPT Admin Guide

This admin guide should get you up and running with installing and running the adapt system as a normal non-root user (desired mode of operation).

Installation

Installing prerequisites on a linux cluster

You should install or ensure the following packages are installed:

- mysql 5.0 or above
- python 2.4 or above
- python-mysql driver (MySQLdb)
- python cherrypy 3.0 web dev environment (<http://www.cherrypy.org/>)
- graphviz (<http://www.graphviz.org/>)

All of these packages can be installed as root or non-root, take your pick. It's probably easiest to use your distro's packaging tools and install as root as much as possible.

Downloading the sources

You should acquire the source distribution to adapt, which is under export control. Save it somewhere (e.g. your home directory). Unpack it to contain directories 'src', 'doc', 'test', etc. I will refer to the directory you unpacked into in this document as ADAPT_HOME.

Modifying your environment

You should add the 'src/server' directory from the source distribution to your environment. Put a statement in your ~/.bashrc file such as:

```
export PATH=ADAPT_HOME/src/server:$PATH
```

Don't forget to log back out and back in, or source your environment file.

Creating a configuration file

You should create a ~/.adaptrc file as follows:

```
[server]
jobmanager = rsh
log = /home/rutt/var/log/adapt.log

[database]
host = your.database.host
```

```

port = 3306
user = rutt
pass = secret
db   = adapt

[rsh]
rsh = ssh -x -o StrictHostKeyChecking=no
rcp = scp -q -o StrictHostKeyChecking=no
hosts = mob01 mob02 mob03 mob04 mob01 mob02 mob03 mob04
directory = ^mob:/data/archive/adapt

[webmin]
users = rutt:ruttpassword aram:arampass kyle:secret
port = 47833
baseurl = http://your.web.hosting.machine:47833

```

In the above, most should be self explanatory, although the following comments are useful:

- your cluster is defined by your `hosts = mob01 ...` line. Feel free to list a host multiple times if it has multiple CPUs for better performance.
- if you want to use rsh instead of ssh as your remote job startup mechanism, replace the above block with the following:

```

[rsh]
rsh = rsh
rcp = rcp
...

```

- the `directory = ^...` line contains Python regular expressions matching hostnames, followed by a colon, followed by where output files should go for matched hostnames. If you have hosts named similarly you can use one regular expression/colon/directory string. If not, use something like `^mob:/data/archive/adapt ^foohost:/tmp/foo/adapt`.
- if you use ssh as your remote shell command, you should set it up so you can ssh to your cluster nodes without entering a password. This is best solved by reading some docs such as doing a google search for 'ssh without password'.
- for the webmin entry, you should list your users and passwords on a single line, the port you want to run the server on, and the URL at which the server is reachable.

Setting up the database

Next, you should run the database administrative actions to create the database specified in your `~/.adaptrc` file. Connect to mysql as a root user. In another terminal, run the command `adapt-mysql-admin` and it should tell you the commands to sent to mysql to grant privileges for your database user to create tables in mysql under the database you've specified above. Copy and paste these into mysql and you should be all set.

Testing the installation

Try running the executable `adapt-checkhosts` and it will try to read your configuration file and ensure you have access to all the hosts in your cluster (i.e. your `hosts = mob01 ...` line) by running a remote shell to each of them.

Next, if that succeeds, you can try running the executable `adapt-server` which is the long running server process. It should create any necessary database tables on startup if they don't already exist

(which they won't at this time). In another terminal, run the command **adapt-server-halt** to stop the server.

Next, if that all succeeds, you can try starting up the web server, which is done by running the executable **adapt-webmin**. If that succeeds, you can point your web browser to the URL given in the above config file 'baseurl' key. You can stop the web server with Ctrl-C.

Running the services

There are two services that should run at all times, the **adapt-server** and the **adapt-webmin** processes.

adapt-server

This is the core long running server process. It does not fork itself off as a daemon. As long as it is running, branches in adapt will have a chance to execute. If it is stopped, the adapt branches will not make any progress. If you want to run it for a fixed period of time, look at the argument **-r**, e.g. **adapt-server -r 8h** to run for 8 hours, **adapt-server -r 5m** to run for 5 minutes, etc. Traditionally, a new experiment was submitted with the command **adapt-submit** and the server process would pick it up and begin executing it; however, a simplified interface for end-users was later developed, which is known as adapt-webmin.

To cleanly stop the adapt-server, you should run the command **adapt-server-halt** which takes no arguments. If you do not stop it this way it could leave some runaway processes behind.

adapt-webmin

This is the back end process for the Web server. You should run it when you want to use the Web based interface to the system. The Web based interface is useful to give a simplified interface for end-users. It also has some administrative features, e.g. it allows the registration of new simulator types, such as melcor, so the user need only bring the minimum necessary input files to get a new experiment going.

Registering simulators

Using the web interface, you should register your simulator so it can be executed by users. An example will be given for melcor simulator now.

1. Visit the main web interface. Click option 3, *add a new simulator*:

ADAPT main menu

1. list experiments
2. launch a new experiment
3. add a new type of simulator

2. Add the new simulator with its respective wrapper scripts and also name any of its input files (the wrapper scripts can be found in the ADAPT_HOME/test directory):

ADAPT main menu::add simulator

Name of simulator:

Web wrapper script:

Main wrapper script:

Add inputs for simulator:

label for input:	<input type="text" value="sbo.cor.tpl"/>	provided by user ▼
------------------	--	--------------------

label for input:	<input type="text" value="sbo.rst"/>	provided by user ▼
------------------	--------------------------------------	--------------------

label for input:	<input type="text" value="sbo_editrules.cor"/>	provided by user ▼
------------------	--	--------------------

label for input:	<input type="text" value="melcor-checkpoint"/>	provided by installer ▼
------------------	--	-------------------------

label for input:	<input type="text" value="melcor"/>	provided by installer ▼
------------------	-------------------------------------	-------------------------

3. For any input files provided by the administrator (i.e. you, the reader of this document), fill them in now for upload (the melcor-checkpoint script and the melcor executable can be found in the ADAPT_HOME/test directory):

Please input the installer definitions below

input melcor:

input melcor-checkpoint:

Click submit and it should be registered.

Contact Info

If you have any bug reports or further questions, send them to adapt@lists.bmi.ohio-state.edu.

1. Introduction

This document gives a description of the ADAPT DET Generation Tool and covers the necessary steps to create input for the ADAPTA software as well as analyze the results from ADAPT experiments with a Graphical User Interface (GUI). Section 2 of this document covers an overview of the ADAPT software as well as a description of the necessary input to generate an ADAPT experiment. Section 3 will cover the operation of the ADAPT GUI.

2. ADAPT User's Guide

Since the development of APETs through manual runs of severe accident simulators is a time-costly process, ADAPT was developed to generate dynamic APETs automatically by means of controlling the simulator execution and dynamic and automatic manipulation of the simulator's input files. ADAPT is designed to take any severe accident simulator and use it as a means of developing a dynamic APET. The algorithm for a single branching point can be described by Figure 2.1.

Figure 2. 1: Pictorial representation of the ADPT algorithm at a single branch.

When an experiment is started, the simulator input files are initialized based on rules specified in a simulator-independent ADAPT input file, referred to as the *edit-rules* file

(described below). Next, the simulator is executed and it runs until a user-specified branching point has been reached. At this point the simulator execution quits and the output file is parsed for the termination cause. When the cause is found ADAPT will check with the *edit-rules* file to determine if the termination cause warrants new branches being created. If new branches are to be created then ADAPT will make the proper changes to the simulator input files and execute the new branches. Branching rules for ADAPT can come in the form of active component set points (i.e. pressure for a valve to open) or discrete points in a cumulative distribution function (CDF) for a particular failure (i.e., a fragility curve for pipe failure or time for power recovery).

The ADAPT software consists of four main components which work together to create, communicate, and assist in the analysis of DETs. These four parts are:

- 1) The ADAPT server – a long running process which waits for new branches and experiments and executes the simulator when necessary
- 2) The ADAPT database – a MySQL database which stores event tree metadata
- 3) The ADAPT web-server – a web-based service which provides a means of communicating with the ADAPT server outside of the server's host network as well as a means of launching experiments and contains a limited graphical capability for event tree visualization
- 4) The ADAPT Graphical User Interface (GUI) – a Java based software which can be used on user's local machine to launch experiments, display the event tree, and assist the user with more complex analyses of the generated event tree.

The ADAPT server, database, and web-server are designed to run on any Linux environment while the GUI, since it is based in Java, can work on any operating system which supports the Java Runtime Environment.

Figure 2. 2: Diagram of information flow between different components in the ADAPT software.

Figure 2.2 shows the direction of information transfer for each component of the software. The Driver, Compute Cluster, and Database are in constant communication to facilitate the execution of the tree. The Java client communicates information to the web-server in order to have control of experiment execution. The client also downloads information from the database in order to display the tree as well as other information. However, the client never directly modifies the database. The STORM middleware (Section 2.4) is utilized to query the compute cluster file system and download plot data which is to be sent to the client. STORM attains the location of all branch data from the database and then downloads it from the compute cluster itself.

2.3 ADAPT Input Overview

In order for a simulator to be compatible with ADAPT it must have three characteristics:

- 1) It must have a text input file
- 2) It must be able to stop itself under user-specified conditions
- 3) It must have restart capability

In order for an experiment with a particular simulator to be launched by ADAPT, the user is required to construct the following:

- 1) A simulator wrapper (script used to execute the simulator),
- 2) A template input file,
- 3) An edit-rules file,
- 4) A checkpoint script, and
- 5) Any other simulator files.

The first component in the above list, the wrapper, is the integral piece behind linking a simulator to ADAPT. The wrapper is an executable script which instructs ADAPT on how to execute the specified simulator (execution commands, command line arguments, etc.), how to read from its output, and to communicate this information to the ADAPT database. A myriad of command-line tools is provided by the ADAPT software package to assist with this last point.

The template input file is utilized by ADAPT to make changes to the simulator input each time a branching occurs. The template file is created by taking the input file for the desired simulator and replacing variables of interest with ADAPT-specific tags which will instruct ADAPT as to what elements in the input to file to replace and when.

The edit-rules file is used to set up which branching rules are desired for this experiment. The edit-rules file is written in an ADAPT-specific format (the format is independent of what simulator is used) and contains a list of all branching rules. The branching rules will detail what to change in the input file when the code execution stops for various reasons. The edit-rules file

also contains the user-input branching probabilities and initial parameters to be used for the first branch in the experiment.

The checkpoint script is required for the ADAPT feature to pause and restart entire experiments. The checkpoint script provides a means of externally stopping the code and creating a restart file such that the experiment can be launched at a later time from that point. The script must simply execute a command which will perform the above-described tasks for a particular simulator.

Finally, some simulators may have more input requirements than just a text input file. In the case of the MELCOR severe accident analysis code, as described above, the restart file is an additional, binary file that must be present when the simulator is executed. Files such as this must also be prepared (i.e., a simulator input deck must already exist).

In the next few sub-sections, details regarding the structure of each of the above-described input files will be provided. Since the MELCOR has already been linked to ADAPT, it will be used in many examples.

2.3.1 Simulator Wrapper Algorithm

For each simulator that is used with ADAPT, a wrapper script must be written. The purpose of the wrapper script is to instruct ADAPT on how to execute the simulator as well as post-process and communicate the resulting information to the ADAPT database. All wrapper scripts must follow an algorithm similar to what is pictured in 2.3.

Figure 2. 3: General algorithm which must be followed by a simulator wrapper script. Those items marked with an asterisk (*) are optional and were implemented in the MELCOR wrapper script.

Figure 2.3 has been constructed based on the wrapper script that is currently used for MELCOR. The logic here should be considered as a general algorithm for wrappers written for other simulators. Actions which are optional but which have been deemed useful in the case of MELCOR have been marked by an asterisk (*). The additional MELCOR-related steps have been shown to demonstrate the flexibility of the simulator wrapper. The steps shown in the figure are described below.

1. The first step in the procedure requires that the root branch be initialized to the values specified in the edit rules file. The command *editrule-apply* is provided by ADAPT to automatically perform this procedure.

2. The second step in the algorithm is optional, namely the setting of the early termination criterion. This allows the user to set the maximum allowed time (wall clock time) each branch will run. If a branch runs longer than this time, it will become checkpointed and execution will be completed at a later time. This feature is used to prevent long-running branches from clogging up the server.
3. The third step executes the simulator. At this point in the wrapper, the necessary execution steps and command line arguments must be entered. Steps 4 and 5 begin the process of determining the cause for the simulator termination.
4. Step 4 will determine if the cause of simulator termination is the result of the experiment being checkpointed. If the experiment has been checkpointed, the file *adapt.cp* will be placed in the execution directory by ADAPT. Searching for this file from the wrapper will determine whether or not checkpointing was the cause of termination. If this is the case, the script must quit.
5. Step 5 checks if the termination cause is because the execution time ran beyond the early termination criterion. If this is the case, ADAPT will have placed the file *term-early* in the execution directory. Checking for this file will determine whether or not early termination is the cause of the simulator termination. If this is the case, the script will proceed no further.
6. Step 6 is an optional step but has been implemented in the MELCOR wrapper script. This step truncates the restart file such that it only contains

the most recent restart dump. This allows for a smaller file to transfer to the next branch. Such a step may be useful if the restart files become large, and is not essential.

7. In step 7 the script needs to search the simulator output for the termination cause and ending simulation time and submit this information to the ADAPT server with a script provided by the ADAPT software. Finally, steps 8 and 9 are charged with checking the edit rules file to determine if there are more branches to submit.
8. In step 8, the *editrule-apply* script is utilized once again. If no more branches are to be submitted then the script quits. If there are child branches which need to be submitted,
9. Finally, the wrapper runs the *adapt-submit* command to submit the required child branches.

2.3.2 Template Input File

ADAPT takes a template of the simulator's input file for a particular experiment as input. What is meant by "template" can be illustrated by the following example which uses a MELCOR input.

As previously mentioned in Section 2.1, MELCOR utilizes "control functions" to allow users to specify new variables without having to modify the MELCOR source code and recompile it. Assume that in the MELCOR input file there is a control function value to be used as a branching condition. Recall from Section 2.1 that the user writes the main problem description into input files for MELGEN which will create a binary input file, or restart file, to

be used by MELCOR. MELCOR also takes in a text file as input (See Section 2.1) to change experiment specific parameters and allows the alteration of some problem values. As the experiment progresses, the restart file is appended and when the simulator quits, the user can restart the simulation from the point at which it stopped. Also, using the MELCOR input file, some experiment-specific parameters can be changed when restarting the code. Figure 2.4 gives a picture of some example MELGEN input while Figure 2.5 gives an example of some MELCOR input. The MELGEN input will set up the logic for our branching rules while the MELCOR input will set up the values at which that logic changes.

```
CF10000 'TimeStop' equals 1 1.0 0.0
CF10010 0.0 200.0 time

CF10100 'TimeCmpr' 1-gt 2 1.0 0.0
CF10106 2 "SHIBBOLETH 101 Something failed" *Message to be printed out when this becomes true
CF10110 1.0 0.0 time
CF10111 1.0 0.0 cfvalu.100

*this will be true when the simulation time becomes greater than the value of CF100, in *this case, 200.0

CF899900 'StopCF' L-EQUALS 1 1.0 0.0
CF899910 1.0 0.0 cfvalu.101

*This will stop the code if CF101 becomes true
```

Figure 2. 4: Example MELGEN input

```
*This is desired we wish to change the value of CF100 from execution to execution

CF10001 2000.0
CF10002 3 0.0 2000.0

*This will raise the value of CF100 to 2000.0, which will increase the amount of time the *code will run (from 200.0 to 2000.0)
```

Figure 2.5: Example MELCOR input

In the above example, the MELGEN input defines a series of control functions which will simply stop the code execution at a user-specified simulation time. Such logic would be useful for something like power recovery which may not be modeled within the simulator and whose probability of occurrence only depends on the simulation time. In the case above, there are three control functions* defined. Control function 100 is real-valued and its value is the time at which it is desired to stop the code. Control function 101 compares the simulation time with the value of control function 100 and will become true when the simulation time becomes greater than this value. Finally, control function 8999 is used to stop the execution (MELCOR allows the user to define a control function for this purpose). The code execution will stop when the value of this control function becomes true. In this case, it will become true when control function 101 becomes true.

Next, in the MELCOR input, in order to have control over the time at which the code quits, the value of control function 100 will be altered (MELCOR allows the modification of control functions in MELCOR input but new control functions may be defined only in MELGEN input, See Section 2.1). In this example case, the value at which control function 101 becomes true is raised from 200 to 2000.

To use the above case as an example and translate what it might be in ADAPT input, only the MELCOR input is examined. For ADAPT input, it would look similar to the following:

CF10001 {V100}
CF10002 3 0.0 {V100}

Figure 2. 6: Example template input for ADAPT input.

In Figure 2.6 the text “{V100}” is an ADAPT variable whose values are defined in the edit rules file. Before MELCOR is executed in each branch, ADAPT will replace these variables with values from the edit rules files depending on the branching conditions. In this way, ADAPT has dynamic control of the variables in the simulator’s input and branching rules can be applied as the tree grows.

2.3.3 Edit-Rules File

The edit-rules file will contain a list of all of the branching rules for a particular experiment. The format of the edit-rules file will be explained by the following example which will be a continuation of the one given above. Figure 2.7 gives an example edit-rules file input.

```

editrulesExample.txt melcor-wrapper-web
1 Input file: sbo.cor.tpl          // Input file which contains variables to be changed
2
3
4 Stopping Rule: sbo.mes SHIBBOLETH 2 // <filename> <magicword> <word_on_that_line>
5
6 Separator: "{" "}"              // These are the brackets that separate the variables
7
8
9 T1 5 400.0 600.0 800.0 100.0 1e20 // Table T1 contains elements which are branching rules
10 T1p 5 5 25 50 75 95 95          // for stopping code 101
11
12 INIT V100 200.0
13
14 101 1 V100 T1                    // Changes to be made if CF 101 stops the execution
15 101 2 V100 1e20

```

Figure 2. 7: Example edit-rules file input.

The first line in the edit-rules file gives the name of the template input file which is to be altered every time a branching occurs. Next, the line beginning with “Stopping Rule” provides ADAPT with the place to look in the simulator output to find the reason why the code stopped.

The syntax of that line is as follows: the first element contains the name of the file to search, the second element is a unique word (SHIBBOLETH in the example) used to identify the on which line the reason for the code stopping can be found, and finally, the number “2” in this case is used to tell ADAPT that the second word on this line (which contained the word “SHIBBOLETH”) will yield the termination cause. In the example case it will pick out the control function number from the message that was printed from control function 101. Note from Figure 2.4 that a unique word had to be specified in a printed message from control function 101. MELCOR has no unique word that is printed when a particular control function changes values so one needs to be forcibly printed to MELCOR’s message file. The next line lets ADAPT know which characters will be used as separators for ADAPT variables. All variables in the template file must be surrounded by these separators in order to be recognized by ADAPT.

The remainder of the input deals with the branching rules, initialization rules, and branching probabilities. Line 5 gives a table of values which are to be used as branching rules. The table above is labeled as “T1” and all tables must follow this syntax, namely “T<number>”. After the table label, the next value gives how many elements are in that particular table and the remainder of the entries are the values contained in the table. The next line is a table which contains the probabilities (given in percent) for table 1. It is labeled as “T1p” and all other probability tables must follow the same naming convention namely, “T<number>p”. The specified value of <number> will tell ADAPT which table these probabilities are for. Line 8 is a variable initialization, notated by the term INIT. The syntax is as follows: INIT (variable to initialize) (initial value). In the example case shown in Figure 2.7, variable V100 is initialized to the value of 200.0.

The last two lines instruct ADAPT on what to change in the template input file when control function 101 stopped the execution. Note here that the first element in this line must match up with the stopping cause found from the “Stopping Rule” line. The syntax of this input is as follows: (stopping rule) (branch number) (variable) (value). For the example case, in line 14, when stopping code 101 is found to have stopped the execution, for the first branch change the value of variable V100 to the next element in table 1. This will also assign this branch the next probability value in the table T1p. More specifically, this branch will change V100 to the value of 400.0 and this branch will have a 5% probability. Line 15 deals with the second branch which occurs from this stopping code. This line instructs ADAPT to change the value of variable V100 to the value of 1.e20. The probability of this branch will be 1 minus the probability of branch 1. If there are multiple variable changes that are required for a particular branching, they are simply written them out as follows:

```
101 1 V100 T1
```

```
101 1 V101 T2
```

```
101 1 V102 T3
```

```
...
```

If values are specified rather than table numbers they must appear in the same format which would be recognized by the simulator being used. It is important to note that if there are more than two branches for a particular stopping code, then tables with associated probability tables must be used to describe the input changes. Namely, if a particular stopping code has N branches, N-1 of them must use tables to specify new input values since all branches must have

an associated probability. The last branch that does not have a specified probability will simply have a probability of 1 minus the sum of the probabilities of the other branches.

3. ADAPT GUI User's Guide

3.1 Introduction

This chapter will outline the basic use of the ADAPT GUI. First, the steps required to install and run ADAPT, along with the system requirements will be discussed. The remainder of this chapter will be devoted to a user's manual which will describe the basic operation of the software in its current state.

3.2 ADAPT-GUI Installation

The installation of the ADAPT GUI is a straightforward process. There is no installer package; the user must simply obtain the ADAPT GUI directory and place it somewhere on his/her computer. It can be placed in any directory, since ADAPT does not make assumptions as to the user's directory structure. The source folder will contain the following items:

- 1) The folder *ADAPTJar*.
- 2) The folder *ADAPTSubmit*.
- 3) The file *adapt.jar*

The folder *ADAPTJar* contains many of the .jar libraries used by the software. The folder *ADAPTSubmit* contains all of the .jar libraries used to provide functionality to connect to the ADAPT web server and STORM server. The file *adapt.jar* is the executable .jar file which can be used to launch ADAPT. To make ADAPT more accessible the user can make a link to this file and place it anywhere he/she wishes. In order to use ADAPT the user must have Java Runtime Environment version 1.5 or higher installed, 64 MB of RAM, and at least a Pentium III, 1.0 GHz processor equivalent or greater.

In order to launch the ADAPT GUI, the user can either: 1) double-click on the file *adapt.jar* (or a suitable link) or 2) from the command line run “java -jar adapt.jar”. Either approach will successfully run the program.

3.3 Projects

In the ADAPT GUI framework there two different kinds of projects that a user can create: static and dynamic projects. Static projects are designed to be for more conventional event trees that have no time information associated with branch position on the tree. Position on the event tree for conventional event trees is determined by which *top event* the event falls under. For these types of projects it is assumed that a previous analysis has been performed and there is simulator output data available. As input for a static project the user provides the location of these plot data and ADAPT can recall that data when it is desired. The purpose of having static projects is to 1) assist the analyst with previous, conventional event tree analyses and 2) to compare conventional event trees with one another and dynamic analyses. Dynamic projects are those that are created by the ADAPT Server. For dynamic projects, the client works in conjunction with the ADAPT Server and database to display and gather information about the event tree.

3.4 Setup

Data for both dynamic and static projects are stored in the ADAPT database which is usually situated on a remote machine (for the purposes of development, all static databases are stored on the user’s local machine). In order for the ADAPT client to be able to retrieve data, the

user's machine must be able to make connections to this database. All that is required from the user is the IP address of the host machine, the port number where the database is being hosted and the user name and password required for database access. If there is a firewall between the client and the server, one can use a proxy or tunnel as appropriate for the shell client.

3.5 Entering Database Information

After the software has been launched, the menu bar at the top of the ADAPT window can be used to go to Edit -> Settings. This will bring up a dialog window that prompts the user to enter information (See Fig. 3.1) on 1) the credentials required for access to the database, 2) the credentials necessary to access the ADAPT web server, and 3) the information necessary to access the STORM server.

In order to access the ADAPT database, the user requires the following information: 1) a username which is authorized to access the database, 2) a password for that username, 3) the host name where the database resides, 4) the database name (the name will be “adapt” or dynamic projects and “static” for static projects), and 5) the port on which the database is being hosted. The database username and password must be provided by the system administrator as well as where the database is being hosted.

The ADAPT web server is used for launching new ADAPT experiments with the client as well as downloading files from the server. In order to access these capabilities in the client the web server must be running and a user name and password are required.

Enter new database information	
User Name:	root
Host Name:	127.0.0.1
Port:	46306
STORM Host:	127.0.0.1
Web URL:	http://bm70.bmi.oar.net
Web User:	kyle
Password:	
Database:	adapt
STORM Port:	10064
Web Port:	47832
Web Pass:	

OK

Figure 3. 1: Database settings dialog.

3.6 ADAPT Display

This section will cover the basic features of the ADAPT display system. The features here are similar in both dynamic and static event tree modes. Unique features of each mode will be discussed in Sections 3.8 and 3.9.

The display of ADAPT is a multi-panel design with each panel performing a unique function. A diagram of the ADAPT display can be seen in Figure 3.2 below. The display consists of 2 panels both containing sub-windows or sub-panels.

- 1) *Main Panel:* This panel contains all windows that can be opened in ADAPT. Event trees, plots, tables, and analysis results are all displayed in this panel. This panel does not contain any particular features but simply serves as a container for windows.

- 2) *Auxiliary Panel*: This panel contains two sub-panels, the *Info Box* and the *Open Project Manager*. These sub-panels have functionality to more easily move through and manipulate information displayed in the Main Panel.
- a. *Info Box*: For many types of window that are opened there is an associated *Info Panel* which displays information or tools which are specific to that window. The window types are: *the Event Tree Window*, *the Plot Window*, *the Branch Information Window* and *the Data Window*. Note that not all windows have an associated Info Panel dialog since they have not yet been developed.
 - b. *Open Project Manager*: This sub-panel contains a list of all open projects and windows in the form of a hierarchical tree (Figure 3.3). The top of the tree is “All Projects” and is split into “Dynamic Projects” and “Static Projects”. Each sub-node under Dynamic/Static Projects is a project name and each sub-node under the project name is an open window. From this panel the user can have direct control over the windows in the Main Panel. The user can activate, minimize, maximize and close windows by clicking (activate) or right clicking (minimize, maximize, close) on the desired node.

Figure 3. 2: Diagram of the different layers and panes of the ADAPT interface.

Figure 3. 3: Description of the Open Projects Manager.

3.7 Info Box Features

The Info Box panel will display different information depending on which type of window is selected. This panel serves not only to display information but to allow the user to interact with these windows in different ways. As mentioned above (Fig.3.2) there are five types of windows which are currently supported by the Info Box, namely an Event Tree Window, a Plot Window, a Branch Information Window, and a Data Window. An Event Tree Window is one which contains either a dynamic or static event tree display. A Plot Window is one which contains a plot (x-y plot, bar chart, pie chart) of either physical data or event tree statistics. A Branch Information Window is one which contains data for a particular branch on the tree. These windows are created when the user clicks on a particular branch (more details about this feature discussed below). Finally, a Data Window is one which displays overarching data for an entire event tree.

For a selected Event Tree Window, the Info Manager displays a list of all branches that are currently on the selected event tree. They are enumerated based on their unique id in the ADAPT database. When the user selects one of these branches in the list, the branch is highlighted on the tree and the tree focuses on that particular branch. This feature allows easier navigation through the tree. Also, right-clicking on the list of branches will allow the option to sort the branches in the list by id, stopping code, or ending simulation time.

When a plot window is selected the user is given various options for the particular plot selected. When a plot window is selected, the Info Box displays a series of options which allow the user to manipulate the plot data. The user can save plot data to a file, view the plot data, and perform analyses on the data.

The last two types of windows, the Branch Information Window and the Data Window do not have Info Box panels which have been developed. These options are going to be built in to future versions of the code.

3.8 Dynamic Projects

This section will cover all details of the dynamic module of the ADAPT client. Section 3.8.1 describes how to create and open dynamic projects. Section 3.8.2 – 3.8.3 describes all available features of the dynamic module.

3.8.1 Creating a Dynamic Project

In order to launch a new dynamic project, the user needs to go to File -> New -> Dynamic Project. This will bring up a dialog with several options (See Fig. 3.4). First there is a list of all available simulators that have been registered with the ADAPT server. If the user wishes to launch an experiment with a previously loaded simulator, he/she simply selects the desired simulator, enters a short experiment description (required) and click OK. This dialog also offers three other options. First, the user can delete the selected simulator. If the user clicks the “Delete Simulator” button he or she will be prompted to confirm the choice, click “Yes” to

confirm and “No” to cancel. Next, the user can update the selected simulator. This is used to update some of the files associated with that simulator. If the user clicks this button he or she will be prompted to enter new files which contain the updated versions of the desired files. Finally, the user can add a new simulator. Clicking this button will bring up a new dialog which will prompt the user to enter the names of all files required by this simulator. The user will upload all files which are used by all ADAPT executions with this simulator (i.e. the simulator executable) and simply give the names of all files which are required by the simulator but which will be different from execution to execution (i.e., input files, edit-rules file).

Figure 3. 4: Description of the “Select Simulator” dialog.

In order to launch an experiment from the ADAPT client, the ADAPT web server must be running on the host specified in the Edit -> Settings menu. The client communicates instructions to the ADAPT server by making HTTP posts to the ADAPT web site which in turn

makes modifications to the database which are then interpreted by the ADAPT Server. If the Web Server is not running an exception will be thrown.

3.8.2 Opening a Dynamic Project

In order to open any projects that have been created, the user needs to go to File->Open->Dynamic Project. This will bring up a dialog like the one shown in Figure 3.5 below.

Figure 3. 5: Description of the “Open Dynamic Project” dialog.

The Open Dynamic Project dialog gives a list of the available projects in the current database and for each project will give a list of the total number and type (type by status in the

queue) of branches that are in this particular project. The dialog will also show the date and time of submission as well as the status of the experiment. Experiments can have three different states:

- 1) *Completed* – All branches are either complete or considered insignificant, there are no branches waiting in queue or currently running.
- 2) *Active* – There are branches either running or in queue. Branches in this experiment are in the ADAPT Server queue and will be executed in the order they are submitted.
- 3) *Checkpointed* - There are branches in this experiment that still need to be executed and the experiment has been paused by the user. No branches in this experiment are placed in the ADAPT Server queue until the user restarts the experiment.

From this dialog the user also has the option of controlling the execution of the experiment and removing the experiment completely from the database. Checkpointing the experiment will pause the experiment execution. The user can restart the experiment if it has been previously checkpointed; this option is not available if the experiment is active. Finally the user can delete the experiment completely from the database and all simulator files from the cluster. The user must exercise caution here since this will delete the experiment for all users and deletion is permanent.

When the user has selected the project he/she wishes to open the user can select whether or not to open it for monitoring or not. If the box “Open Monitoring Session” is checked, the GUI will poll the database every 10 seconds and check for new additions or

changes and update the event tree accordingly. This option is designed for real time monitoring of currently running projects. If the user knows that the project will not be updated or the user wishes to look only at the project in its current state, un-check this box and the event tree will never be updated.

3.8.3 Dynamic Tree Display Features

One of the main purposes of the ADAPT Client is to provide a means of displaying the results of a dynamic event tree experiment which is generated by the ADAPT server. The client provides a graphical representation of the event tree, plots of the physical data, and performs analyses for the entire event tree. This section explains the features of the Dynamic Even Tree display system and how the user interacts with them.

The event tree display provides a means of graphically representing the event tree in a way that is typical to the style of displaying a conventional event tree with one major exception. In conventional event trees the horizontal axis represents an analyst-determined sequence of events and branchings only occur at these event points. In this dynamic event tree display system, the horizontal axis corresponds to time with the default display in seconds. Hence, distances between completed branches are determined by the actual difference in simulation time between the branches. Any branches which are not completed are placed an arbitrary distance away from their parent branch and this distance has no physical meaning. While an experiment is running, the event tree can be setup to constantly monitor the database

for new additions, when new branches are added they are placed an arbitrary (but constant) distance away from their parent and when the execution of this branch completes the branch will be replaced at the proper distance away from its parent branch depending on how long it took to complete.

On each branch that is displayed on the tree, there is a variety of information which is available to aide the user. There are two basic schemes for information display: one for completed branches and one for all other types of branches. The basic schemes are shown in Figs 3.6 and 3.7 below.

Figure 3. 6: Diagram of the information found on a completed branch.

Figure 3. 7: Diagram of the information found on a branch that is not completed (format is shared by all non-completed branches).

Each branch is given a unique id in the ADAPT database. This is the primary method of identifying different branches. For completed branches, the branch will have both a stopping code and a completed simulation time. The stopping code for a branch is a numerical code which represents the reason why this particular branch execution came to an end. These stopping codes are defined in the edit-rules file when a new experiment is submitted. If the stopping code “Completed” is present, it means that this experiment ran to its preset execution time. How the ADAPT-server retrieves this information from the simulator output files must be set up in the wrapper for the particular simulator. If the simulator execution is terminated for a reason other than by running to the preset end time or by a code specified in the edit-rules file it is flagged as “Abnormally Terminated”. The simulation end time is something that is also pulled from the simulator output file and must be accounted for in the simulator wrapper. Non-completed branches will not have stopping times or stopping codes, so instead of this

information, the branch's compute state is displayed (explained in Table 3.1). Finally, all branches have an associated probability and this information is displayed last on all branches. This probability represents a "cumulative" probability; the probability of the scenario to this point. This number does not represent the probability of a single branch.

Since ADAPT is designed to monitor these experiments in real time, there must be a way to determine the status of a branch in the ADAPT-server system. To aide with this, several states can be assigned to branches to describe their situation within the ADAPT database. There are six different compute states that a branch can be in. The branches are color coded to allow for more easy identification. Table 3.1 gives a list of all of the states a branch can be in along with their color code and a brief description.

Branch State	Color Code	Description
Completed	Green	The branch's execution has been completed with a user-defined stopping code
Checkpointed	Dark Blue	The branch has been paused by user action
Running	Cyan	The branch is currently executing
Queued	Yellow	The branch is waiting to be executed in the queue
Insignificant	Orange	The branch's probability falls below the experiment threshold, the branch will not be submitted to the queue
Abnormally Terminated	Red	The branch execution has terminated via some non-user specified reason

Table 3. 1: Description of the different branch state's and their corresponding color codes.

Many of the features of the Dynamic Tree display system can be accessed by right-clicking. If the user right-clicks on a single branch, the following menu is displayed:

Figure 3. 8: Menu shown when user right-clicks on a particular branch.

This menu allows the user to better trace scenario pathways visually in the tree. In this menu the user can select all children of a particular branch (all children that resulted from this branch, all generations) and select the scenario pathways which led to this branch. Selecting either one of these options will paint the affected branches red while painting the selected branch blue. The user can deactivate this feature with the “De-select Children” and “Un-highlight history” options which will become available when the user selects either “Select Children” or “Highlight history”.

3.8.3.1 Event Tree Right-Click Menu

There are also many options available to the user when he/she right-clicks on the dynamic tree display itself (not on any particular branch). The following menu is shown when the user right-clicks on the tree display:

Figure 3. 9: Menu shown then the user right-clicks on the event tree.

This menu provides many different options at the tree-level. The different menus shown in the above figure will be discussed in turn.

The first option (Zoom) allows the user to scale the tree in and out to limit or expand the view of the tree. There are several sub-options provided with this menu. There are the options to scale the tree to pre-programmed scales, namely 200%, 100%, 50%, and 25%. Next, there is a “Zoom to fit” option which will try and scale the tree such that it fits in the displayable screen.

After this, there is the option of a “Custom” zoom. Selecting this option will produce the following dialog:

Figure 3. 10: Custom Zoom dialog window.

This dialog allows the input of user-defined scales. There are two different ways to scale the tree in this dialog. The first shown is a “General Tree Scale”. This will take the number entered in the box on the right and apply this factor to both the x- and y-dimensions of the branches on the tree, scaling each dimension equally. The second scale is the “Time Scale Factor”. This will take the number input into the box on the right and apply this factor to the horizontal dimension only. This feature exists because of the possibility of an uneven distribution of branches with respect to time (there may be sections of very fast branching and sections of very slow branches and a single scale on both dimensions will not allow the user to see all information). For both options, the user can either use the slider bar for both factors (the tree will scale as the user manipulates the slider) or enter the factors manually and end by pressing the “Enter” key.

The next menu provided is the “Branches” menu which has two sub-options, namely the “Branch Settings” option and the “Branch Statistics” option. The Branch Settings option allows the user to change how information is displayed on each branch. A picture of the dialog which is

shown for this menu option is given in Figure 3.11. The dialog gives the user a list of all branching codes, along with a text name for the stopping code (the default is simply the numerical code) which the user can alter, and the color for all branches of this type. If the user changes the text name for a particular stopping code it will be placed on all branches of this type (the text name will replace the stopping code on the completed branches). Finally, the user can select what information can be displayed on branches. The user can choose to display or not display the branch's simulation time, stopping code, stopping code name, database id, or probability.

Figure 3. 11: Picture of the “Branch Settings” dialog.

The Branch Statistics menu item opens a dialog which provides the user with some data and statistics on the overall experiment. The dialog is broken into two tabs, namely

the tabs “Info”, and “Branch Plots”. Pictures of both are shown below in Figures 3.12 and 3.13, respectively. The Info tab provides some basic information about the experiment such as how long it has been running, how many branches are on the tree etc. Also, it gives distributions of stopping codes and compute states on the tree.

Figure 3. 12: Description of first tab of “Branch Statistics” dialog.

The Branch Plots tab allows the user to retrieve information about the event tree in a more visual way. There are currently options to plot here which are provided in a drop down box at the top of the dialog. These options are:

- 1) Branch Simtime Density – This plot will show how many branches are executing at any given time (simulation time). This helps the user to determine where most of the branching is taking place. The user can also create this plot for one particular branching rule to determine where a particular branching rule is making the most impact. The user must also specify a time step. This will determine the resolution at

which the software looks for branches. An example of one of these plots is given in Figure 3.13.

Figure 3. 13: Second tab in “Branch Statistics” dialog.

- 2) Branch Simtime Distribution – This plot gives the distribution of how long each branch executed (in simulation time).
- 3) Branching Rule Tally – This option allows the user to make either a bar or a pie chart which will give the distribution of stopping codes throughout the event tree. An example of one of these plots is shown in Figure 3.14.
- 4) Before and After – This plotting option will give a bar chart of the number and type of branches which precede or follow a particular stopping code on the tree. ALL

stopping codes which precede or follow a particular stopping code will be tallied here. The user has the option here of ignoring insignificant, abnormally terminated, Checkpointed, and those branches marked as completed in the tally. An example of this option is shown in Figure 3.15.

Figure 3. 14: Example of a “Branching Rule Tally” plot.

Figure 3. 15: Example of a “Before and After” plot.

The next “right-click” menu provided for the dynamic event tree is the “End State” menu. Within the End State Menu there are two sub-items: “Consequence Table” and “Download End State Files”. Within the Consequence Table menu there are two options, namely the option to create a new Consequence table or load a previously existing one. A Consequence Table is a GUI tool used to display scenarios, their corresponding probabilities as well as provide a place to enter consequences for this particular scenario. Some examples of consequences could be the fraction of core melt, probability of Steam Generator Tube Rupture (SGTR), final water level in the Reactor Pressure Vessel (RPV), etc. Currently, these values must be input by the user. However, future versions of the code will allow direct manipulation of simulator plot data for automatic generation of consequences. An example of a Consequence Table is shown in Figure 3.16. The naming convention for the scenarios is as follows. As an example, the following is the first scenario listed; scenario 631-/322-310-56-/322-/322-/56. This

character sequence is a string of stopping codes with the “/” character representing a logical NOT. The character string is read from right to left with the earliest stopping code listed first. The scenario should be read NOT Stopping Code (SC) 56, NOT SC 322, NOT SC 322, SC 56, SC 310, NOT SC 322, SC 631. The “NOT” branches refer to a branching point in which a particular branching rule was questioned, but the pathway was taken where this branching condition did not occur (i.e., if the branching rule was questioning whether or not a pipe failure occurred, two branches will be created: one with a pipe failure, one with no pipe failure). The table is also provided with the database id of the final branch in the sequence as well as a place for the user to insert comments. Right-clicking on the table will allow the user to save this Consequence Table. Consequence Table files have the default file extension of .csq.

Scenario	Ending ID	Probability	Comments	Consequ...	Conseque...	Conseque...	Conseque...
631-/322-310-56-/322-/322-/56	209	8.02E-4					
631-322-310-56-/322-/322-/56	210	8.02E-4					
631-/631-310-56-322-/322-/56	221	2.41E-4					
631-310-/310-56-322-/322-/56	218	1.9E-4					
56-631-337-/56-322-/56	234	0.018818					
56-631-/631-/631-337-/56-322-/56	240	1.23E-4					
56-631-/631-337-/56-322-/56	236	7.42E-4					
56-631-310-/337-/56-322-/56	242	9.41E-4					
56-/631-/631-/631-337-/56-322-/56	239	1.23E-4					
56-631-310-/322-/56-/322-/322-/56	212	1.23E-4					
56-631-310-322-/56-/322-/322-/56	216	1.23E-4					
56-631-310-/56-322-/322-/56	226	7.43E-4					

Figure 3. 16: Example of a Consequence Table.

The Download End State Files option provides a means to download all simulator and ADAPT files associated with end state branches. A list of all scenarios, their ending branch

id, a list of available files to download (with their files sizes), and which host the final branch was executed on is provided as a means of selecting which files to download. An illustration of this feature is given in Figure 3.17.

Ending Branch id	Hostname	Branch Code	Scenario	File List	File Size	Download?
209	bm66.bmi.oar.net	631	631-/322-310-56...	adapt-stderr.txt	0 k	☐
210	bm66.bmi.oar.net	631	631-322-310-56...	sbo-nontruncate...	35105 k	☒
212	bm70.bmi.oar.net	56	56-631-310-/322...	adapt-stderr.txt	0 k	☐
216	bm69.bmi.oar.net	56	56-631-310-322...	adapt-stderr.txt	0 k	☒
218	bm65.bmi.oar.net	631	631-310-/310-56...	adapt-stderr.txt	0 k	☐
221	bm67.bmi.oar.net	631	631-/631-310-56...	sbo.cor.tpl	6 k	☒
226	bm65.bmi.oar.net	56	56-631-310-/56...	adapt-stderr.txt	0 k	☐
234	bm64.bmi.oar.net	56	56-631-337-/56...	adapt-stderr.txt	0 k	☐
236	bm63.bmi.oar.net	56	56-631-/631-337...	sbo.ptf	148088 k	☒
239	bm65.bmi.oar.net	56	56-/631-/631-/6...	adapt-stderr.txt	0 k	☐
240	bm63.bmi.oar.net	56	56-631-/631-/63...	adapt-stderr.txt	0 k	☐

Download

Figure 3. 17: Picture of a Download Table.

The table gives (left to right) the id of the final branch in a particular scenario, the next column gives the host where the final branch was executed (and hence the data is being stored), the third column gives the stopping code of the final branch in the scenario, the “Scenario” column gives the text representation of the scenario. The “File List” column gives a list of all files which are available for download for this particular scenario with the next column

being the size of whatever file is selected in kilobytes. The final column allows the user to select which files form which scenarios to download. Once the download choices have been made, pressing “Download” button in the lower left-hand corner of the dialog will initiate the download. The table can also be sorted by any of the columns by clicking on the column header (i.e. to sort the table by branch code click the branch code header).

The next option in the event tree “right-click” menu is the “Dynamic Tree Settings” option. Choosing this option will bring up the following dialog shown below in Figure 3.18. Whenever an option is selected in ADAPT which requires knowledge of what scenarios are in the event tree, the scenarios are calculated and gathered. In general a scenario is a sequence of branches which corresponds to a unique event sequence. However, given that there are branches on the event tree which do not correspond to physical situations (branches that are currently running, branches that are checkpointed, etc.) the user has the option of having ADAPT omit these branch types in the calculation of scenarios. This is done through the Dynamic Tree Settings option. The dialog shown allows the user to select which to include in scenario calculation. The user has the option to include checkpointed, running, queued, abnormally terminated, and insignificant branches. The user can also select to include “All Others”. This option allows the user to ignore from scenario calculation all successfully completed branches (no matter what the stopping code is).

Figure 3. 18: Experiment settings dialog.

The “Export” option will be used to export dynamic event tree data into SAPHIREⁱ. SAPHIRE is a fault tree/event tree PRA analysis code which is utilized by the Nuclear Regulatory Commission (NRC) and is maintained by Idaho National Lab (INL). The concept of this feature of ADAPT has been developed but has not been implemented at this time. Such an addition will be implemented in future versions of the software.

The “Monitor Experiment” option allows the user to choose whether or not to monitor the current experiment for updates. When opening an experiment, the user is given the option to allow for the possibility of updates by polling the database every 10 seconds for additions. Should the user wish to change if the experiment is updated or not having this option

selected (a check will appear next to the option) will monitor the experiment while not having it selected will preclude the code from performing updates.

The final menu which appears in the dynamic tree “right-click” menu is the “Experiment” menu. This has several options which allow the user to control the execution of the experiment and even delete the experiment from the ADAPT database. There are three options in this menu, namely “Checkpoint Experiment”, “Restart Experiment”, and “Delete Experiment”. The “Checkpoint Experiment” branch will pause the execution of any running branches in the experiment. The “Restart Experiment” option will execute all branches that have been checkpointed. Finally, the “Delete Experiment” option will permanently remove the experiment from the database. When this option is selected the user is first prompted to confirm this action since deletion of an experiment is an irreversible procedure.

3.8.3.2 Branch Dialog

Left-clicking on a branch brings up the Branch Information Dialog (See Figure 3.19) which retrieves information about a single branch. This dialog has three tabs. The first tab contains some basic information about the branch, the second tab contains information related to its execution on the compute cluster, and the final tab allows for the plotting of available simulator data.

The first tab, the “Basic” tab, is shown below in Figure 3.19. This tab gives some basic information about the branch that has been selected. It gives the user the experiment which this branch belongs to, the branch stopping code (if it is complete), its ending simulation time (if it is complete), the branch’s individual probability, and the cumulative probability (the

probability of the scenario this branch belongs to up to this point). If the user would like to alter the branch's individual probability, he/she enters a new probability in the "Branch Probability" box and hit the "Change Prob." button. This will save the branch probability to the settings file for this experiment and alter the cumulative probability of it self and any child branches accordingly. Below the first two rows of information there are two lists which give this branch's parents (left) and its children (right). The list of parents is a list of ALL branches which led to this particular branch (the branch's parent, grandparent, etc.). The list of children only lists those branches which are DIRECT descendents of this branch. Next to these lists there is a space for the user to enter comments about this branch. Once the desired comments have been entered, press the "Save Comments" button and the user's comments will be saved in the settings file for this experiment. Changes to the probability and comments are saved onto the user's hard drive and are automatically loaded when this experiment loaded.

Figure 3. 19: Picture of the “Basic” tab of a Branch Information Dialog.

The second tab on the Branch Information Dialog is the “Compute” tab and a picture of this tab is shown below in Figure 3.20. This tab provides information about the branch’s status in the cluster. The branch’s status, it’s execution host (if it has executed), it’s time of submission to the ADAPT server, the time at which it executed (if it has executed), the time at which the branch’s execution completed (if it has completed), as well as the branch’s database id and the branch’s job id are all given here. The branch’s database id is unique to each branch; however, each branch can have many different job ids. Each different job represents an attempt to execute a branch. If a branch was attempted to be executed twice it will still have its

single branch id but will have two job ids associated with it. Finally, the last feature available on this tab is the ability to download individual files associated with this branch. In order to download the desired file, simply click on the desired file and click the download button. The user will be prompted to enter where the file is to be downloaded and then it is added to the Download Manager (discussed below). Only one file can be selected for download at a time. The large empty space on the right hand side of the dialog is to be used for future development and will contain a means of interacting and viewing the standard output of each branch.

Figure 3. 20: Picture of the “Compute” tab of a Branch Information Dialog.

The final tab is the “Plot” tab and a picture of this tab is given below in Figure 3.21. On the left hand side of this dialog there will be a list of all available plot variables for this experiment (it is assumed that all plot variables are the same for every branch in the experiment). In order to plot a variable, select it from the list and click the “Plot” button. Multiple plot variables can be selected at once and they will all be plotted on the same axis. Currently ADAPT does not have access to the units of the plot data so plot units must be entered manually. The “Extract” button allows the user to take the data that has been plotted and save it to a multi-column text file. The “View” button will allow the user to view the data (text form). The “Undock Plot” button will remove the plot from the Branch Information Dialog panel and place it as an internal frame into the main display window.

Figure 3. 21: The Plot Tab in the Branch Information Dialog.

3.8.3.3 Branch Timeline

Since time is an important aspect of dynamic event tree analysis, a graphical timeline has been added to keep temporal perspective within the tree. The timeline will scroll as the user scrolls through the tree and will also scale as the tree is scaled. When the user clicks on the timeline, the time at the current mouse position is shown and a vertical line is displayed on

the event tree from that point. Also, the user can change the spacing of major and minor intervals on the timeline by right-clicking on it and selecting the “Settings” option. A dialog similar to Figure 3.22 is then displayed. This dialog allows the user the option to display the major tick marks, display the times as those tick marks and set the spacing between them. Also, the options are available for displaying minor tick marks, showing times at those tick marks and setting how many minor intervals should be displayed between two major tick marks.

Figure 3. 22: Picture of the Timeline Option Box Dialog.

3.8.3.4 Download Manager

One of the features of the ADAPT code is the ability to download simulator output files from the cluster on which the experiments is being run. This can be done either by using the download table feature or downloading files for individual branches through the

Branch Information Dialog. In both cases downloads are placed into a queue and displayed in the Download Manager. Here the user can view the progress of a single download and see what files are currently in the download queue. A picture of the Download Manager is given in Figure 3.23.

Figure 3. 23: Description of the Download Manager dialog.

If downloads are set up through the Download Table, the files will automatically download to the experiment's main directory and be automatically named based off of which branch the files are being downloaded for. For example if the user wishes to download the file output.txt for branches 100 and 101 which belong in experiment 5, the output.txt file for branch 100 will be saved in <ADAPT_HOME>/e5/100output.txt and the output.txt file for branch 101

will be saved in <ADAPT_HOME>/e5/101output.txt. The automatic naming of files is used to aide the user in downloading multiple (possible dozens to hundreds) files at once without having to place and name each one. When multiple downloads are created from the Download Table, all of them are placed in the Download Manager and downloaded one at a time. If files are downloaded from the Branch Information Dialog, the user is prompted where to place the file and the file shall be named before downloading commences. Multiple files can still be placed in queue when downloading from the Branch Information Dialog.

3.9 Static Tree Module

The second module of the ADAPT GUI, the Static Module, is used for creating and displaying static event trees. The horizontal axes of these event trees do not represent time, instead the horizontal axes represent the branching rules that occur (or top events line the top of the tree). An example of a static event tree is shown in Figure 3.24. In event trees such as these, the probability of these top events are usually determined by fault tree analysis or sensitivity analysis, something which is not provided by ADAPT. It is assumed that probabilities are known by the analyst.

Figure 3. 24: Example static event tree.

The purpose of the Static Tree Module is two-fold. First, this tool is not meant to be a replica of other event tree-based PRA tools such as SAPHIRE, but more meant as an organizational tool for event tree analyses which are coupled with the execution of severe accident simulators for each scenario on the tree. Since some of these analyses could involve hundreds of scenarios, the Static Module will allow for the linking of event tree scenarios and simulator output for better organization. While the Static Module is less developed than the

Dynamic Module, this module will, in the future, allow analysis of these event trees coupled with the analysis of the simulator output data. The second purpose of the Static Module is to provide a means of comparing static tree-type analyses with dynamic event tree analyses. Currently the link between the two modules does not exist. However, the framework has been set up to provide means of comparing event trees created with both modules. Such functionality will be implemented into future versions of the code. The following sections will give more details as to how to use the Static Module in its current state.

3.9.1 Creating a New Static Tree

In order to begin creating a new static tree, select the File menu and choose New- > Static Tree. Selecting this menu option will bring up the dialog shown in Figure 3.25.

Figure 3. 25: The New Static Tree Input Dialog.

Each static experiment requires a name and the name must be unique from any other experiment name already in the database (an error will be issued if a name is entered that already exists). Next, the user can select the option of whether or not to actually save this tree input into the database or only create this tree for one-time use. There are currently two methods of entering static event tree information: 1) input by table and 2) input by file. These two methods require a similar logic for input; they both require a list of top events and a list of scenarios. Consider the example event tree given in Figure 3.24. The list of top events would be as described by Figure 3.26.

Evacuation	Fire Containment	Fire Control
------------	------------------	--------------

Figure 3. 26: Example top events (taken from Figure 3.23).

Note that the initiating event is not considered part of the top events; it is input at another time. Next, the list of scenarios would be:

S1	S2	S3
S1	S2	F3
S1	F2	S3
S1	F2	F3
F1	S2	S3

F1	S2	F3
F1	F2	S3
F1	F2	F3

(with S=success, F=failure)

Figure 3. 27: Example tree input for event tree given in Figure 3.23, note that the section in parentheses is not part of the input.

This input would be sufficient to create a static event tree. Details about each different method of input are given in the next two sections. For each of the methods of input, if the user chooses to save the static tree in the database, a dialog similar the one shown in Figure 3.28 is displayed.

Figure 3. 28: Static tree dialog to enter plot file path and new experiment description

In this dialog the user will specify the super directory where the simulator output data is stored (if the user chooses to). The plot files must be arranged in a particular directory structure in order to be understood by ADAPT. Figure 3.29 shows a diagram of how the directory structured must be arranged.

Figure 3. 29: Diagram of necessary directory structure for static tree plot data to be interpreted by ADAPT.

The directory name which is placed in the input dialog is the path of the “Root Directory” in Figure 3.28. Currently, ADAPT only has the capability to read text files in which the data are in tab or space-delineated columns. However, both the Dynamic and Static Modules may require access to simulator plot data which may not be in text format. Work is currently underway for implementing the ability to read many types of simulator plot files and this feature will be implemented into future versions of the software.

3.9.1.1 Tabular Input

The tabular input option allows the input of scenarios with a table. When the user selects this option and enters the plot data file path and experiment description, he/she is prompted to enter the number of columns and the number of rows for the input table. The number of columns corresponds to the total number of branching points or top events in the tree and the number of rows corresponds to the total number of scenarios. Once this information has been entered, the user will then be prompted to enter the names of the top events as shown in Figure 3.30.

Once the top events have been entered, the user enters the scenario data which will be similar to what is shown in Figure 3.31.

Branching Point A	Branching Point B	Branching Point C
Success	Success	Success
Success	Success	Fail
Success	Fail	Success
Success	Fail	Fail
Fail	Success	Success
Fail	Success	Fail
Fail	Fail	Success
Fail	Fail	Fail

Figure 3. 31: Example tabular input.

The event tree created from the data entered above has the following form as shown in Figure 3.32 below:

Figure 3. 32: Example event tree used for the input in Figure 3.30.

Note that if particular branching rule does not apply for a particular scenario, enter “n/a” at that point in the scenario and no branching will occur at that branching point for that scenario.

3.9.1.2 File Input

The second method of input is file input. When the new experiment dialog is shown, check the “File Input” box and enter the name of the input file in the box or select the “Browse” button to search for the file desired. Just as the tabular input, file input takes a list of top events and scenarios to generate the tree. As an example we will again use the tree in Figure 3.33. The format is a list of scenarios with the branches delineated by tabs. It can be illustrated as follows:

Line 1, all top events: {top event 1}<tab>{top event 2}<tab>...

Line 2, scenario 1: {branch 1}<tab>{branch2}<tab>...

... ..

Line N, scenario N-1: {branch 1}<tab>{branch2}<tab>...

The input for our example tree would look like this:

Branching Point A	Branching Point B	Branching Point C
Success	Success	Success
Success	Success	Fail
Success	Fail	Success
Success	Fail	Fail
Fail	Success	Success
Fail	Success	Fail
Fail	Fail	Success
Fail	Fail	Fail

Figure 3. 33: Example file input for event tree shown in Figure 3.31.

3.9.2 Opening a Previously Created Static Tree

In order to open a previously created event tree which has been saved to the ADAPT database go to File->Open->Static Tree. Choosing this option will open the dialog that is shown below in Figure 3.34.

Figure 3. 34: Open Experiment dialog box for the Static Module.

The dialog to open a static tree has three components to it. First, on the left there is the list of all available projects in the database to open. On the top of the dialog there is a “Status” box. Since probabilities on the event tree have to be entered manually in the current framework, the status of “complete” or “incomplete” refers to whether or not all probabilities

have been entered into the tree. If the status is “incomplete” then one or more of the branches still have their default probability value. Finally there is a description box for each experiment. The user enters a description when the static project is originally created and this description is displayed here. To open a particular project, select it from the list and press the “OK” button.

3.9.3 Event Tree Display

Now that all of the basics of creating and opening static event tree projects have been covered it is now appropriate to delve into the features of the static event tree display system. Like the dynamic event tree display system many of the options are available through right or left-click either on a particular branch on the tree itself. This section will cover the details of how to use the static event tree display.

Unlike dynamic even trees, the horizontal axis of the event tree does not correspond to time. Along the top of these event trees are the top events which dictate where the branchings will occur. As there would be in a classical event tree, the top events are displayed across the top of event tree as the user entered them in the input process. Since these trees may grow larger than the screen can fit, the user can scroll through the tree and the top event markers at the top of the screen will scroll as well as to allow the user to keep perspective while navigating through the tree. An event tree which has been opened in ADAPT is shown below in Figure 3.35.

Figure 3. 35: Description and example of Static Event Tree display.

3.9.3.1 Event Tree Right-Click Options

When the user right-clicks on the event tree display, the menu shown in Figure 3.36 is displayed. There are 4 options here to choose from.

Figure 3. 36: Menu generated when the user right-click on the static event tree

The “Zoom” Menu allows the user to zoom the view of the tree in and out. There is less functionality here than there was in the DET display, the user only has the option of scaling the tree to 200%, 100%, 50%, and 25% of its nominal size. There is no horizontal-axis-only scaling in the static tree system, all scale factors are applied to all dimensions of the tree.

The next option “Find Scenario” allows the user to locate a particular scenario in the tree by its scenario number. Scenario numbers are assigned by their order in the input. This means that the first scenario that is input (either in file input or table input) is scenario number 1, the second scenario input is scenario number 2 and so on. Selecting this option will bring up a dialog which will ask the user for the desired scenario number. When the scenario has been found in the tree the tree is scrolled to that scenario and all branches belonging to that scenario are highlighted in red.

The “Display” option has four sub-options; they are “Probabilities”, “Find Defaults”, “Scenario Probabilities”, and “Distribution”. The “Probabilities” option will display the probability of each branch directly beside it on the event tree display. The “Find Defaults”

option will highlight all branches which still have the default probability of 1.0 as their probability value. All branches which still have the default value will be highlighted in orange. The “Scenario Probabilities” option will display the probability of each scenario just to the right of the final branch in each scenario. All of the above options can be deactivated by selecting the same option again in the menu. The final option “Distribution” will display a pie chart distribution of the final branches in each scenario. An example of this is shown in Figure 3.37.

Figure 3. 37: An example end state distribution plot.

The last option available in this menu is the “Print” option. The static module has a rudimentary printing capability built into it. While printing out the entire tree is not completely

feasible, this functionality built in to ADAPT will allow the user to print sections of the tree. The section of the tree is whatever is being displayed on the static tree display at the time the user selects to print. The print interface is whatever is native to the operating system on which ADAPT is being run.

3.9.3.2 Branch Right-Click Options

The user can interact with individual branches in two ways, either by left-clicking or right-clicking. Right clicking on a particular branch will give a menu similar to that which is found by right-clicking on a dynamic tree branch. A picture is shown in Figure 3.38 for reference.

Figure 3. 38: Menu generated when the user right-clicks on the event tree.

Selecting the “Select Children” option will highlight ALL branches which are produced as a result of this branch (i.e., this does not just select direct children; it selects ALL

children of this branch). The branch itself is highlighted in blue and all of the child branches are highlighted in red. Selecting the “De-select Children” option will deactivate this feature. Selecting the “Highlight History” option will highlight the scenario pathway to which this branch belongs up to that branch. The branch itself will be highlighted in blue while the scenario members will be highlighted in red. In order to deactivate this feature, select the “Unhighlight History” option.

3.9.3.3 Branch Left-Click Interaction

The second way to interact with branches on the tree is by clicking on them. Clicking on a single branch will produce a dialog box similar to the one in Figure 3.39.

The image shows a 'Branch Information' dialog box with a 'Basic' tab. It contains a 'Branch Probability' field set to 1.0 with an 'Update Probability' button. Below this is a list of 'Associated Scenarios' with '16' selected. To the right is a 'Branch Description' text area with the placeholder 'Enter Description Information.'. At the bottom is a table with two columns: 'Branch Point' and 'Result'. The table contains five rows of data. An 'OK' button is at the bottom right.

Branch Point	Result
Reactor Coolant ...	21g/p
SG Depress before...	Y
TDAFW runs until...	N
Valves reclose at ...	n/a
RCS depress afte...	1P,TSC

Figure 3. 39: Static Module branch dialog.

A variety of information is displayed in this dialog. First, atop the dialog the branch probability is displayed. The user has the option of altering the branch probability here by entering a new value and clicking on the “Update Probability” button. Below this, there is a box labeled “Associated Scenarios”. This gives a list of the scenarios that this branch belongs to (the root branch belongs to all scenarios while the final branch in a scenario belongs to only one scenario). Next to this is a place where the user can enter a description of this particular branch. Finally, at the bottom of this dialog there is a table of this history of this branch. This table gives a list of all top events that have been in this branch’s past and the value of the branches which belong to this scenario.

[15] Idaho National Engineering Environmental Laboratory, Lockheed Martin Idaho Technologies Company, “NUREG/CR-6532: Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 6.0”, U.S. Nuclear Regulatory Commission, 1999

[16] “JFreeChart”, <http://www.jfree.org/jfreechart/>, May 2008